BNL 18733
AMD 663

BROOKHAVEN NATIONAL LABORATORY
ASSOCIATED UNIVERSITIES, INC.
under contract with the
UNITED STATES ATOMIC ENERGY COMMISSION

INFORMAL REPORT

APPLIED MATHEMATICS DEPARTMENT

March 1974

BROOKHAVEN COMPUTER USER'S GUIDE

by

K. Fuchel, S. L. Padwa,
A. Scott and C. Stewart

BROOKHAVEN COMPUTER USER'S GUIDE

K. Fuchel
S. L. Padwa
A. Scott
C. Stewart

January 1974

[illegible]

BROOKHAVEN COMPUTER USER'S GUIDE

Table of Contents

1.0 INTRODUCTION

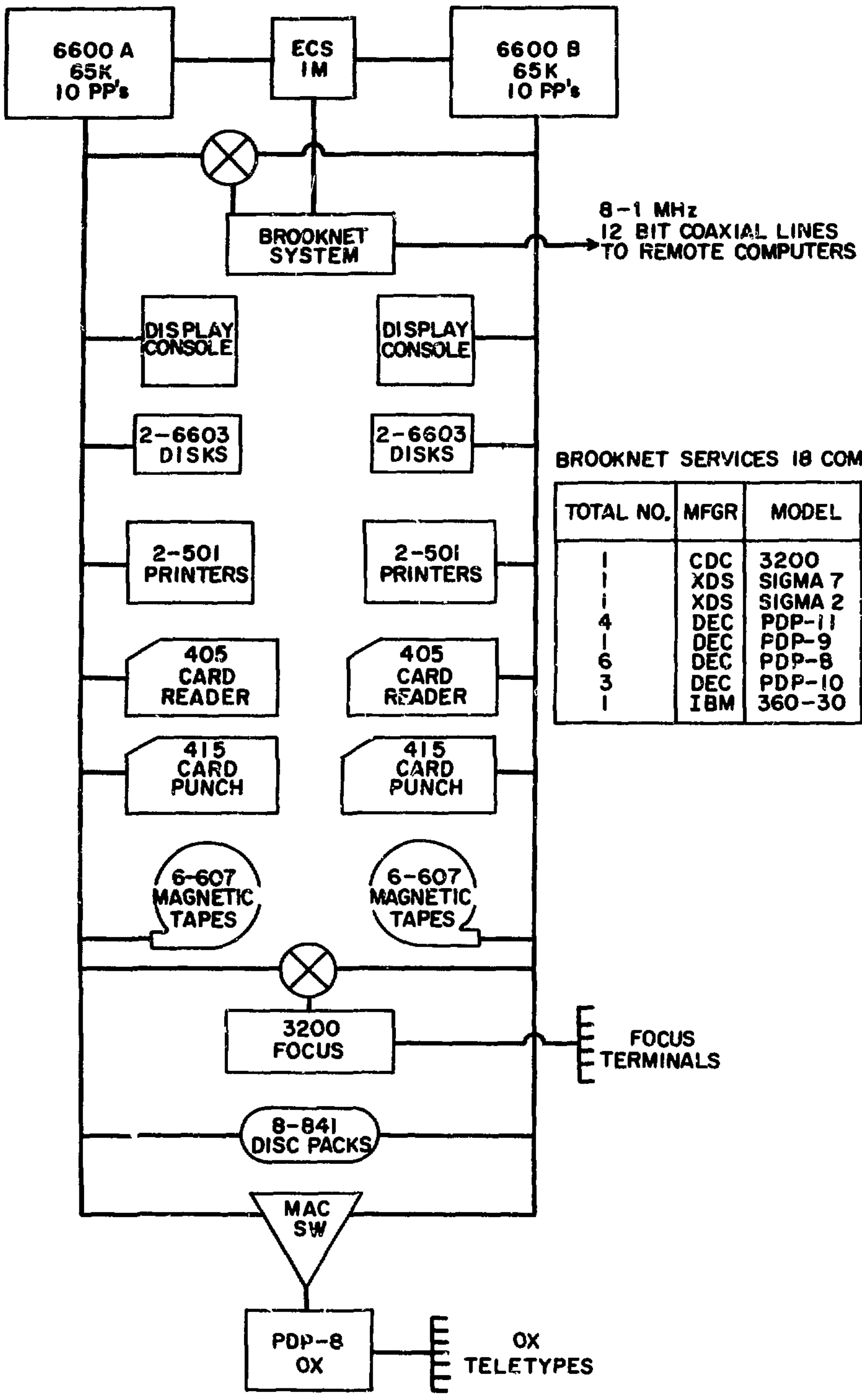
This User's Guide is intended to satisfy several needs. Primarily it is intended to give a survey of facilities and services available at the Central Scientific Computing Facility (CSCF) and to serve as a guide to their usage. A second goal is to make available in one moderate-size manual the information which a new user needs to get started, as well as to serve as a reference for the Brookhaven Computing Community. We intend this to be a working document, continuously being updated and improved. The last page is a form for suggesting improvements, pointing out errors, or communicating to the Applied Mathematics Department any other comments that might make this Guide more useful.

Two Control Data Corporation 6600 computers provide the primary computational facility at the CSCF. Each of the CDC 6600's has 65536 words of 60-bit central memory, two 6603-II disk drives each with a capacity of 75 million characters, six 7-track magnetic tape drives, two 1000 line per minute printers, and card reader/punch equipment. The two computers share one million words of Extended Core Storage (ECS), and eight 841 disk-packs with a total capacity of 250 million characters. The operating system in use in SCOPE 3, heavily modified by BNL. (The original version of this system is described in Control Data 6400/6500/6600 Computer Systems Reference Manual, Publication No. 60189400.) A more detailed description of the CDC 6600 hardware and software can be found in Chapter 8.

A CDC 3200 computer, wholly dedicated to running the FOCUS remote access system, is linked to one of the 6600's. This system allows creation and maintenance of files from keyboard terminals, either hard-wired, or over telephone lines, and the submission of jobs to the 6600's. A network of remote computers is also linked to one of the 6600's over the BROOKNET system; some of these computers are used as remote batch stations. A PDP-8 runs an operator extension (OX) facility which can be used to query the system as to a job's status.

Figure 1-1 depicts the CSCF configuration.

Open shop programming is the rule: the Applied Mathematics Department offers introductory and advisory assistance to users who are developing their own programs. Programming services are available, but there is a charge for these services.



BROOKNET SERVICES 18 COMPUTERS AS FOLLOWS:

TOTAL NO.	MFGR	MODEL	EQUIPMENT LOCATION
1	CDC	3200	MATH
1	XDS	SIGMA 7	PHYSICS
1	XDS	SIGMA 2	MEDICAL
4	DEC	PDP-11	BIOLOGY-PHYSICS
1	DEC	PDP-9	ACCELERATOR
6	DEC	PDP-8	MATH,CHEM,AGS,BIO
3	DEC	PDP-10	AGS
1	IBM	360-30	ADP

Figure 1-1: CSCF Configuration (March 74)

2.0 GUIDE FOR NEW USERS

2.1 INTRODUCTION

The Central Scientific Computer Facility (CSCF) is located in Building 515, 61, Brookhaven Avenue, adjacent to the offices of the Applied Mathematics Department.

The CSCF consists of the computer room, the Ready Room, and the Computer Information & Assistance (CI&A) office. Theoffice of the computer librarian is in room 1-35 of the adjacent wing of the AMD.

The Ready Room can be used for program preparation and analysis. The following equipment is available to users:

- -keypunch machines and verifiers
- -punch card reproducers and interpreters
- -an IBM 407 for listing Hollerith cards
- -a card sorter
- -a magnetic tape cleaner
- -film scanner for 35 mm film such as can be generated by the Calcomp plotters
- -an OX teletype which may be used to query the system about a job's status
- -a number of teletypes for the FOCUS remote job entry system
- -a paper tape reader which works in conjunction with a FOCUS teletype
- -a BROOKNET remote I/O terminal consisting of a PDP-8 with card reader and printer.

The ready room can be used as a temporary storage area for punch cards provided that the cards are placed in a metal carrying case. Shelves are provided for these cases at the user's own risk.

The CI&A office supplies general information on the CSCF operation and system, as well as programming assistance. The CI&A telephone extension is 4176. Suspected hardware or software failures should be reported to the CI&A or to the CSCF shift supervisor when the CI&A office is closed.

The Computer Library contains the documentation for programming at the CSCF. Manuals and their updates are issued by the librarian whose telephone extension is 4144. Several subroutine libraries are available, such as the International Mathematical and Statistical Library (IMSL), a small VIM library, a Scientific Subroutine Package (SSP) and subroutines written at BNL (SBNL). Descriptions may be obtained from the librarian.

Submission of jobs to the CDC 6600 can be via:

- -card deck submitted over the counter to be read in by a computer operator
- -self-service card reader located in the ready room

-Brooknet remote batch stations, one of which is located in the ready room
-the FOCUS system which also allows the creation and editing of files from teletypes

Note that card decks must be punched using the IBM 026 character set. Decks punched on an IBM 029 must be converted. A program, CARDCON, is available for this purpose.

User tapes (see section 2.5) will be accepted by the operations staff, and these tapes will be made available to the user's programs.

Output will be filed alphabetically in the CSCF output area, or delivered to the user's building if so requested. Output remoted to a Brooknet terminal must be picked up directly by the user.

2.2 PROBLEM NUMBERS

All jobs submitted to the CDC 6600's must have a valid problem number for accounting purposes. Each number is assigned in response to an authorized requisition for computer time. For detailed problem number information, call AMD, extension 4105.

2.3 JOB CARD FORMAT

The BNL job card is normally brown and has thirteen fields, each of fixed length in a fixed location on the card. The fields are separated by heavy black lines, one card-column wide, printed on the face of the card. The format and use of each field on the job card are given below.

Please note that a job will be aborted if embedded blanks or illegal characters (8 and 9 are illegal characters in octal fields) are found on its job card.

Field 1: Job Number Columns 1-7

The job number consists of two parts, a fixed 4-character problem number and a 3-character user number. The first character of the problem number is a department identifier, as follows:

Department	Identifier(s)
Accelerator	A
Applied Mathematics	Y, Z, V, W, M
Applied Science	N
Biology	B
Chemistry	C
Instrumentation and H. P.	I
Medical	D
Physics	P

Department	Identifier(s)
Reactor Operations	R
Laboratory Administration Divisions	E
Non-BNL	X
Special Account for High Energy Physics	Q

The remainder of the 4-character problem number consists of a three digit number which is assigned usage by the AMD on the computer request form.

The 3-character user number consists of any alphabetic or numeric information that could uniquely identify jobs using the same problem number. Some departments use the first character to identify the individual user, and he uses the remaining two for sequencing his jobs.

Field 2: Priority and Type Column 9-11

The first two columns in this field contain the right justified priority number for the job. The following matrix shows how the priority is used to select jobs and how the selection is affected by what mode the machine is in. A zero indicates the category of jobs is transparent to the selection algorithm.

