Latest drafts and blue

28-C-Pt. 6

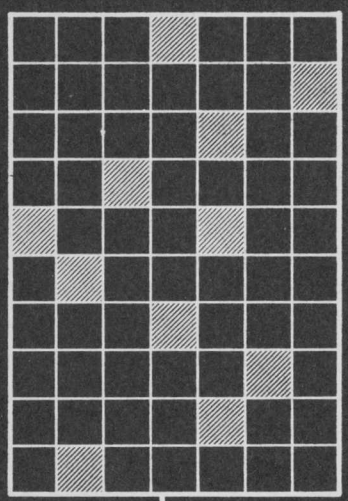
CDC 6600 Control Cards, Chippewa Operating System, KAFB Computing Center

By

J. B. Peterson

UNITED STATES
DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY
Albuquerque, New Mexico

CDC 6600 Control cards Chippewa Operating System KAFB Computing Center



By
J. B. Peterson

New Mexico District
Water Resources Division

UNITED STATES
DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY
Albuquerque, New Mexico

CDC 6600 Control Cards

Chippewa Operating System, KAFB Computing Center

By

J. B. Peterson

Open-file report

Prepared by the New Mexico District, Water Resources Division

July 1969

Contents

Contents - Concluded

CDC 6600 Control Cards

Chippewa Operating System, KAFB Computing Center

By

J. B. Peterson

Introduction

The various control cards used by the Kirtland Air Force Base Control Data Corporation 6600 computer system are described in this report.

This report is compiled from various U.S. Air Force Weapons Laboratory unpublished notes and Control Data Corporation technical manuals. Special thanks are given to Mr. David W. Lane for the overall organization of this report and for writing the VERIFY card instructions. Recognition and appreciation is extended to Major John T. Holland, and Lieutenants Arthur P. Trantolo and John A. Barry, Jr. of the U.S. Air Force Weapons Laboratory, Kirtland Air Force Base, and Mr. Louis Abeyta of Control Data Corporation for their comments and helpful suggestions.

Control Cards

Definition of a Job:

A job consists of one or more central programs which are executed with a common set of data files. JOB CONTROL CARDS are used to identify the programs and data files and to sequence the program executions.

Job deck structure:

Each job deck must begin with a JOB card and end with an END OF FILE card. All JOB CONTROL CARDS must follow the JOB card in the desired order of execution. A RECORD SEPARATOR card follows the last JOB CONTROL CARD. Program and data cards then follow in the order of input requests.

Normal handling of a Job:

In normal operations the JOB CONTROL CARDS, program decks, and data decks are input through one of the card readers to the disk and become the INPUT file for the job. At this time the JOB card is analyzed and the job priority is noted. When this job has top priority of the waiting jobs, it is assigned the next available CONTROL POINT, its first JOB CONTROL CARD is read up and if sufficient hardware is available the operation requested begins. Once a CONTROL POINT has been assigned, the job retains it until all control cards have been executed in sequence, an EXIT condition occurs, or the operator DROPs the job. The job will have access to the central processor according to how its priority ranks with other jobs assigned control points. When a job has completed execution and released its CONTROL POINT its OUTPUT file is released to the printer stack, PUNCH file is released to the punch stack, FILMPL, FILMPR, MF35PL, MF35PR, are released to their respective film stacks. Other disk files are released and lost, except those files declared COMMON.

Handling of Files:

All input and output actions of a central program must involve a named file. This file may be a disk file, a magnetic tape file, a punched card file, or a printer file. Which physical unit is associated with a file name is a function of the ASSIGN control statement (operator response to a REQUEST control card) and is not a function of the central program coding directly. When a physical unit has been ASSIGNED to a job it belongs to that job exclusively. If a file name appears for the first time as the argument of a PROGRAM CALL or PROGRAM HEADER CARD, it is assumed to be a disk file and is assigned by the system to disk storage at that time. Thus, the absence of a REQUEST control statement defines a disk file. It should be noted that only one file mark can be written on the disk. Otherwise, disk files are identical to and interchangeable with tape files in the programming sense.