		Internal Priority			Order of Selection		
Job Card Priority	Time Limit	Prime Mode	Normal Mode	Standby Mode	Prime Mode	Normal Mode	Standby Mode
16	Any	71XX	71XX	71XX	1st	1st	1st
10	<1000sec	42XX	42XX	42XX	2nd	2nd	2nd
10	≥1000sec	500	4000	4000	0	3rd	3rd
7	<1000sec	22XX	2000	22XX	3rd	4th	4th
7	≥1000sec	300	2000	2000	0	5th	5th
1	Any	100	100	100	0	0	6th

Priority 16 is for RUSH jobs. These jobs begin execution immediately and their output is processed as soon as possible.

Priority 10 is for jobs requiring RAPID turn around. During PRIME time, RAPID jobs requiring less than 1000_8 seconds are run ahead of similar NORMAL jobs. During NORMAL time, RAPID jobs, even if they require more than 1000 seconds, are run ahead of all NORMAL jobs.

Priority 7 is for NORMAL jobs. Most of these run during PRIME mode; the exception being jobs requiring more than 1000_8 seconds which are held off until NORMAL mode.

Priority 1 is for STANDBY jobs. These jobs are run on a first come first serve basis during STANDBY mode. The charge rate for these jobs is one fifth the rate for NORMAL jobs.

All other priorities are undefined and should not be used. Note that a higher job card priority will result in a higher charge rate for the job as described on page 7-2.

In the table of internal priorities, XX is determined by the time limit and maximum field length. Lower time limits and smaller field lengths result in higher internal priorities.

For most jobs, internal priorities are incremented every 10 minutes. For long (>1000 seconds) RAPID and NORMAL jobs, internal priorities are incremented once per hour. For STANDBY jobs, priorities are updated once a day. Thus, time spent in the input queue is a factor in job selection.

The third column in this field is reserved for an alphabetic suffix to the priority number. This one-letter suffix, the job type, will permit more efficient job scheduling. It gives the system some indication of the relative usage of peripheral processor (PP) and central processor (CP) time for a job. At present, jobs fall into four types according to the following definitions:

Type	Definition
A	PP time less than 1/2 CP TIME
B	PP time approximately 1/2 to 2 times CP time
C	PP time approximately 2 to 4 times CP time
D	PP time greater than 4 times CP time.

Because of variations in the PP/CP time ratio between successive runs of a job, a user can only give a suitable estimate by selecting the appropriate job type designation. Type B or C should be used when running a job for the first time. See p. 7-2 for further details.

Field 3: Time Limit Columns 13-16

Maximum central processor (CP) running time in seconds (octal) right adjusted in the field. Note that 100_8 seconds is approximately one minute. Jobs with small time limit are assigned higher priority than long running jobs. A job which exceeds its time limit is aborted.

Field 4: Field Length Columns 18-20

Maximum number of central memory locations in (octal) thousands used at any time during the job, right adjusted in the field.

Field 5: Line Limit Columns 22-23

The maximum of printed output for the job in (octal) thousands of lines, right adjusted in the field. A 77 or * in this field is an exception and allows nearly $1,000,000_8$ lines. The line limit is not used for scheduling purposes. It provides protection against jobs running wild producing reams of unwanted output at your expense.

Field 6: Number of Tape Drives Column 25

The maximum number of tape drives (0 through a maximum of 7) required at any one time by the job. A REQUEST for a tape, followed by a RETURN, and a later REQUEST, requires just one drive. Blank is interpreted as 0. A job which requests more tapes than are reserved is aborted.

Field 7: Number of Files Column 27-28

A BK in this field is used to restrict job to execution in the BROOKNET machine. If this is not relevant, the field should be left blank.

Field 8: ECS Field Length Columns 30-33

The maximum ECS field length in thousands (octal) of words needed for direct user controlled access by the job. Blank is interpreted as 0.

Field 9: Blank Columns 35-40

This field is completely unspecified and should be left blank.

Field 10: Name and Extension Columns 42-67

The name of the user submitting the job and his telephone extension must be punched in this field.

Field 11: Punch Column 69

Punch an X in this field if you want your punch output saved.

Field 12: Plot Column 71

Punch an X in this field to avoid possible loss of plot output.

Field 13: Building Number Columns 73-80

If the job is to be delivered, punch a D in column 73, and your building number and a U or D if necessary to indicate upstairs/downstairs in columns 74-80.

Fields 1, 2, 3, 4, 5, 10 must be punched for all jobs. Fields 6, 7, 8, 11, 12, 13 should be punched when appropriate but need not contain an explicit zero.

2.4 DECK STRUCTURE

The definition of a job at the BNL CSCF begins with a brown job card, and terminates with an orange End-Of-File card, which contains a 6-7-8-9 multipunch in column 1. Any information between these two cards is entirely up to the user. Records are separated by a card with a 7-8-9 multipunch in column 1. The first record of a job is the control card section, which orders the logical flow of the job.

ELEMENTARY EXAMPLES OF DECK STRUCTURES

1. Compile only, obtaining a binary deck for each subroutine as well as the main program.

```
Job card          (FL = 45000)
RUN(P)
7-8-9  record separator
       PROGRAM SIMPLE(INPUT,OUTPUT)
       .
       .
       .
       END
       SUBROUTINE XYZ(A,B,N)
       .
       .
       .
       END
6-7-8-9 end of file
```

2. Load object decks and execute, with data

```
Job card          (FL = whatever job requires)
INPUT.
7-8-9 record separator
    - binary decks -
7-8-9 record separator  N. B. binary decks
7-8-9 record separator  terminate on a double
                        record separator or an
                           end of file, if no data
                     - data -
                6-7-8-9 end of file
```

3. Compile, list only, request a tape and execute, with no data

```
Job card
RUN(S)
REQUEST, TAPE12, 8, X, R, A123.BRØWN,J.
LGO.
7-8-9 record separator
      PROGRAM THIRD(OUTPUT,TAPE 12)
      .
      .
      .
6-7-8-9 end of file
```

4. Compile a subroutine, load object decks and execute with no data. In case of an error, dump first 5000_8 locations of the program.

```
Job card
RUN(S)
LOAD(INPUT)
LOAD(LGO)
EXECUTE.
EXIT.
DMP(5000)
7-8-9  record separator
       SUBROUTINE PQR
       .
       .
       .
       END
7-8-9  record separator
    binary decks -
6-7-8-9 end of file
```

5. Compile and execute program using two data decks

```
Job card
RUN(S)
LGO.
7-8-9 record separator
       .
       .
       .
       1st Data Deck
       .
       .
       .
7-8-9  record separator
       .
       .
       .
       2nd Data Deck
       .
       .
       .
6-7-8-9 End-of File
```

6. Compile and execute two unique programs

```
                    Job card
                    RUN(S)              Compile 1st Program
                    LGO.                Execute 1st Program
                    RETURN(LGO)         Drops Object Deck File
                    RUN(S)              Compile 2nd Program
                    LGO.                Execute 2nd Program
                    7-8-9
                                  .
                                  .
                                  .
                                  First Fortran Program
                                  .
                                  .
                                  .
                    7-8-9
                                  .
                                  .
                                  .
                                  Second Fortran Program
                                  .
                                  .
                                  .
                    6-7-8-9
```

2.5 MAGNETIC TAPES

2.5.1 MAGNETIC TAPE TYPES -A-Series

There are several types of tapes used at the CSCF. The most common type is the A-series tape. These tapes are issued by the CSCF and never leave the computing area. A-series tapes provide temporary storage for CSCF users. Only data which can easily be restored should be placed on A-series tapes.

Reel numbers are assigned to users in the CSCF. The reel numbers are then used on the tape REQUEST control cards in the users' deck. To write on an A-series tape, the user must place his name on the REQUEST control card.

2.5.2 MAGNETIC TAPE TYPES - Non Standard

Many users have the need to bring information to or from the CSCF. One of the easiest methods to do this is by the use of magnetic tape. The CSCF provides (temporary) labels for user tapes. These tapes are called non-standard tapes. The non-standard label is placed on the tape and the tape canister. A non-standard tape is submitted at the same time as the user's job. An operator will log-in and store the tape in bins provided for these tapes. Since there is a limited amount of space for non-standard, tapes users should transfer the information from a non-standard tape and write it to an A-series tape. The non-standard tape can be removed from the CSCF and used as backup for the A-series tape.

2.6 FILE NAMES

Certain names are reserved for special files and should not be used as names for programs and subroutines. Among these are:

INPUT

This file contains the job. The first record consists of the job's control cards. By the time the job gets into execution, the file is positioned past this record. The remainder of the file consists of programs and data records.

OUTPUT

Every job is assigned an OUTPUT file which will eventually be printed, unless it is sent to a remote station. Before the OUTPUT file a page consisting of the Job name (in large letters) and the date, time, system and machine is printed. The user's dayfile, a chronological record of the job, is appended to the OUTPUT file at job termination.

PUNCH/PUNCHB/PAB

These files are similar to OUTPUT but are directed to the card punch instead of the printer. PUNCH is for BCD cards, PUNCHB is for binary object decks, PAB is for full 80 column binary.

PLOT/FILM

There are no post-processing functions to handle these files; use of these names should be avoided.

NULL

This is the name of a dummy file which is never written. Thus COPYBR(A,NULL,4) merely skips four records on file A.

Note: Avoid using file names identical to system programs such as REWIND, RUN, COPY, etc.
If such a file is created it will not be possible for the program to use the corresponding system program.

2.7 Non BNL Users

The policy of Brookhaven National Laboratory is to make its general purpose Central Scientific Computing Facility (CSCF) available to non-BNL users in accordance with A Guide to the Use of Brookhaven National Laboratory Central Scientific Computing Facility by Other Members of the Scientific Community (October 1973), available from the AMD.

User of the CSCF must be authorized in advance by the Associate Director for Administration.

3.0 CSCF SERVICES AND PROCEDURES

The Central Scientific Computing Facility includes a great deal more than computers. Auxiliary equipment, such as key-punch and card reproducing machines, are available; a library and technical services are provided for the user community. This chapter describes these aspects of the Computing Facility.

3.1.1 Auxiliary Equipment

Considerable equipment in the Facility is available to users on a self-service, first-come-first-served basis. Users should avoid tying up this equipment for periods longer than ½-hour, and they should turn off the equipment when finished. The Computer Facility staff stand ready to provide enthusiastic and learned assistance to anyone experiencing difficulty using this equipment. An inventory follows...

CONTROL DATA 405 Card-reader	--	This is used to input jobs to the CDC 6600. Directions for its use are attached to the reader.
OX-system teletype	--	This returns information about jobs in the CDC 6600. Directions for its use are attached to the teletype.
FOCUS-system terminals	--	4 model 35 units are available in the CSCF. There is also an Infoton terminal adjacent to the Paper Tape unit.
FOCUS Paper Tape Facility	--	For reading and punching paper-tape under control of the FOCUS system. Instructions for its use are contained in the FOCUS User Manual, and in Paper Tape Facility User's Guide, by A. Harris & H. B. Chiang, available from the computer librarian.
BROOKNET Remote Terminal	--	REMOTE #15, a PDP-8 computer giving access to the BROOKNET system (see Sec. 6.5). and including a model 33 teletype, Soroban Card-reader, and Mohawk line-printer. Instructions for the terminal's use are available at the terminal.

IBM Unit-Record equipment	--	026 and 029 key-punches 519 reproducers 557 interpreters 083 sorter 059 verifier.
KYBE mag tape cleaner	--	for cleaning magnetic tapes. Instructions are on the unit.
CALCOMP film viewer	--	for examining CALCOMP CRT output.