Naming of Files:

File names must begin with a letter and may have up to seven characters. There are two special names which are implied with each job. These names are INPUT and OUTPUT. The name INPUT refers to the file from which the JOB CARDS were read. This file is the result of loading the card deck for the job. The name OUTPUT refers to the file which ends in printed copy at the completion of the job. These two names must be avoided in assigning names to temporary files and other I/O files. Files will be named through use of Fortran PROGRAM HEADER CARDS.

One file may be equivalenced to another file by:

A.) Define first file. Put file name on program header card.

B.) Equivalence second file. SECOND = FIRST

All references to either file will reference file FIRST.

Example - Equivalence Tape 1 to Input data cards

```
PROGRAM STUFF (INPUT, OUTPUT, TAPE 1 = INPUT)
```

General comments:

a) All control cards start in column "1" and end either with a "." or ")". Any information on the card after the "." or ")" is considered comment and is not used for processing.

b) There are 96 central memory words set aside for control cards. If a control card has N characters, then it will use $\Gamma((N+2)/10)$ words which is the least integer greater then $(N+2)/10$.

JOB Card

Function: The first card of every job is called the JOB card. Its function is project and time accounting.

Note: All numbers on the JOB card are to the base 8 (octal) rather than to the base 10 (decimal).

Format:

A. The mandatory system jobcard format will be:

JBN,PR,TL,FL.NAME,PROJ,ORG,TELE.

(1) JBN Job Initials 3 to 7 characters. The first three must be alphabetic, the rest alphanumeric. The system will use and display the first 5. The system job counter will be two alphanumeric characters (6 and 7). The method of assigning the job initials is as follows: The first three characters are the programmer's initials (must be registered at KAFB before use) the rest of the characters can be anything the programmer wants to use.

(2) PR Priority 1 to 77 octal inclusive. Seven through 30 must meet restrictions on standard priorities.

Standard priorities will be based on field length. These priorities will be assigned by the programmer in accordance with the following.

Priority	Permitted
6_8	Any field length
7_8	FL.GE.70000_8
10_8	FL.GE.100000_8
.	
.	
.	
27_8	FL.GE.270000_8
30_8	FL.GE.300000_8

Non-Standard priorities will be based on (a) the nature of the job or (b) assigned by Chief, Computer Group when required.

Priority	Nature of the Job
1-5	Background Job
60	Express Job
67	Job using display console (having DIS. control card in deck)
70-77	Emergency

(3) TL Time Limit 1 to 77770 octal seconds inclusive.

(4) FL Field length 1 to maximum octal locations. Maximum is 400000_8- resident. Currently, approximately 361000_8. The field must be terminated by a period. Field length may not be omitted, left blank, or punched zero.

(5) Name is the last name of the programmer.

(6) PRØJ is the name of the project at KAFB that computer time is billed to.

(7) ORG is the office from which the deck came.

(8) TELE is the telephone number of the programmer.

B. The optional system job card format will be:

JBN,PR,TL,FL,EFL.NAME,PROJ,ORG,TELE.

The same definitions as above apply with the addition of

EFL extended core storage field length 1-1730

(octal number of 1000 octal word blocks)

If present, must not be blank or zero.

Selection of a job for assignment to a standard control point will be determined by the oldest job of the highest priority that will fit in available core (unless the stack is locked, see 4).

An emergency priority (70-77) in the stack will lock the stack, except for express jobs into express control points, until core is available to run the job.

Express jobs will be selected by express control points using the same criteria as standard CP's. The system will tag a job as express if (a) its priority is .GT. 6 and .LT. 70, (b) its time limit is .LE. 200 and (c) its FL is .LE. 60000. Express jobs will also go through standard CP's.

If the "clear" flag is set the control point will be vacated with zero field length. Otherwise, one pass will be made through the stack attempting to fit a job before any adjustment to the CP's field length is made. If no job is available, the field length will be reduced to 300 locations, this process will be repeated with short waits in between, until a job is found or the clear or error flag is set by the operator.

Because of the time involved in moving ECS and the system tie-ups possible, there will be no intra-job RFL capability provided for ECS.

Example:

```
JBPGBR,6,100,50000.PETERSØN,HYDRØ,USGS,8432246.
```

RUN Card

Function: RUN card is to call the FORTRAN compiler to compile a program.