Additional equipment in the Computing Facility is available through the operations staff. The CALCOMP graphics/plotting facility, including the models 580 and 780 paper plotter and model 835 CRT film plotter, is available as scheduled and operated by the staff. Instructions for preparing data and submitting a request for this equipment are contained in the document "835 Plotting System", available from the CSCF Library. The high-speed CONTROL DATA line printers are operated by the staff, as are the 6600 magnetic tape units and the FOCUS unit-record equipment.

3.1.2 Consumables, Data Storage

The things incidental to computer use are also available to CSCF users. The Center keeps on hand a store of "A-series" magnetic tapes which may be assigned to users possessing a bona fide Job Number. These tapes are assigned for a one year period, may be protected against unauthorized (but not inadvertent) use, and are permanently stored in the CSCF. The user may cancel an A-tape assignment before the assignment year is up; after the year is up the user will be notified by the CSCF staff. The A-tape storage facility is not intended as a data archive but rather as a temporary haven for the currently active data files and to spare the user the inconvenience of having to carry around a lot of tapes between computer jobs. Long term data storage, and the security of "essential" data, is the user's responsibility, and is to be accomplished by means other than the A-tape facility.

The user is warned that tapes may develop defects and become unreadable (particularly under heavy use). It is suggested that users test their tapes (using the available tape testing programs) both upon being issued a new tape and whenever the contents of the tape may be destroyed.

To this end, tapes may be purchased from Central Stores (Cat. #D-94000). Tapes thus possessed by the user are not handled in the Center as A-series tapes, but must enter the CSCF as an "N-tape". An "N-tape" label must be affixed to the tape for this purpose. These labels are available from the CSCF staff. The staff, for its convenience, maintains a limited storage for these N-tapes, together with a catalog of its contents. These tapes will be held in the facility at the discretion of the staff, and if the tape is not used for a reasonable period of time it will be returned to its owner.

Similarly, metal card carriers may be purchased from Central Stores (Cat. #S22100), and limited storage for these carriers is provided in the CSCF. Except for the general provision against insurable loss, the CSCF assumes no responsibility as to the integrity of this storage. Carriers stored in this facility must be properly labeled with the owner's name, address, telephone extension, and the date of last use of the box. Boxes may then be stored for thirty days from that date, beyond which the box will be returned to the user. Cards stored in paper-board boxes are as dust - be warned.

The Center limits its liability with respect to the storage of THINGS and DATA to reasonable care and handling; write protection is offered for A-tapes; a careful catalog is kept for N-tapes under its roof. The ultimate guardian of a user's data is the user himself; only the user is a judge of the "irreplacibility" of his records, and such records should never be left under the Facility's care.

Computer printer output is distributed by the staff as it comes off the printers. Every reasonable effort is made to assure that this output gets to its destination. But this product, like sand at the beach, is voluminous and tends to pile up — it is in everyone's interest that paper output be kept to a minimum and disposed of with dispatch.

3.2 CSCF Information Services

3.2.1 CSCF Library

The CSCF Library is a central archive for documents of interest to users. It contains a store of manuals for use with CSCF equipment and software, several Program Libraries, and serves as a catalog facility for the wide range of programs submitted by users. The library's collection of manuals span a range of manufacturer's publications, especially those from Control Data, but including some from IBM and others. These manuals include all the hardware and software documents needed in the normal course of working in the CSCF. Of these there is an "open distribution" set comprising documents available free to users. Other manuals from the library must be signed for, and returned.

There are several Program Libraries available from the CSCF Library, including the SCOPE operating system, Compilers, FORTRAN and COBOL function libraries, and numerous standard utility programs.

These are stored on magnetic tape as UPDATE (see Control Cards, Chapter 4) Program Libraries, and are available through the OBTAIN control card (see p. 4-23). Of particular interest to users will be the 6600 User Program Library (called VIM for 6000 in Roman Numerals), the IBM SHARE Subroutine Library (called SSP), the BNL Program Library (called SBNL), the International Mathematical Subroutine Library (called IMSL), the U of C Biomedical Statistics Programs Library (called BMD), and the NATS Eigenfunction Library (called NATS). Documents describing these libraries with instructions on how to obtain copies of the routines contained in them are available in the CSCF Library. The Library also keeps on hand documents describing these programs and their use and care.

Users are encouraged to swell the Libraries resources by submitting any programs of their own which they feel would be of interest to others. The main requirement in this regard is an adequate write-up. The write-up should at least describe the routine, indicate how it works, what it does, how it is to be used, any restrictions governing its use, and how and where its source code may be obtained.

3.2.2 Bulletin Board

A Bulletin Board is maintained as a vehicle for providing timely information about the CSCF, its equipment, new features in the operating system and utility software, and known, suspected, or reported problems. This information is posted daily.

3.2.3 Computer Information and Assistance Office

The Computer Information and Assistance (CI&A) Office, which is located in the CSCF ready-room, serves as a channel of information in the inverse direction — from the user back to the Facility. This channel exists primarily to allow the user to alert the staff to malfunctions in the hardware or software supported by the CSCF. The software in this case includes the CDC 6600 and CDC 3200 operating systems, utility programs for both systems such as FOCUS, Compilers, Assemblers, Update, and the like.

There are system programmers within the Applied Math department who are only too eager to search out and destroy bugs in these programs. However, the mere fact that a user program is in some way functioning differently today than it did yesterday is insufficient evidence to unleash these "experts" upon a probably innocent piece of software. The claim of malfunction with respect to "system" routines must be supported with documentary evidence, including, but not limited to, core dumps, all dayfile messages, user source code with input data, and much learned argument. In a great many cases a user's anxiety concerning "system" routines may be eased by consulting the Bulletin Board, where he may compare his problem with the "known bugs" list. There he may find his misery listed.

The CI&A Office also stands ready to advise users with respect to (known) changes in the operating system of utility routines, and the impact of these changes on the user's work in the Facility. Obviously a user problem may be due to a change in one of these products; although these changes would be listed on the Bulletin Board, their ultimate effect is sometimes hard to predict (and sometimes surprising!). Bear in mind, however, that while a changed routine is now working differently than before, and possibly working a hardship upon some users, it is also working better than before. Hence the descriptive title "Up-Grade".

Finally, the CI&A Office is charged with consulting services to the user community. This mandate is specifically directed toward helping new users get familiar with a new (for them) computing environment. Users who have brought themselves and their code from another installation will have many subtle difficulties awaiting them, and the CSCF will endeavor to offer real assistance as well as sympathy. However, the office does not exist to offer a wholesale debugging service; to the extent that a user becomes less a "new" user, so does the sympathy part dominate the consulting equation.

3.3 Programming and Consulting Services

Users whose requirements go beyond the basic responsibilities of the CI&A office may wish to make use of the CSCF Computer Programming Service. The user should note, however, that a separate charge is made for such services. Programming Services will provide help in using the Facility and its services as well as assistance in implementing and debugging code for computers falling within its area of responsibility. In addition, consultant services are available with respect to mathematical and Programming techniques and problems.

In particular, Programming Services is willing to offer the type of assistance detailed below.

1. The design, programming, testing and documentation of entire application software systems or packages for use with the main CSCF computers or other computers as appropriate and feasible.

2. The programming and testing of user-defined application programs and the subroutines and the investigation of programming and mathematical procedures or algorithms necessary or expedient for meeting user needs.

3. The conversion of existing programs written for other computers, equipment configurations or operating systems to function properly in the Brookhaven CSCF environment. The modification, extension or re-writing of such programs for increased capability or efficiency or to meet special user needs.

4. The investigation of causes of run failures for existing programs and assistance in correction or circumvention of the problem.

5. Consultation on the use of the Facility and its procedures, general aspects of programming, numerical analysis, CSCF library subroutine use, special problems in existing programs, and suitability (or otherwise) of computing equipment and programming techniques to specific user-defined projects.

The use of Programming Services requires that a separate Programming Service Requisition be filled out and signed by an appropriate individual having (fiscal) account authorization.

This authorization is in addition to the Computer Usage Requisition which must be completed before the user can make use of the CSCF computers. Outside (non-BNL) users should consult A Guide To The Use of Brookhaven National Laboratory's Central Scientific Computing Facility for procedures to be followed. Both outside and BNL prospective users of the CSCF Programming Services are requested to discuss their plans or requirements with the appropriate CSCF personnel before filling out any firm requests for programming assistance. This is essential for acceptance and scheduling of the request. In its turn, the Facility will make every effort to provide realistic estimates of the cost and time involved in the project.

4.0 CONTROL CARDS

4.1 INTRODUCTION

A job consists of one or more central programs which are executed with data files. Control cards are the first logical record of the input file; they identify the programs and their data files; and sequence program execution.

Each job must begin with a job control card which is described in Chapter 2. All control cards, and only control cards, must appear between the job card and the first record separator (A record separator card contains a 7-8-9 multi-punch in column 1.)

Control cards start in column 1 with the control card name. If there are parameters, they are separated from the name by either a comma or a left parenthesis; the parameters are separated by commas. The parameter field is terminated by a period or a right parenthesis. The terminator must be present even if there are no parameters.

Information to the right of the terminator is treated as comment, and may extend up to column 80. Consequently, only one control statement per card is permitted. Note that COMMENT and REQUEST control cards have significant information to the right of the terminator.

Imbedded blanks should be avoided. Control card names should not be used as names of user programs.

Control cards are normally executed in sequential order. The only exceptions are EXIT, CXIT and FIN cards which are used to specify processing after an error.

Control card names represent three basic types of operations:

1) A command to the system to set up a condition or to take some action;
2) A command to load and execute a file attached to the user's control point;
3) A command to load and execute a system utility program, or a library program.

The control card scan proceeds in the above order; i.e. the system will first see if the name is one of the special system action ones, then search the file name table for a file of matching name attached to the user's control point, and, finally, search for a matching name among the utility and library programs.

Fortran callable equivalents exist for many of the system action control cards.

4.2 SYSTEM ACTION CONTROL CARDS

RFL - (REQUEST FIELD LENGTH)

The memory requirements of a job, referred to as the job's Field Length (FL), can vary for each phase of execution. The FL specified on the Job Card is the maximum that any step of the job requires. Field Lengths are expressed in octal thousands, e.g. 40K means 40000_8 = 16384. Typically, a compilation may require 44K, the job execution 70K, and a file copy 5K. Certain control cards require a minimum field length; this is indicated in the description of the control card as FL=nK.

Field length management is automatic and there is generally no need to use an RFL card. The loader reduces the field length to the minimum required, and nearly all system utility routines do likewise. An RFL card turns off the automatic field length management. However, the field length for both Central Memory (CM) and Extended Core Storage (ECS) may be changed during the execution of a program. The format of the card is:

RFL (C=X,E=Y) where C=X specified a CMFL of X (octal) and E=Y specifies an ECSFL of Y (octal). The default is CM, i.e. RFL(X) sets the CMFL to X.

The new field lengths may not exceed those specified on the Job card.

REDUCE

After the automatic field length management has been overridden by an RFL card, the FL management may be restored:

REDUCE (C) restores automatic management of CMFL,
REDUCE (E) restores automatic management of ECSFL,
REDUCE (C,E) or REDUCE. restores both kinds of FL management.

MODE

MODE,n. or MØDE(n)

This control card may be used to change the arithmetic exit mode. n is a single octal digit.

The arithmetic exit mode determines which arithmetic error(s) will cause an "ARITHMETIC ERROR" error exit. (See the description of the EXIT control card for a discussion of the consequences of an error exit.)

There are basically three types of arithmetic errors, "modes" 1, 2, and 4. Combinations of these are described by the sum or "inclusive or" of the respective numbers. Definitions of the three types of arithmetic errors are given below.

The arithmetic exit mode normally is set at 7. In other words, the 6600 is primed to recognize all three types of arithmetic errors and their combinations. If the arithmetic exit mode were set to 0, no arithmetic errors would be recognized. If the arithmetic exit mode is set to a value between 0 and 7, some but not all arithmetic errors would be recognized. For example a "MODE, 3." card would permit mode 4 errors to go unrecognized.

Changing arithmetic exit modes from 7 to some other value is a dangerous procedure. In some cases very reasonable-looking results may be generated which are wrong or misleading.

Definitions of the Types of Arithmetic Errors

Mode 1 is "address out of range", i.e. an attempt to reference a location outside of the job's field length. This type of error usually arises because of an improper subscript for a dimensioned variable, or because of failure to include a needed subroutine.

Mode 2 is "operand out of range". This occurs when a floating point arithmetic unit (add, multiply, or divide) has received an infinite operand.

Mode 4 is "indefinite operand". A floating point arithmetic unit (add, multiply or divide) has attempted to use an indefinite operand.

A table of infinite and indefinite operands is given below. This table makes use of the fact that twenty octal digits will completely describe a 60-bit 6600 word. The letter "X" in the table means "any octal digit".

SPECIAL OPERAND FORM	REPRESENTATION (20 octal digits)
PLUS INFINITY	3777X... X
MINUS INFINITY	4000X... X
PLUS INDEFINITE	1777X... X
MINUS INDEFINITE	6000X... X

Infinite and indefinite operands usually arise because of a division by zero.

The arithmetic error does not occur when the infinite/indefinite is created but when it is used. Consider the following FORTRAN code -

```
A = B/0.0
C = A*B
```

It is second line of code that causes the error.

A different use of MODE controls the type of messages put out on the job's dayfile.

MODE (PARTDF)	is the default option
MODE (FULLDF)	causes <u>all</u> dayfile messages concerning the job to be output; this is intended for tracing possible problems due to rollout/rollin.
MODE (OFFDF)	suppresses all dayfile messages until job termination, or the next MODE card. After job aborts, the MODE reverts to PARTDF or FULLDF, whichever was last in effect.

SWITCH

SWITCH,n. or SWITCH(n)

This control card changes the pseudo sense switches for reference by a subsequent Fortran program; n = 1-6. The settings are preserved at the control point and copied to location 0 of the user's FL for use by his central program. All switches are initially off. Thus the first SWITCH(n) card sets switch n; a subsequent card clears it. Switches may be set and cleared by control card, Fortran call, and console commands.

Example:

```
SWITCH,6.
```

COMMENT

COMMENT. comments

This control card places the contents of the card on the display scope, and halts the job until the operator types "GO." Inasmuch as operators are not normally stationed at the console, this card should <u>never</u> be used except in the rare case where special operator action is required before a job may proceed. Should this be necessary, prior approval of the shift supervisor must be obtained.

In the event that it is necessary to give the operator special termination instructions, the null control card should be used. This card has a period (.) in column 1, followed by the comments. It does not halt the job, but it is available for operator inspection provided it is repeated at least once per nine control cards.

Example:

```
.SET SENSE SWITCH 6 TO TERMINATE JOB
```

REQUEST

Any file not specifically requested by control cards is assigned by the system to storage on a disk unit. A job need not request card reader and printer for normal input/output since its cards are already stored in the job input file on disk, and output for a printer is sent to the job output file on disk.

Since control cards of a job are processed in order, equipment assignment must be made before the corresponding file is referenced.

This control card is used to request the operator to assign to the file named "LFN" the appropriate magnetic tape. Card format is

REQUEST, LFN,$\left\{\begin{matrix}2\\5\\8\end{matrix}\right\}$,$\left\{\begin{matrix}X\\Y\end{matrix}\right\}$,$\left\{\begin{matrix}R\\W\end{matrix}\right\}$,nnnn. user's name.

The 2, 5 or 8 refers to tape density settings of 200, 556 or 800 bpi.

X or Y refers to the type of tape; tapes written under SCOPE3 may be type Y and must then be read as Y-tapes. Tapes generated on other computers or under a different operating system and tapes intended to be read elsewhere must be type X. R or W refers to "read only" or "write"; a job which attempts to write on a Read only tape, will be aborted. nnnn is the reel identification number.

Examples:

```
REQUEST,TAPE1,5,X,R,A123.   SMITH.
```

Density for TAPE1 is 556 bpi, type is X, read only.

```
REQUEST,TAPE2,8,Y,W,B1542.  DØE, J.
```

Density for TAPE2 is 800 bpi, type is Y (can only be read on a Scope operating system and the tape may be both read and written. While waiting for operator action, the jobs FL is automatically reduced to 100_8 words.

ASSIGN

With a few exceptions, the system assigns files to Extended Core Storage (ECS) as long as there is space available. However, it may sometimes be advantageous for a user to force a file to reside on a specific device. Short files perform better on ECS, files with parallel activity can have conflicts reduced if they reside on different disks.

The format of the control card is:

```
ASSIGN,LFN,DT.
```

where LFN is the file name, and DT the device type.

The device type can indicate a class of devices (e.g. mass storage), or a specific device (disk packs No. 4). The permissible entries for DT are:

A	any mass storage device
*AC	any 6603-II disk
*AZ	any 841 disk pack pairs
*AR	ECS (these files should be short, high activity ones)
*PF	any device acceptable for Catalogued files
NN	where NN is a two-digit number specifying the Equipment Status Table ordinal of the device. Permissible values are: 01 = first 6603 02 = second 6603 03 = ECS 04, 05, 06, 07 = each of four 841 disk pack pairs.

Examples:

```
ASSIGN,TAPE1,*AC.          assign TAPE1 to a 6603
ASSIGN,QNAME,05.           assign file QNAME to the 841
                           corresponding to EST ordinal 05.
```

A more comprehensive write-up is available from the Computer Library.

ACCESS

If a file residing on magnetic tape is to be used frequently, it is desirable to use ACCESS instead of REQUEST. ACCESS stages the tape file onto disk and catalogues it, so that on a subsequent ACCESS it is immediately available.

Example: To replace the REQUEST card

```
REQUEST,TAPE2,5,Y,R,A1234.   SMITH.
```

with an ACCESS card, so that the file will be taken from disk if it is already there, or from tape if not, use

```
ACCESS(TAPE2,5,Y,R,A1234,TPC-SMITH)
```

If TPC-SMITH is omitted, the name from the Job Card will be used.

ACCESS offers many options; for a full description see Computer Library Document No. 1168.

GØ

This control card allows the user to ignore tape parity errors, and to continue processing. The format of the card is:

$GO(f_1=n_1,f_2=n_2,\ldots,f_k=n_k)$

where

f_i = tape file names

n_i = maximum number of allowable unrecovered read errors on file f_i

If n_i is omitted, the default value will be 10. If n_i is set to zero, the GO condition will be turned off. If n_i is ≥ 63, it will be assumed that all errors are to be ignored.

The intent of this control card is to allow users to ignore parities in non-critical information and yet protect themselves against mistakes. Unjustified use of n=63, just like setting your job's line limit to infinity, defeats the whole purpose of flexible controls.

COMMON

COMMON,filel. or COMMON(filel)

This control card has two effects:

1. If the file name, filel, has "common" status in the File Name Table (FNT) and is not being used by another job, it is assigned to this job. If the file is being used by another job, this job must wait until the file becomes available.

If the file name, filel, does not have "common" status, the job must wait until common status is assigned.

2. If the file name, filel, already appears as a local file name for the job, the file will be assigned common status in the file name table (FNT) and becomes available to any succeeding job after this job is finished.

The common status assigned to a file may be terminated by a RELEASE control card.

A word of caution: common files are not retained over dead starts which normally occur at least once every 24 hours.

RELEASE

RELEASE,file1. or RELEASE (file1)

With this control card, the common file named file1 currently assigned to this job will be dropped from common status and assigned local status in the File Name Table (FNT).

Example: RELEASE,BFILE.

The common file, named BFILE, attached to this job is changed to type local so that it will be dropped at the end of the job.

See also RETURN.

RETURN

RETURN(file1,file2,...,filen)

The RETURN control card is used to release files and their equipment when they are no longer needed by a job. the RETURN card applies to files on both disk and tape. Returning a tape file makes the tape transport available to another user (or the same user, but for a different file). Returning a disk file makes the file available to others if it is a catalogued or common file.

CXIT. EXIT. FIN.

The CXIT and EXIT control cards are used to separate the control cards to be processed in the event of a compilation or execution error, respectively, from the control cards associated with normal processing. For example, a user may normally wish to obtain a dump following an execution error, but not after a compilation error.

There are several types of execution errors:

1.	TIME LIMIT.	Job has used all the central processor time it requested.
2.	ARITHMETIC ERROR.	Central processor error exit has occurred. A dump of the exchange jump package plus an area around the stop location is automatically written on OUTPUT.
3.	PPU ABORT.	PP has discovered an illegal request, such as an illegal file name or request to write outside job field length.

4.	CPU ABORT.	Central program has requested that the job be terminated.
5.	PP CALL ERROR.	Monitor has discovered an error in the format of a PP call entered in RA+1 by a central program (can occur if a program accidentally writes in RA+1, as can condition 3).
6.	OPERATOR DROP.	Operator has requested the job be dropped.
7.	CONTROL CARD ERROR.	
8.	ECS PARITY ERROR.	
9.	COMPILATION ERROR.	

The FIN control card is used to separate the control cards which will be processed regardless of the presence or absence of errors from the control cards associated with normal processing. Following a compilation from tape, for example, the tape should be returned whether or not the compilation was successful.

When a compilation (or execution) error occurs, an error flag is set and a search is made for CXIT (or EXIT) or FIN control cards. When one of these cards is found, the error flag is cleared and normal control card processing resumes with the following card.

If a CXIT or EXIT card is encountered when the corresponding error flag is clear, a search is made for the next FIN card and control card and processing resumes from there. The job terminates when the end of control card record is found.

If an execution error exit takes place because of a TIME LIMIT, the user is allowed a limited amount of additional time. Consequently, the control cards following an EXIT card should perform tasks in descending order of importance. In general the limited amount of time available will permit the user to write tape mark(s), rewind and unload tape(s), and dump a reasonable amount of core.

Example:

```
Job Card
RUN(S)                              call the compiler
REQUEST,TAPE3,5,Y,R,A9999. NAME.    If compilation OK, get tape
LGO.                                Load and go
CXIT.                               If compilation error, go to...
PEANUTS.                            ... this program is for FOCUS
                                    users
EXIT.                               If execution error, go to...
DMP(10000)                          ... here, and dump 10K of memory
FIN.                                No matter what, come here, and...
INPUT.                              ...do something, say, another
                                    program
```

TAPESET

This control card is used to specify continuation tapes when handling a multi-reel tape file.