Format:

RUN(m,fl,cl,bl,i,Ø,ll)

m	MODE
S	Compile and list source program.
G	Compile, no listing, execute.
B	Batch compile with source listing.
C	Batch compile, no listing, execute.
I	Compile, list source code and punch binary cards. (not including library subroutines)
P	Compile, (list source program), punch binary cards complete with all library subroutines.
L	Compile and list source and object code.
D	Compile, list source program, and execute.
A	Compile, list source program if Fortran errors, and execute.

fl Field length of object program in hundreds$_8$, 360000$_8$ maximum; set to job field length on JOB card if omitted, but is mandatory if B or C compile mode is selected.

cl Blank common length$_8$ of object program; set to common length designated in main program if omitted.

bl I/Ø buffer length$_8$ of object program; set to 2011$_8$ if omitted.

i Compiler input file name; INPUT assumed if omitted.

Ø Listable output file name; OUTPUT assumed if omitted.

ll Line limit of job, about a stack of a half an inch of paper (10000$_8$ lines) if omitted, 377700$_8$ maximum.

Example:

```
RUN(S,,,,,,10000)

RUN(D)

RUN(S,,,,TAPE4,,65700)
```

Comments:

a) Normally the programmer will need to concern himself only with the compilation mode (m) and the line limit ll. However, if he is using the AFWLT tape, then he will need to set the input file i.

b) If ECS is being used then RUNEC is the compiler to use. The parameters in the RUNEC card are the same as in the RUN card.

c) Except for the D and A modes, the object code is put out on disk as a file whose name is given on the program header card. For the D and A modes the object code is put only in central memory, and the unused central memory storage returned to the system.

LIST, NOLIST Card

Function: To allow the programmer selective listing of source programs. The option is activated by a card of the following format placed between subprograms in the source deck.

Format:

Column 1

*LIST

*NOLIST

Either option stays on until a new control is encountered by the compiler.

Example:

```
      PROGRAM MAIN
           .
           .
           .
       END
      *NOLIST
        SUBROUTINE A
           .
           .
           .
       END
      *LIST
        SUBROUTINE B
```

SUBROUTINE A will not be listed.

Comments:

Errors encountered while the compiler is running under *NOLIST are handled in the same manner as compiling under A mode. Under no conditions should A mode and this option be used together.

Section 2

Peripheral processor dependent control cards.

Tape REQUEST Card

Function: The functions of a tape request card are: 1) to have a tape drive assigned to your project and 2) set the density of the tape. If the TAPE request card is left out, a disk file is assumed.

Format:

REQUEST TNAME,DEN. LIB LOC OTHER

(1) TNAME is the name of the file, like TAPE 1, TAPE 2, etc.

(2) DEN is the density of the information on the tape.

DEN	DENSITY
LØ	200 BPI
HI	556 BPI
HY	800 BPI

(3) LIB LOC is the library location that is assigned to the tape at KAFB.

(4) OTHER is any other information you want to put on the card for identification purposes.

Comments:

a) All 1 inch tapes are HY density regardless of what is on the request tape card.

b) The LØ density is primarily for the off-line calcomp plotter.

c) Not all 1/2 inch tapes are certified for HY or 800 BPI. Care must be taken in using HY density on 1/2 inch tapes.

d) To minimize operator tape handling, scratch files should go out to the disk. However if on a scratch file you have data intermixed with file marks, you will have to use a physical tape.

e) A maximum of 136 BCD characters can be written on a 1/2 inch tape per logical record.

f) Request tape cards should come after the RUN card (except for A, D, G, and C compile modes). In case there are FORTRAN errors detected during the compilation unused tapes are not mounted.

g) All 1 inch tapes are binary (odd) parity.

Set exit MODE Card

Function: This control statement assigns the arithmetic exit mode setting to the job until further specification. If no MODE card is present the system sets MODE7.

Exit modes are:

MODE1. Address out of range.

MODE2. Operand out of range.

MODE3. Address or Operand out of range.

MODE4. Indefinite Operand.

MODE5. Indefinite Operand or Address out of range.

MODE6. Indefinite Operand or Operand out of range.

MODE7. Indefinite Operand or Address out of range or Operand out of range. (Default mode)

Set SENSE SWITCH Card

Function: This control statement sets the indicated sense switch for reference by subsequent programs within this job.