For those users who do not use the preferred RETURN, re-REQUEST method, the following procedure may be used:

Immediately before the REQUEST for the first reel, continuation tapes should be specified using the control card:

```
TAPESET,FIRST,NEXT1,....,NEXTN.
```

where FIRST is the reel on the REQUEST, and the continuation tapes are NEXT1 through NEXTN in that order. If more than one TAPESET card is needed, the FIRST field on the following card should be the last reel mentioned on the previous TAPESET card.

Example:

```
TAPESET(A123,B456,C789)
TAPESET(C789,Z60)
REQUEST,TAPE1,8,R,Y,A123.
```

specifies that the sequence of reels to be mounted is A123, B456,C789,Z60.

A Fortran callable equivalent, TAPSET, exists, and should be used before a Fortran REQUEST call.

Example:

```
CALL TAPSET(4LA123,4LB456,4LC789)
CALL REQUEST(...)
```

RNT (REDUCE NUMBER OF TAPES)

Column 25 of the Job Card contains the number of tapes reserved by the job. This is the maximum number of tapes the job will use at any one time. If this maximum number is not required throughout the whole job, the RNT card should be used to reduce the number reserved. The format of the card is:

RNT(N) which reduces the number reserved to N, or RNT. which reduces it to zero. An attempt to increase the number of tapes reserved will abort the job.

4.3 LOAD AND EXECUTE

LOAD

The LOAD card directs the operating system to load a file.

```
LOAD(File1)
```

The system searches the File Name Table (FNT) for the file, and if it does not appear there, the system searches the system library.

All types of loading, normal, segment, or overlay, may be loaded by LOAD cards but they may not be mixed in one operation. One job may contain any number of LOAD cards. The type of loading for all LOAD cards within a job is determined from the first record of the first named file.

The loader will rewind any file, except the INPUT file, before loading it. Any disk file will be loaded up to a zero length logical record or an end-of-file indicator. A zero length logical record is created on the INPUT file by a pair of record separator cards (7/8/9 in column 1). An end-of-file indicator is created on the INPUT file by a file separator card (6/7/8/9 in column 1).

Once the loading process has been initiated by a LOAD control card, it must not be interrupted. The only cards that can immediately follow a LOAD card are:

1) Another LOAD card,
2) An EXECUTE card,
3) A NOGO card,
4) A MAP card,
5) A NOREL card, or
6) A program call card.

Any of these cards will continue the loading process. An attempt to use some other type of control card immediately after a load card could destroy the program already loaded.

EXECUTE

The execute card completes loading and transfers control to a program.

EXECUTE(name,$p_1 p_2$,....p_i) or EXECUTE.

name	Entry point of the program. If name is absent, control will be transmitted to a name in a table generated either by the compiler (from a FORTRAN PROGRAM card)or by the assembler (from an COMPASS END card).
p_i	Parameters that are passed to the program to be executed.

Completion of normal loading includes:

1) Filling all unsatisfied references with entry points from the system library or with out-of-bounds references, and
2) Writing a load map for the program on the job's output file. Program execution then begins at the entry point named on the EXECUTE and/or in the table generated by the assembler or compiler.

When segments are loaded, unsatisfied references are not filled and execution begins in the first segment. Subsequent segments may be loaded by user calls in the program. When overlays are loaded, execution begins in the main overlay. Subsequent overlays may be loaded by user calls in the program.

NOGO

NØGØ.

This card completes the loading operation, but bypasses execution.

When a NOGO card appears, the loader completes loading of the present program by satisfying external references wherever possible and writing the memory map on the output file. Since execution is bypassed, the NOGO card is generally used to produce a memory map for the purpose of finding loading errors and/or execution field length.

PROGRAM CALL

The format of this load-and-go card is:

```
NAME(name1,name2,...,namen)      or NAME.
```

NAME is the name of the system or user program being called; namei can be the names of all files referenced by program NAME, or a string of parameters to be passed to the program.

The file, NAME, is loaded, and control is transferred to the entry point of the main program included in file NAME.

```
Examples:  RUN(S)
           COPYBR(TAPE2,TAPE3,100)
           LGO.
```

As can be seen from the above examples, most of the control cards discussed in this chapter are specific examples of program call cards.

The program call: NAME. is approximately equivalent to:

```
LOAD(NAME)
EXECUTE.
```

However, it should be noted that while LOAD loads the complete file, PROGRAM CALL does not load routines with the same names as those already loaded by LOAD cards. The user is notified, however, when a "duplicate" routine is not loaded.

The above feature of the (PROGRAM CALL) card may be used to simplify the inclusion of new versions of old routines. Thus if "OLDDISK" is a disk load-and-go file of a complete program and "NEWSUB" is a disk load-and-go file of a new version of a particular subroutine, then

```
LOAD(NEWSUB)
OLDDISK.
```

will substitute the new version prior to execution.

The program call card, like LOAD, will rewind any file except INPUT prior to loading.

The arguments of a Program Call can be substituted for those specified on the FORTRAN PROGRAM card. For example: A FORTRAN program starts with:

```
PROGRAM XYX(INPUT,OUTPUT)
```

i.e. it will read data from the INPUT file, and write output to the OUTPUT file. If the loading of this program (after compilation) is done with the card:

```
LGØ(TAPE1,TAPE2)
```

the program, with no modifications, will read input from TAPE1 and write output onto TAPE2.

MAP

This control card governs the storage map generated by a load sequence. The options are:

MAP(ØFF), the system default, produces no load map.
MAP(ON) produces a full load map
MAP(PART) suppresses the entry references portion of the load map
MAP(ERR) will cause the full load map to be written to a file called MAP instead of to the OUTPUT file, so that the user can opt to print it only if an error occurs. For example:

```
RUN(S)
MAP(ERR)
LGO.
EXIT.
REWIND(MAP)
COPYBF(MAP,OUTPUT)
```

SELECT

Compilers produce a load and go relocatable binary deck residing on some file, usually LGO. The loader loads such a file, then attempts to load any missing routines from the system library. It is sometimes desirable tc be able to take the missing routines from some other file, possibly one supplied by the user. SELECT prepares a load and go file by taking one such file and adding to it any missing routines from another. It can also prepare an index of such a file, which speeds up subsequent uses of SELECT, and it can produce a storage map.

The format of the control card is:

```
SELECT(MODE,I=FILE1,L=FILE2,O=FILE3,X=FILE4)
```

where *MODE* = B, D or N, any of which may be preceded or followed by an M to indicate that a map is required.

I=...	specifies the load and go file (default, I=LGO)
L	specifies the user's library file, where missing routines are to be found, (default L=LIB)
O	specifies the file to which the storage map is to be written (default O=OUTPUT)
X	specifies an alternative execution file. (For overlay jobs, the default is X=XEQ)

The three modes accomplish the following:

<u>B mode</u>: Builds a directory of the relocatable routines on the file specified by L, and saves the directory and the routines on file I.
Thus I will contain the same routines as L, but will carry the directory as the first record, thus speeding subsequent uses of SELECT.

<u>D mode</u> The file specified by L must have a directory as prepared by B, above. Then routines on L referenced by routines on I are added to file I. If an X argument is present, the routines in I plus those taken from L are written to the X file.
For overlays, the presence of an X argument is assumed, and file XEQ is used as default.

<u>N mode</u>: This mode does the same as D mode, except that the file specified by L does not contain a directory. This might be used if it was deemed uneconomical to first construct a directory.

Restrictions:

The file specified by L may not contain BLOCK DATA subprograms.

A main program must not be on the L file; if there is a main program, it must be on the I file.

In overlay mode, the order of the I file must be:
0,0 level first, primary and all its secondaries, primary and all its secondaries, etc.

SELECT uses an intermediary file called CLGO.

<u>Examples</u>:

(1) Compile a group of subroutines residing on UPDATE tape A1234, build a directory for them, and save them on tape A5678;

```
Job Card
REQUEST,OLDPL,5,Y,R,A1234.       Update format source code
UPDATE(F)                        Transfer routines (with or without
RETURN(OLDPL)                        corrections) to COMPILE file
RUN(S,,,COMPILE)                 Compile COMPILE, producing LGO.
SELECT(BM,I=LIB,L=LGO)           Place directory plus routines in
REWIND(LIB)                                                   LIB
REQUEST,XXX,5,Y,W,A5678.
COPYBF(LIB,XXX)                  Save LIB on tape
```

(2) Compile and execute a main program, taking routines from file LIB of the previous example:

```
Job Card
RUN(S)
REQUEST,LIB,5,Y,R,A5678.
SELECT.                          Equivalent to SELECT(D,I=LGO,L=LIB)
LGO.
```

(3) Compile and execute an overlay program taking subroutines from the alternate library file RUNIO, a catalogued file:

```
Job Card
RUN(S)                           Compile overlays
*RUNIO.                          Attach file RUNIO
SELECT(N,L=RUNIO,X=NEWFILE)      Complete LGO from RUNIO, (result to
NEWFILE.                         Load NEWFILE and execute.   NEWFILE.)
```

4.4 SYSTEM UTILITY PROGRAMS

4.4.1 Compiler Calls.

Compiler and assembler calls are special cases of the Program Call. In this section only the most common ones are described. For others, see the Chapter 5 on Languages, or the appropriate reference manual.

RUN (FL = 40K minimum)

The RUN card calls the FORTRAN compiler to process the next record of the source file (usually the job's INPUT file). Its general format is described in Chapter 5.

RUNN, RUNO

The RUN compiler is periodically modified. Whenever a new version is produced, it is first put into the system under the name of RUNN (RUN-NEW), and users are encouraged to try it. If there are no complaints, RUNN eventually becomes RUN, and the previous version of RUN becomes RUNO (RUN-OLD).

RUNN and RUNO take the same arguments as RUN.

FTN (FL 40K minimum)

The FTN card calls the FORTRAN EXTENDED Compiler to process the next record of the source file. The general format of the FTN card is described in Chapter 5.

COMPASS

The COMPASS control card calls the assembler which processes source code in CDC 6600 assembly language, and is described in Chapter 5.

4.4.2 FILE MANIPULATION UTILITY PROGRAMS

COPY (COPY TO DOUBLE FILE MARK) (FL = 5K)

COPY(file1, file2)

The binary file named file1 is copied onto file2 until a double file mark or an end-of-information mark is detected on the first file. Both files are then backspaced over the last file mark. If parameters are omitted, INPUT, OUTPUT are assumed.

COPYBF (COPY BINARY FILE) (FL = 5K)

COPYBF(file1, file2,n)

The number of binary files specified by n (decimal) are copied from file1 to file2. If parameters are omitted, INPUT,OUTPUT,1 are assumed.

COPYEOI (COPY TO END-OF-INFORMATION) (FL = 5K)

COPYEOI(file1,file2)

Copies all the information in file1 (including imbedded files) to file2.

COPYBR (COPY BINARY RECORD) (FL = 5K)

COPYBR (file1,file2,n,ℓ)

Copies n binary records of level ℓ from file1 to file2. If parameters are omitted, INPUT,OUTPUT,1,0 are assumed.

This operation terminates on reading a file mark from file1 or when the required number of records has been read. In the former case, a message to this effect appears in the DAYFILE. A file mark is written on file2 and positioned before the file mark.

COPYCF (COPY CODED (BCD) FILE) (FL= 5K)

COPYCF(file1, file2,n)

The number of coded (BCD) files specified by n (decimal) are copied from file1 to file2. If parameters are omitted, INPUT,OUTPUT,1 are assumed.

COPYCR (COPY CODED LOGICAL RECORD) (FL = 5K)

COPYCR(file1,file2,n,ℓ)

Copies n coded logical records of level ℓ from file1 to file2 parameters are omitted, INPUT,OUTPUT,1,0 are assumed.

This control card will not enable the user to copy single line or card images on tape or disk unless the logical record consists of just one line image.

If an END-OF-FILE is encountered before the required number of logical records have been copied, the END-OF-FILE is written on file2 but this file is left positioned before the END-OF-FILE so that the user can over-write it.

The following example illustrates how the user can manipulate card images on the disk file INPUT. The same manipulation could be done if the card images were on a Y-Tape (these logical records could have been put on tape prior to this run by using a COPYCF(INPUT,FILE2) control card.)

Example:

```
JOB CARD
COPYCR(INPUT,OUTPUT,1)
COPYCR(INPUT,NULL,1)
COPYCR(INPUT,OUTPUT,1)
7/8/9 in COLUMN 1-RECORD SEPARATOR CARD
CARD1
CARD2
7/8/9 IN COLUMN 1-RECORD SEPARATOR CARD
CARD3
CARD4
7/8/9 IN COLUMN 1-RECORD SEPARATOR CARD
CARD5
7/9/9 IN COLUMN 1-RECORD SEPARATOR CARD
CARD6
6/7/8/9 IN COLUMN 1-END OF JOB CARD
```

Produces on OUTPUT

```
CARD1
CARD2
CARD5
```

Note that the first COPYCR puts CARD1 and CARD2 on the file OUTPUT because these cards are contained in the logical record following the record of control cards. The next logical record(CARD3,CARD4) does not appear on OUTPUT because this record has been copied to the NULL file. CARD5, the next logical record, is copied by the third COPYCR card. CARD6, the last logical record of the job's INPUT file, is not copied at all.

COPYSBF (COPY SHIFTED BINARY FILE) (FL = 10K)

COPYSBF(file1,file2,n)

n binary files of coded information are copied from file1 to file2, shifting each line one character to the right and adding a leading blank. If parameters are omitted, INPUT,OUTPUT,1 are assumed.

Since the lead character of a print line is interpreted as a carriage control and is thus lost, COPYSBF can be used to print files which do not provide a leading control character.

COPYRF (COPY RANDOM FILE)

COPYRF(FILE1,FILE2,RB=nn)

This routine copies one random file from FILE1 onto FILE2, where either may be tape files. FILE2 will be a valid image of a 6603 outer zone random file. The parameter RB-nn should be either omitted, or be one of the following:

RB=56 if FILE1 is an image of an 841 file
RB-50 if FILE1 is an image of a 6603 inner zone file.
(This is the default value.)

COPYRF will produce a valid random file for direct use only when FILE2 is on a medium which has 64 PRU's per record block. However, COPYRF may be used to copy random files to and from tape, as long as one does not attempt to use the file on tape.

BKSP (BACKSPACE LOGICAL RECORD) (FL = 5K)

BKSP(file1,n)

This program allows backspacing of multiple logical records as specified by the decimal n. Backspacing will terminate if file1 becomes rewound.

REWIND (FL = 5K)

REWIND(file1,file2,...,filen)

This central program rewinds file1,...,filen.

SKIP (FL = 5K)

The SKIP control card is used to reposition multi-file files. This control card has the following two forms:

a) SKIP(file1,n)
b) SKIP(file1,nR)

In both forms the first argument, file1, is the file name for the tape to be repositioned. In form a) the second argument, n, is the decimal number of file marks to be skipped in the forward direction; in form b) the letter R appended to the second argument indicates that n file marks are to be skipped in the reverse direction. After skipping, the tape is left positioned on the far side of the last file mark skipped with respect to the direction of skipping and at the first piece of information (in the forward direction) within the file. An X tape will be positioned just before the file mark on the reverse skip.

CATALOG (FL = 36K)

CATALOG(file1,file2)

This routine provides a convenient means for identifying the contents of a file containing SCOPE binary decks. It includes information as to record and routine lengths in addition to checksum information which will uniquely identify particular versions of routines on that binary deck file.

The information is taken from file1 and listed on file2. If parameters are omitted, LIBRARY,OUTPUT are assumed. CATALOG output is formatted as follows:

RECORD	Number of the logical record with respect to its position on tape. Zero-length records produce a record number.
LENGTH	Length of the logical record as determined from the total number of central memory words contained in the binary cards.
PACKAGE	Routine name. A dash appears whenever a legitimate name does not appear in bits 59-18 of the first word of the logical record.
CKSUM	a 12-bit cheksum of the logical record or routine.

The following job will catalog a tape, number A1001, containing SCOPE binary decks.

```
(JOB)
REQUEST,BINDECK,5,Y,R,A1001.  USER NAME.
CATALOG(BINDECK)
6/7/8/9  IN COLUMN 1.  FILE SEPARATOR CARD
```

COMPARE (FL = 13K)

```
CØMPARE(file1,file2,n,level,m,r)
```

This routine compares one or more consecutive records on file1 with the same number of consecutive records on file2 to determine if they are identical.

The comparison begins at the point where each file is currently positioned and continues for n records of the level specified or a higher level. Discrepancies are listed in the OUTPUT file for the first m errors found in a record; this is done for r records.

If the records compared including level numbers are identical, the message GØØD CØMPARE is placed in the dayfile; otherwise the message is BAD CØMPARE.

Examples:

```
        CØMPARE(RED,BLUE)
```

Compare the next record on file RED with the next record of file BLUE.

```
        CØMPARE(GREEN,BLACK,3,2,5,20)
```

Compares the first 3 records of level at least 2; the first 5 discrepancies (on a word to word basis) will be printed for the first 20 records in error.

4.4.3 CATALOGUE FUNCTIONS

A catalog file is a file on mass storage catalogued by the system so that its location and identification are known to the system. Catalog files will not be destroyed accidentally during normal system operation (including deadstart) and they are protected by the system from unauthorized access according to controls specified when they are created.

A catalogued file is identified by two names and a cycle number. These are the Permanent File Name (PFN), the Logical File Name (LFN), and a digit specifying either one of two cycles (versions). The name by which a file is referenced in a program is known as the FNT name.

Three functions are available to the user under the catalogued file system as control card commands or as Fortran callable subroutines.

ADDPRO - catalogue an existing local mass-storage file.
ATTACH a previously catalogued file to a control point. (an attached control card uses an * in column 1.)
PURGE a file from the catalog file directory.

For details, refer to Computer Library Document No. AMD 656.

OBTAIN

OBTAIN is a program which retrieves files from either the system catalogue or from a user maintained library on magnetic tapes.

Instructions on how to set up a user library in OBTAINable format is contained in Library document No. 1204. Many system programs are available via OBTAIN. The format of this control card is:

```
OBTAIN(PFN,DIR,LFN,FNT,CYNO)
```

where *PFN,LFN,FNT,CYNO. would be the equivalent ATTACH,

and *DIR,DIR,DIR. would attach the directory.

OBTAIN first attempts to attach the catalogued file PFN,LFN,CYNO for use with FNT name FNT. If it fails, an attempt is made to get the directory, DIR, first as a COMMON file, then from the catalogue. If the directory is found, it is searched for the file. If a matching entry is found, OBTAIN requests the indicated magnetic tape reel, finds the file, copies it to disk, and catalogues it.

```
Example: To perform an update from YPL (The file YPL, YPL
         contains the source program of most system routines.)
         Job Card
         OBTAIN(YPL,YSTAPES,YPL,OLDPL)   (Get file YPL, YPL
         UPDATE(Q)                         with FNT name OLDPL)
```

4.4.4 MISCELLANEOUS CARDS

DMP (DUMP STORAGE TO OUTPUT FILE)

The function of this control card is to transmit information from core storage to the user's OUTPUT file.

There are several forms of the DMP card as described below. Each form identifies itself in the first line of the dump with a single word "label".

(a) "DMP." or DMP(0,0)

DMP without arguments reformats the Exchange package area and labels the P, RA, FL, EM, RE, FE and MA registers as well as the individuals A, B and X registers. It also prints the contents of those central memory locations whose addresses are found in the A and B registers. These locations contents are labeled C(register) and each of them is printed on the same line as the corresponding register.

A dump of words RA through RA+100 is produced followed by a dump of the area around the location indicated by the P register immediately follows an "Exchange-package printout. If P > 77, then C(RA+P-77) through C(RA+P+77) are printed; otherwise C(RA+0) through C(RA+100) are printed.

This dump is labeled "DMPX.".

(b) DMP (n1,n2) where n1 and n2 are octal numbers satisfying.

$0 \leq n1 < n2 \leq$ JOB FIELD LENGTH (FL) + 1.

This form of DMP dumps from n1 up to and including n2 and is labeled "DMP". If n2 is > FL, DMP stops at FL.

(c) The control point area of the requesting program is dumped if a card of the form

DMP(n,n) n ≠ 0

is encountered. The label reads "DMPC.".

(d) Absolute core dumps are produced by a

```
DMP(4xxxxx,4yyyyy)
```

where xxxxx defines the lower boundand yyyyy defines the upper bound plus one of the absolute core locations wanted.

For example:

```
DMP(400060,400200)
```

dumps absolute locations 60-177 inclusive containing a portion of the Operating System known as the "peripheral processor communication areas". The label on absolute core dumps reads "DMPA.". These dumps are used primarily by systems programmers.

All versions of DMP use an octal format, i.e. each word in Central Memory is represented by 20 octal digits.

All versions of DMP print only the first word of an identical block of words in central memory.

All versions of DMP print up to four central-memory words per line. If the initial address (first dump argument is not divisible by four, the first line is truncated so that the second line begins with a word whose address is divisible by four. If there is a block deletion as described in the above paragraph, then a line is truncated, if necessary, so that following lines begin with addresses divisible by four. Normally, an address is printed only for the first word of a line. If there has been a block deletion, however, the address of the first word following such a deletion is printed to indicate the deletion. If this word is not the first one in a line, the address is separated from the following printout of its contents by a → character.

For all versions of DMP the output buffer is emptied before any output is generated for a job. A busy output file will not result in the loss of a dump.

<u>DMPECS</u> (FL = 10K)

This control card has the following format

```
DMPECS(x,y,f,ℓfn)
```

The program dumps the users area of ECS from location x to location y and writes the output to file ℓfn. (ØUTPUT, if ℓfn is omitted).

f selects the print format

f = 0, or 1	4 words in octal and display code per line
f = 2	2 words in 15 bit octal parcels and in display code per line
f = 3	2 words in 12 bit octal bytes and in display code per line
f = 4	2 words in octal and is display code per line.

SETCORE, SETECS

These control cards preset a job's Central Memory or Extended Core Storage. There are two arguments which determine the pattern to be used:

SETCORE(a,b)   SETECS(a,b)

where a may be:

ZERO	for	0000	0000	0000	0000	0000_8
INF	for	3777	0000	0000	0000	0000_8
INDEF	for	1777	0000	0000	0000	0000_8
NGZERO	for	7777	7777	7777	7777	7777_8
NGINF	for	4000	0000	0000	0000	0000_8
NGINDEF	for	6000	0000	0000	0000	0000_8

and where b may be:

ZERO	no effect
ADDR	places 0400400000_8 plus the address of the word into the low-order half of each word.