Example:

```
SWITCH2.
```

Error EXIT Branch Card

Function: This control statement is used to separate the control cards associated with the normal execution of the job from a set of control cards to be executed in the event of an error exit. If this control card is reached by sequential execution of the job, it terminates the job. However, if an error exit occurs anywhere in the job before this card is reached, a control branch is made and the control statements following the EXIT card are executed.

Format:

EXIT.

CEXIT Card

Function: Similar to the EXIT card except only after errors found in the compilation of the program will control pass to the CEXIT card.

Format:

```
CEXIT.
```

RFL Card

Function: To change the amount of central memory storage assigned the job.

Format:

RFL(f1)

(1) f1 is the new field length assigned to your job.

Example:

RFL(42000) which changes the field length to 42000_8 locations.

COMMON File Card

Function: This control statement causes one of two possible actions.

First, if the named file already appears as a local file name for the job, the file is transferred to common status and is available to other jobs when released by the termination of this job.

Second, if the named file has not been previously defined as a local file, a search is made for the file name in the list of common files. If a common file with this name is located, and if the file is not being used by another job, it is assigned to this job until released by the termination of this job. If the file named does not yet appear in the list of common files, or if the file is being used by another job, this job must wait until the file becomes available.

Examples:

```
COMMON ZWRITE.
COMMON DFILE1.
```

RELEASE File Card

Function: This control statement causes the status of the named COMMON file to be changed from COMMON to LOCAL. Thus at completion of the job, the file will be dropped.

Examples:

```
RELEASE ZWRITE.
RELEASE DFILE1.
```

PROGRAM CALL Card

Function: To load a program from the disk to central memory for execution.

Format:

NAME(fa,fb,fc,...,fz)

(1) NAME is the name that appears on the PROGRAM NAME CARD, must start with a letter and be no longer than 7 alphanumeric characters long.

(2) fa, fb, ..., fz are file names.

Example:

CHEAT.

T16AG(TAPE3)

Comments:

a) Parameters are included in the call card when you want to change the name of a file, otherwise the parameter names in the PROGRAM NAME card become the default names.

Section 3

Central processor dependent control cards.

COPY Card

Function: To copy binary information from one file to another to a double end-of-file.

Format:

COPY(fa,fb)

(1) fa is the source file name.

(2) fb is the destination file name.

Example:

COPY(TAPE1,TAPE2); copies from TAPE1 to a double end-of-file to TAPE2.

Comments:

Both files are backspaced over the last end-of-file at the end of copy. Needs 2 K to run.

COPYBF Card

Function: To copy binary information from one file to another.

Format:

COPYBF(fa,fb,nf)

(1) fa is the name of the source file (input assumed if omitted).

(2) fb is the name of the destination file (output assumed if omitted).

(3) nf is the number of files to copy (1 assumed if omitted).

Example:

COPYBF(TAPE1,TAPE2,3); copies 3 binary files from TAPE1 to TAPE2.

Comments:

Needs 2 K to run.

COPYBR Card

Function: To copy binary information from one file to another, record-by-record.

Format:

COPYBR(fa,fb,nr)

(1) fa is the source file name (INPUT assumed if omitted).

(2) fb is the destination file name (if omitted skips nr records on file fa).

(3) nr is the number of records that are to be copied (if nr is omitted one record is copied).

Example:

COPYBR(TAPE1,TAPE2,42); copies 42 binary logical records from TAPE1 to TAPE2.

Comments:

Needs 2 K to run.

COPYCR Card

Function: To copy BCD information from one file to another, record-by-record.

Format:

COPYCR(fa,fb,nr)

(1) fa is the source file name (INPUT assumed if omitted).

(2) fb is the destination file name (OUTPUT assumed if omitted).

(3) nr is the number of BCD records to copy (if nr is omitted one record is copied).

Example:

COPYCR(TAPE1,TAPE2,37); copies 37 BCD records from TAPE1 to TAPE2.

Comments:

Needs 2 K to run.

COPYCF Card

Function: Copy BCD information from one file to another.

Format:

COPYCF(fa,fb,nf)

(1) fa is the name of the source file; if left out becomes INPUT.

(2) fb is the name of the destination file; if left out becomes OUTPUT.