If either argument is omitted, it is assumed to be ZERO. If both are omitted, they are assumed to be NGINDEF and ADDR. This default option has the effect to create an ERROR MODE 4 if the word is used in a floating-point arithmetic operation, or an ERROR MODE 1 if it is executed.

Note that SETCORE sets CM from RA+2 to RA+FL-1, but that the upper end of the job's CM may be overwritten by loader tables. Of course, SETCORE only presets memory for the very next program, and should appear immediately before the load cards.

BSEND

BSEND,N,FNAME.

This control card flags the file, FNAME, for routing to BROOKNET remote number N at job termination, or when the file is RETURNed.

Files created by BROOKNET remotes are catalogued, so a BSEND will also enter the file into the catalogue. FNAME will be used as the Logical File Name, and the Job Name will be used as Permanent File Name.

If this name-pair is not unique, an 8th letter will be appended to the Job Name.

Example: BSEND,6,OUTPUT.

If FNAME is not OUTPUT, the file must be forced to reside on disk; BSEND does this, provided that the file does not already exist.

A Fortran-callable version is available; the calling sequence is:

CALL BFSEND(N,FNAME,NRET)

where N is the remote number, FNAME is the file name in left-justified-zero-fill display code (set up by a nL.... expression), and NRET is an error return, set to zero if the BSEND succeeded and to an error code in case of trouble.

For more details including an expanded version analogous to a Catalogue ADDPRO, refer to the BROOKNET-2 USER's GUIDE, or Library document 1165.

BATACH

$$\text{BATACH,}\left\{\begin{matrix}\text{N}\\ \text{PFN}\end{matrix}\right\}\text{, LFN,CY,}\left\{\begin{matrix}\text{W}\\ \text{E}\\ \text{R}\end{matrix}\right\}.$$

This control card attaches to the job the file identified either by PFN (Permanent File Name), LFN (Logical File Name) and CY (Cycle No.) or by LFN, CY and association with remote N. The attach is done with the specified permission W (Write), E (Extend) or R (Read). If necessary, BROOKNET is instructed to return the file so as to make it available.

Write permission allows the user to write anywhere on the file; Extend permission allows writing only after the end-of-information, Read specifies read-only.

A Fortran callable version exists; the calling sequence is:

CALL BFATACH($\left\{\begin{matrix}\text{N}\\\text{PFN}\end{matrix}\right\}$, LFN,CY.1L $\left\{\begin{matrix}\text{W}\\\text{E}\\\text{R}\end{matrix}\right\}$, NRET)

For details, see the BROOKNET User's Guide, or Library document 1165.

<u>REMOTE</u>

REMOTE,FNAME

This control card marks the file, FNAME, for routing to the CDC 3200 FOCUS system at job termination. It is automatically inserted into the control card record when a remote file is specified in a FOCUS SEND command.

<u>REMOTB</u>

REMOTB,FNAME.

This control card designates as a binary file, a file already marked for routing to the CDC 3200 FOCUS system by a REMOTE card. REMOTB is not generated by FOCUS but must be inserted into the control card record whenever a binary file is to be returned to FOCUS.

5.0 LANGUAGES

5.1 FORTRAN

FORTRAN is the most commonly used of the scientific programming languages. Two compilers are available for use on the 6600.

5.1.1 RUN

The standard Fortran (RUN Compiler) is described in the Control Data Fortran Reference Manual (CDC Publication Number 60174900). The RUN Fortran Compiler is called by the RUN Control Card whose general format is

```
RUN(CM,,BL,IF,OF,RF,,AS,CS)          or          RUN.
```

CM Compiler Mode.

- S Compile and list source statements.
- P Compile, list source statements and punch object deck. (The binary object deck is written to both the PUNCHB and the LGO file and can be loaded from either. Compass decks assembled under RUN(P), however, write their binary decks only to PUNCHB).
- L Compile and list source statements together with the object code generated for them.
- M Compile and list source statements together with the object code generated for them. Also punch binary object deck.

BL The length (Octal) of each I/O buffer in the object program. Two commas precede the "BL" field.

IF Name of the input file that contains the source program to be compiled.

OF Name of the output file for print output from the compiler.

RF Name of the file on which the compiler will write the binary load-and-go translation of the compiled programs. See comments in P above.

as ANSI switch. This switch may be used to select the ANSI type of interaction between a list of data items in an input or output statement and a FORMAT statement. If it is non-zero, the interaction at execution time will take place according to ANSI specifications. Essentially, this concerns any list items that remain to be processed after the outermost right parenthesis of a FORMAT statement has been encountered. A description of the ANSI specified interaction is given in the CDC FORTRAN manual in Section 9.8. If this switch is omitted, the interaction will be the same as for standard 6600 FORTRAN. Two commas precede the "as" field.

CS Cross-reference switch. If this switch is non-blank, the compiler will produce a cross-reference listing. If omitted, no cross-reference listing will be produced.

The extra commas prior to the "BL" and "AS" fields occur for historical reasons. The RUN card formerly contained two additional fields.

The defaults for the above parameters are

CM = S

BL = 1011 (OCTAL)

IF = INPUT

OF = OUTPUT

RF = LGO

The RUN compiler can automatically call the COMPASS assembler (See Section 5.2 below), and consequently may be used to process mixed FORTRAN and COMPASS decks, provided that the first card of the COMPASS deck contains IDENT in columns 11-15, and that the END card also starts in column 11.

5.1.2 FTN

The FORTRAN EXTENDED (FTN) compiler, which provides the ability to optimize either compilation or compiled code is described in the Control Data Fortran Extended Reference Manual (CDC Publication Number 60176600). The FTN compiler is called by the control card whose general format is

```
FTN(P1,P2.....PN)        or        FTN.
```

The P_I are parameters specifying files or options, as follows:

```
I=lfn specifies the source file (default I=INPUT)
L=lfn specifies the file to which the listing is
      written (default L=OUTPUT)
B=lfn specifies the file to which the relocatable binary
      output is written (default B=LGO)
OPT=m specifies the level of optimization desired;
      m=0 is fast compile mode
      m=1 is standard compile mode
      m=2 is fast object code mode
```

5.2 COMPASS

COMPASS is the assembly language for the 6600. It contains all machine instructions for both CPU and PPU programs, as well as most other features that one would expect an assembler to have. The format of the compass control card is

```
COMPASS(P1,P2,...,PN)        or        COMPASS.
```

The meaning of the parameters, as well as a complete description of compass may be found in control data 6600 Compass Version 2 Reference Manual (CDC Publication Number 60279900).

5.3 COBOL

COBOL (Common Business Oriented Language) is designed to simplify the programming of business data processing problems. With the Cobol language, the user produces easily modifiable source programs using English language type statements, resulting in shorter program development and low program conversion costs. An example of a control card set for using COBOL is

```
COBOL(P1,P2,....PN)     or     COBOL.
OBTAIN(YS,YSTAPES,COLIB,LIB)
SELECT(NM)
LGO.
```

Full details on the meanings of the parameters on the COBOL control card, as well as a full description of COBOL may be found in Control Data COBOL Reference Manual 6000 version 3 (CDC Publication Number 60253000).

5.4 SORT/MERGE

SORT/MERGE provides the capability to sort and/or merge up to 32 files. The control card for calling SORT/MERGE is

```
SORTMRG.
```

A full description of SORT/MERGE may be found in the Control Data SORT/MERGE Reference Manual 6000 Version 3 (CDC Publication Number 60252600.

5.5 SNOBOL4

SNOBOL4 is a language used for character string manipulation. It may be obtained with the control card

```
SNOBOL.
```

Details on the use of SNOBOL4 may be found in the CSCF Library (Writeup Number R 02 2213 and in "The SNOBOL4 programming language -- second edition", by R. E. Griswold et al, Prentice Hall Inc., 1970).

5.6 LISP

LISP , a list processing language, is a formal mathematical language designed primarily for manipulation of symbolic rather than numeric data. The standard User's Manual for acquiring a knowledge of the use of LISP is the "LISP 1.5 Programmer's Manual" published by the M.I.T. Press.

LISP may be called with the control cards

```
*LISP.
LISP.      or      LISP(P1,P2)
```

For a description of the parameters, and details of the version of LISP available here, please see "The University of Texas 6400/6500 LISP 1.5: An adaptation of MIT LISP 1.5", which is available in the CSCF Library.

5.7 SIMULA

SIMULA (SIMUlation LAnguage) is a dynamic language which describes a sequence of actions rather than a permanent relationship. It is particularly well adapted to simulation applications and contains the ALGOL-60 language as a proper subset.

SIMULA may be called with the control cards

```
*SIMULA.
SIMULA(P1,P2,...PN)
```

For a description of the parameters and of the language, please see the Control Data SIMULA Reference Manual (CDC Publication Number 60234800 Revision E), which is available in the CSCF Library. For a description of local variances from that manual, please see CSCF Writeup Number W007.

5.8 ASSEMBLERS FOR OTHER MACHINES

5.8.1 PAL

The PAL Assembler for the PDP-8 computer may be obtained with the control card.

```
PAL.
```

A description of the assembler can be found in the CSCF Library (Writeup Number 1220).

5.8.2 PDP11

A 6600/PDP-11 binary to binary translator and the macros for compass assembly of PDP-11 programs may be obtained with the file names PDP11 and SPD11 respectively. A description of the assembler can be found in the CSCF Library (Writeup Number 1007).

5.8.3 SYMBOL

Symbol is an assembler for the XDS-920, which runs on the6600. The control cards to call and execute symbol are

```
*SYMBOL.
SYMBOL.
```

A full writeup is available in the CSCF Library (Writeup Number 1232).

6.0 SOFTWARE FACILITIES AND PACKAGES

6.1 Libraries

Three subroutine libraries are maintained for general use. These are: (1) SBNL consisting of routines written locally; (2) VIM, consisting of routines received through the 6000 user's group, and (3) SSP which is the SHARE subroutine package.

To print a list of the subroutines with deck number, keyword and a short description of each routine on the Program Library as found in the CSCF Library, the following control cards may be used:

```
*SBNLLST.
COPYBF (SBNLLST, OUTPUT)
*VIMLST.
COPYBF (VIMLST, OUTPUT)
*SSPLST.
COPYBF (SSPLST, OUTPUT)
```

All update deck names are a four digit number corresponding to the write-up number or the last four digits of the classification code published periodically in the newsletter and available from the CSCF librarian.