(3) nf is the number of files to copy from fa to fb. If nf = 0 copying is continued until there is a double end-of-file mark. If nf is left out one file is copied.

Example:

COPYCF(TAPE1,TAPE2,3); copies 3 files from TAPE1 to TAPE2.

Comments:

Needs 2 K to run.

COPYSBF Card

Function: To copy a binary file of coded information, shifting each line one character and adding a space.

Format:

COPYSBF(fa,fb)

(1) fa is the source file name (INPUT assumed if omitted).

(2) fb is the destination file name (OUTPUT assumed if omitted).

Example:

COPYSBF(TAPE1,TAPE2); copies 1 file from TAPE1 to TAPE2, shifting a character for each line.

Comments:

Suppose you have a 1/2 inch BCD tape (TAPE6) with card images and you want to list the tape. If you use COPYCF(TAPE6,OUTPUT,1) to list the tape every time a "1" appears in column one you will get a page eject, or if you have "+" in column one, the line will get printed over, etc. Regardless of what is in column one it will not get printed.

If you try COPYSBF(TAPE6,OUTPUT) you will get nothing but parity errors and junk.

What you can do is copy TAPE6 to a disk file, this gives you a binary mode file. Then copy the disk file to the OUTPUT file to list it.

```
COPYCF(TAPE6,DISK,1)
REWIND(DISK)
COPYSBF(DISK,OUTPUT)
```

are control cards that will work.

COPYSBF needs 12 K to run.

VERIFY Card

Function: To compare two binary files word-by-word to a file mark and give indication of differences.

Format:

VERIFY(fa,fb,n,fo)

(1) fa is one file name (Default FILE1).

(2) fb is the other file name (Default FILE2).

(3) n is number of comparison failures to be listed per record (Default 1).

(4) fo is the output file name (Default OUTPUT).

Example:

VERIFY(TAPE1,TAPE2, 5) which compares one binary file on TAPE1 and TAPE2 and prints the first 5 words that fail to compare in each record.

Comments:

Output from VERIFY has the following forms.

a) Header: THIS IS VERIFY (TAPE1,TAPE2, 5)**

b) No Errors: VERIFY OK

c) Word Compare Errors:

RECORD NUMBER WORD TAPE1 TAPE2

3(000003) 8(000010) 04271456413356343333 26052211063100000552

d) Unequal Number of Words in Record:

324(000504) EXCESS WORDS IN RECORD 10(000012) OF FILE TAPE2.

e) Unequal Number of Records in File:

3(000003) EXCESS RECORDS IN FILE TAPE1.

f) Any error causes the message:

VERIFY FAILURE DROP. OR GO.

and lets the operator either continue the job or abort it.

Messages b) and f) are also entered in the job's DAYFILE.

SKFIL Card

Function: To skip over a specified number of files.

Format:

SKFIL(fn,nf)

(1) fn is the name of the file.

(2) nf is the number (decimal) of files to skip on fn,

$0 \leq nf < 4096_{10}$

Example:

SKFIL(TAPE1,2) skip 2 files on TAPE1.

Comments:

a) The control card SKFIL(TAPE1,2) has as its FORTRAN counterpart

CALL SKFILE(5LTAPE1,2)

BKSP Card

Function: To backspace over records.

Format:

BKSP(fn,nr,mode)

(1) fn is the file name.

(2) nr is the number of records to backspace

nr is decimal

(3) mode = B for binary records

= C for BCD records

Example:

BKSP(TAPE1,3,C); backspaces 3 BCD records on TAPE1.

Comments:

a) The control card BKSP(TAPE1,1,C) would have as its FORTRAN counterpart BACKSPACE 1.

REWIND Card

Function: To rewind or reposition the file to the load point or beginning.

Format:

REWIND(fa,fb,fc)

(1) fa is the file name of the file to rewind.

(2) fb is another file to rewind.

(3) fc is another file to rewind.

Can take up to three arguments.

Example:

REWIND(TAPE1,TAPE2) which rewinds TAPE1 and TAPE2.

Comments:

a) The control card REWIND(TAPE1) would have as its FORTRAN counterpart REWIND1.

UNLOAD Card

Function: To rewind and unload the tape from the tape drive.

Format:

UNLOAD(fa,fb,fc)

(1) fa is the file name of the tape to unload.

(2) fb is another file name of a tape to unload.

(3) fc is another file name of a tape to unload.

Can take up to three arguments.

Example:

UNLOAD(TAPE3,BILL) which unloads TAPE3 and BILL.

Comments:

a) To speed up operations, UNLOAD cards should be used at the completion of the run to unload tapes you have used.

b) The control card UNLOAD(TAPE3) would have as its FORTRAN counterpart

CALL UNLOADS(3) or

CALL UNLOADS(5LTAPE3)

c) If UNLOAD or UNLOADS is used before termination of the job it should be followed by RETURN or RETURNS (see next page).

RETURN Card

Function: To release files no longer needed by your job.

Format:

RETURN(fa,fb,fc)

(1) fa is the name of a file to be released.

(2) fb is the name of another file to release.

(3) fc is the name of another file to release.

Can take up to three arguments.

Example:

RETURN(TAPE1) which returns the file TAPE1 to the system.

Comments:

a) Suppose you do not want a listing of your program,

RUN(S)

RETURN(OUTPUT)

would throw away everything on the OUTPUT file that you had accumulated to that point.

b) The control card RETURN(TAPE3) would have its FORTRAN counterpart

CALL RETURNS(3) or

CALL RETURNS(5LTAPE3)

CLSFRN Card

Purpose: To assist the computer operators in identifying the printer output of a classified run. A call to this routine produces two consecutive pages of printout, each of which contains in large letters the following:

```
* * * * * * * * * * * * *
*     CLASSIFIED        *
*                       *
*          RUN          *
*                       *
*  (Classification)     *
* * * * * * * * * * * * *
```

Format:

CLSFRN(A) where

A = C for a classification of CONFIDENTIAL

A = S for a classification of SECRET

A = T for a classification of TOP SECRET

Example:

```
JOB CARD
  OTHER CONTROL CARDS
CLSFRN(C)
RUN(S)
WHAM.
CLSFRN(C)
  OTHER CONTROL CARDS
7/8/9
            PROGRAM WHAM (OUTPUT)
              .
              .
              .
              .
              .
            END
7/8/9
6/7/8/9
```

Comments:

a) This routine should be called twice in a classified run, once before the compiler call control card (RUN card) and once after the program execution control card. This will produce a two-page classified run identification before the program listing and a two-page classified run identification after the program printer output (but before the dayfile).

b) CLSFRN requires a minimum of 3400 storage locations.

INDEX Card

Function: INDEX analyzes FORTRAN source decks and produces for each routine in the deck a complete cross reference index that gives: (1) All statement numbers and variable names used in the routine and, (2) All references to the statement numbers and variable names. The statement number and variable name list is numerically and alphabetically sorted. The references are given by type: (e.g., "PR" for an appearance of a symbol in a PRINT statement, "EQ" for an EQUIVALENCE statement, etc.)

By using INDEX, one can immediately determine such things as: Unnecessary statement numbers; whether a variable appears in a common statement or not; where a variable is printed or read by the program; where other statements refer to a numbered statement; and so forth.

Examples:

DECK SETUP FOR USING INDEX WITH A PROGRAM ON CARDS.

```
  STR,6,500,44000.STRANGELOVE,571000,WLW,3517.
INDEX.
7/8/9
     (your source deck)
  6/7/8/9
```

DECK SETUP FOR USING INDEX WITH A PROGRAM ON THE AFWL TAPE.

```
  STR,6,500,44000.STRANGELOVE,571000,WLW,3517.
  DAFWL.
  INDEX(TAPEZ)
  7/8/9
  (AFWL CONTROL CARDS.)
  6/7/8/9
```

Comment:

INDEX requires 44000 octal core locations for use on the CDC-6600.

Section 4

Note: Control cards found in section 3 will destroy the exchange jump package area.

DMP Card

Function: To print out blocks of memory and some registers.

Format:

DMP.

DMP(ca)

DMP(ca,cb)

(1) DMP. dump out the exchange jump area and a selected portion of memory where the job came off.

(2) DMP(ca) dump out the memory location from 0 to ca.

(3) DMP(ca,cb) dump out memory locations ca to cb.

Example:

DMP(1200) which prints out the first 1200 words in memory.

DMPEC Card

Function: To print out blocks of ECS.

Format:

DMPEC(A1,A2,A3,A4)

(1) A1 = FIRST WORD address in octal (MANDATORY PARAMETER)

(2) A2 = LAST WORD address in octal (MANDATORY PARAMETER)

(3) A3 = DUMP MODE (0 or 3 = OCTAL DUMP,Ø20)

(1 = REAL DUMP,E20.5)

(2 = INTEGER DUMP,I20)

A3 is optional and preset to 1.

A4 = ECS DUMP OUTPUT file name.

A4 is optional and preset to OUTPUT.

Caution:

DMPEC is a central memory program and will destroy portions of central memory within a job's field length. If central memory dumps are desired, all calls for DMP must appear prior to any DMPEC call.

Example:

DMPEC(100,2000,2) which prints out ECS words 100 through 2000 in INTEGER FORMAT.

Comments:

Needs 7 K to run.

Section 5

Examples.

Sample Control Card Set-ups

No physical tapes, with a source listing.

```
1)  JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.
    RUN(D)
    7/8/9
        PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
              .
              .
              .
    7/8/9
        data
    6/7/8/9
```

No physical tapes, without a source listing.

```
2)  JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.
    RUN(A)
    7/8/9
        PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
              .
              .
              .
    7/8/9
        data
    6/7/8/9
```

The next example is similar to 1).

3) JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.

RUN(S)

NOTAPE.

7/8/9

```
      PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
            .
            .
            .
```

7/8/9

      data

6/7/8/9

If the object length of NO TAPE is 41000 words in example 1 the storage would be cut down to 41000 words for execution but example 3 would not. To get the same result then:

4) JBDNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.

RUN(S)

RFL(41000)

NOTAPE.

7/8/9

```
      PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
            .
            .
            .
```

7/8/9

      data

6/7/8/9

Another way of getting the same result as example 2) is:

```
5)  JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.
    RUN(S)
    RETURN(OUTPUT)
    RFL(41000)
    NOTAPE.
    7/8/9
         PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
              .
              .
              .
    7/8/9
         data
    6/7/8/9
```

Assume that the program NOTAPE had been written for card input, but for some reason the data is on a tape which you call DATAT. Without any changes to the program the program will run with the following control cards:

```
6)  JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.
    RUN(S)
    RFL(41000)
    REQUEST DATAT,HI.     XY 982     1/2 INCH  556 BPI
    NOTAPE(DATAT)
    UNLOAD(DATAT)
    EXIT.
    UNLOAD(DATAT)
    7/8/9
```

Assume now that the OUTPUT of NOTAPE is going to be used as input to another program and therefore you want to save the output on a tape, examples 7) and 8) show how this may be done.

7)
```
JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.
RUN(S)
RFL(41000)
RETURN(OUTPUT)
REQUEST OUTPUT,HY.  XX982    1 INCH    800 BPI
NOTAPE.
UNLOAD(OUTPUT)
EXIT.
UNLOAD(OUTPUT)
7/8/9
     PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
          .
          .
          .
7/8/9
     data
6/7/8/9
```

8) JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.

RUN(S)

RFL(41000)

REQUEST DATAOUT,HY. XY 982 1 INCH 800 BPI

NOTAPE(,DATAOUT)

UNLOAD(DATAOUT)

EXIT.

UNLOAD(DATAOUT)

7/8/9

```
      PROGRAM NOTAPE(INPUT,OUTPUT,TAPE1)
            .
            .
            .
```

7/8/9

data

6/7/8/9

Another way of getting the same result is:

```
9)  JBPNTP,7,300,72000.PETERSON,HYDRO,USGS,8432246.
    RUN(S)
    RFL(41000)
    REQUEST DATAOUT,HY.     XY 982     1 INCH     800 BPI
    NOTAPE.
    UNLOAD(DATAOUT)
    EXIT.
    UNLOAD(DATAOUT)
    7/8/9
         PROGRAM NOTAPE(INPUT,DATAOUT,TAPE1,OUTPUT = DATAOUT)
              .
              .
              .
    7/8/9
         data
    6/7/8/9
```