DECK name ranges:		
	SSP	0001 thru 0999
	SBNL	1001 thru 1999
	VIM	2001 thru 2999

1. To obtain just a listing of SBNL routine 1085:

```
JOB CARD
OBTAIN (YPL, YSTAPES, SBNL, OLDPL)
UPDATE (Q)
RUN (S,,, COMPILE)
7/8/9
*COMPILE 1085
6/7/8/9
```

2. To obtain a source deck and listing of 1085:

```
JOB CARD
OBTAIN (YPL, YSTAPES, SBNL, OLDPL)
UPDATE (Q,C = PUNCH)
RUN (S,,, PUNCH)
7/8/9
*COMPILE 1085
6/7/8/9
```

3. To obtain a listing, source deck, object deck and include the routine in a compile and execute with your Fortran program.

```
JOB CARD
OBTAIN (YPL, YSTAPES, SBNL, OLDPL)
UPDATE (Q, C = PUNCH)
RUN (P,,, PUNCH)
RUN(S)
LOAD (PUNCHB)
LGO.
7/8/9
*COMPILE 1085
7/8/9
    (Fortran Program)
6/7/8/9
```

For VIM and SSP routines replace OBTAIN cards in above examples with:

```
      OBTAIN (YPL, YSTAPES, VIM, OLDPL)
or    OBTAIN (YPL, YSTAPES, SSP, OLDPL)
```

and insert appropriate "*COMPILE" card.

6.2 Utilities

Various utility routines/packages are provided. Since there are too many to list here (and since new ones are often born), only a small selection is included in this chapter (along with the CSCF Library writeup number of each) to illustrate the types of utilities available.

6.2.1 835 Plotting System

The 835 plotting system consists of a Calcomp Model 835 Electronic Plotter, a Calcomp Model 565 Electro-Mechanical Plotter, and a Calcomp Model 780 Magnetic Tape Unit. Various subroutines are provided for using this equipment (Writeup #AMD 577).

6.2.2 MATECS

MATECS can perform real algebraic operations on large matrices stored in ECS. It can multiply matrices of order 200. (CSCF Writeup #1175).

6.2.3 MATRIX

MATRIX enables the user to perform standard algebraic matrix opertions (e.g. Transpose, Symmetric Product, Ergenvalue/Ergenvector, etc.) (CSCF Writeup #1058).

6.2.4 TYPIST

TYPIST provides an easy method of producing and maintaining documents. This is accomplished by preparing on a machine readable medium (i.e. punched cards, disc file) free form text with interspersed control commands and parameters from which TYPIST can assemble the pages of a finished document (CSCF Writeup #1181).

6.2.5 ANATAPE

ANATAPE analyzes magnetic tapes on the 6600, including optional tape dumps (CSCF Writeup #1201).

6.2.6 CARDS, KARDS

These routines punch a coded file, one 80-column card per record (avoiding the extra cards punched by standard copy routines when record size exceeds 80 characters). (CSCF Writeup #1137)

6.2.7 DEBLOCK

DEBLOCK provides the facility to read and transfer non-SCOPE magnetic tapes including belocked BCD Tapes, i.e. fixed size logical records without terminators ($\leq$ 136 characters), and packed BCD tapes, i.e., logical records ending with a specified terminator ($\leq$ 150 characters). (CSCF Writeup #1203).

6.2.8 CARDCON

CARDCON converts card images from one format to another via an arbitrary user-supplied dictionary. (CSCF Writeup #1092)

6.2.9 DEFOCUS

DEFOCUS provides a means of accessing FOCUS files from the 6600 directly, allowing a user to continue his work when the 3200 computer and/or the FOCUS system are unavailable. (CSCF Writeup #1180)

6.2.10 UPDATE

UPDATE is a system program used to create, manipulate and maintain library files. See the CDC SCOPE Reference Manual (CDC publication #60189400).

6.3 Catalog Files

A catalog file is a file on mass storage catalogued by the system so that its location and identification are known to the system. Catalog files will not be destroyed during normal system operation (including deadstart) and they are protected by the system from unauthorized access according to controls specified when they are created.

A "Users Guide to Catalog Files" is available from the CSCF Library (Writeup Number AMD 65[illegible]). Please refer to this writeup for full details on the use of catalog files.

6.4 FOCUS

FOCUS is a multiple access file-handling system operating on a Control Data 3200 which is connected to a Control Data 6600. The system provides remote terminal users the facilities to: Create, store, modify and reference source and binary files; Create programs in the form of executable files, and submit them for execution on the 6600; print the contents of files at the remote terminal.

Details on the use of FOCUS may be found in the "Focus User's Guide, Revision February, 1973" available from the CSCF library.

6.5 BROOKNET

BROOKNET is a digital communications network which provides a data link between the CSCF and various user supplied computers at the laboratory. A remote computer can use BROOKNET for one or both of the following:

1) Remote Input/Output of ordinary jobs to be run on one of the 6600's.
2) Fast, direct data transmission between an experimental area and a file at the CSCF.

A full description of BROOKNET and how to use it is "BROOKNET-2 A Guide to Brookhaven Online Computer Network". (AMD 631).

User facilities BSEND and BATACH are detailed in a separate writeup (#1165). Both of these are available from the CSCF library.

7.0 CSCF ACCOUNTING AND SCHEDULING

7.1 Multiple Resource Accounting System

The Multiple Resource Accounting (MRA) system attempts to distribute the CSCF operating cost over the users as equitably as possible while minimizing variations in charges based upon the varying workload characteristics of a multi-programmed machine. Computer usage is measured in Computer Charge Units (CCU), and eight resources are taken into account as indicated in the following table:

Resource	CCU's
512 words of CM in use for 1 hour	1
4096 words of ECS in use for 1 hour	1
32768 words of disk storage in use for 1 hour	1
Tape drive in use for 12 minutes	1
One second of CP time (36 CCU/hour)	.01
One second of PP time (7.2 CCU/hour)	.002
One mounting of an N-series tape	.75
One mounting of any other tape	.5

There is a minimum charge of 0.15 CCU's per job.

The 6600 computers obviously provide more than eight resources, but only those for which a practical means of measurement is currently available were incorporated into the MRA. The number may be expanded in the future.

The first four items in the above list are time integrals which are approximated by a sampling technique. Approximately every 10 microhours (36 ms.) control point activity is examined and a decision is made whether or not to charge for the previous interval. If the job is either using the CP or has a non-system PP working for it, the job is charged for the previous interval at the level of resource usage that exists at that point in time.

The amount of CM, ECS and disk storage utilized is truncated to the nearest multiple of 512, 4096 or 32768 words respectively, to establish the level of usage. The amount of disk storage and the number of tape drives in use are updated at approximately two second intervals; CM and ECS usage is updated when a change occurs.

Central processor and peripheral processor usage is computed from the times shown in the dayfile at the time the job terminates.

Tape mount usage is recorded every time an operator must mount a tape and hence includes each continuation tape for multiple reel files as well as each tape separately requested.

7.2 Financially Weighted Priority System

Four financially weighted priority categories are now in effect. They are called RUSH, RAPID, NORMAL and STANDBY. Each priority category has been assigned a weighting factor which describes the rate of charge for processing in that category relative to the rate of charge for processing in the NORMAL category. The initial values of these factors, which were recommended by the Departmental Computer Coordinators, are as follows:

RUSH	4.00
RAPID	2.00
NORMAL	1.00
STANDBY	.40

All jobs in the RUSH category will be performed immediately and may be preceded only by EMERGENCY or other RUSH jobs.

Jobs in the RAPID category will be performed after emergency and RUSH jobs have been completed and may be preceded only by EMERGENCY, RUSH, other RAPID jobs.

NORMAL category jobs will be performed after EMERGENCY, RUSH, and RAPID jobs have been completed, and will be preceded only by jobs in those categories, other NORMAL jobs, or pre-scheduled production.

The STANDBY category represents the lowest level of priority. Jobs in this category will be processed only when all other processing has been completed. Additional preventative maintenance and engineering changes will take precedence when necessary.

Job card priority numbers 1, 7, 10, 16, have been assigned to the STANDBY, NORMAL, RAPID, and RUSH categories, respectively. Only jobs with these job card priority numbers will be treated as described above. All other job card priority numbers which are undefined will be treated as described below. However, all job card priority numbers, including undefined numbers, will be assigned to one of the four categories, RUSH, RAPID, NORMAL and STANDBY, for charging purposes. The treatment which will be accorded jobs which have been given undefined job card numbers cannot be predicted.

7.3 Notes for Better Scheduling

1) Improper job type (A, B, C, or D) specification on the Job card can result in more expensive processing. If two CP-bound jobs are active at the same time, they interfere with each other, result in longer execution time for both, and thus increase their real-time residence in CM.

2) Underestimating line limit on job card. Line limit is not used for scheduling a job, it is entirely for your protection to keep a job that is caught in a loop from producing enormous amounts of output.

The amount of output actually produced, or rather an estimate of this amount, is used for output scheduling, not the line limit on the job card.

3) Users actually submit RUSH jobs, then do not pick up their output for 24 hours, or else ask for the output to be returned to them via the slow delivery service. Less dramatic is the user who submits a priority 10 (RAPID) job, pays double the normal rate, and then doesn't pick up his output for many hours. The OX teletype terminal in the ready room enables you to gauge fairly well when your job will be run.

4) Too many users submit the job, which they just finished coding, with a whole drawer of data cards. Obviously, the job will not run the first few times until all Fortran errors are corrected. Why submit all data cards? Handling of these big decks is non-trivial. For that matter, why handle most of the enormous decks that are being submitted. Often, data or even source code could be stored on tape, data which you expect to use frequently during the day could be put in a *COMMON* or catalogued file, or why not use the FOCUS system?

To summarize, we recommend that users

- exercise care in indicating job parameters on the job card;
- reduce your program's field length whenever possible by judicious use of ECS;
- eliminate excessive card handling by not submitting voluminous data decks when the probability of getting past the compiler is practically nil;
- use the OX teletype for helping you to decide on job priorities;
- use the FOCUS system for your file handling and job submission;
- ask for assistance whenever needed;

7.4 Print Queue Scheduling

Printing of output files will be scheduled according to an algorithm that takes into account a job's priority, time in the input queue, and length of the output file. The length considered is the approximate number of lines generated, not the maximum punched in the Job card. The system does not actually count lines of output, but estimates the line count from the volume of information in the output file. Thus, the limits given below are only approximate.

Highest priority in the print queue goes to RUSH jobs, whose output will always get printed on the first available printer. Output files of under 1000_8 lines come next, among them priority 10 jobs taking precedence over the priority 7 ones. Following these short output files come the ones having less than 20000_8 lines, again with the priority 10 over 7 precedence, and within each of these groups ordering will be essentially by time spent in the input queue.

Print jobs of over 20000_8 lines, or approximately 9 minutes of printing, will be held up as described below. While this might cause some hardship to a user, it will prevent the currently occasional tie-up of printers causing RUSH jobs to have their printing delayed an inordinate amount of time.

Printing of these held up jobs on a one-printer system will only take place after operator intervention. On a multi-printer system which has no jobs under 20000_8 lines waiting in the print queue, one of the held up jobs will be allowed to print at one time, thus tying up one of the printers for extended time periods, but only until a backlog of regular print jobs develops, at which time regular scheduling again becomes effective. Of course, the operator can always supersede automatic scheduling. In exceptional circumstances, especially at night, the operations staff will try to cooperate with users having special requirements.

Held up jobs are also priority ordered and priority 10 jobs in the holding queue will be printed before the priority 7 ones, whether under the manual method on a one-printer system or the automatic one on a multi-printer machine.

7.5 Job Refunds

On occasion, computer time is charged for the running of jobs that have failed due to circumstances beyond the user's control such as machine errors, system errors or human error. In these cases, the computer center personnel will provide a refund in the form of a restart on rerun of the job in question providing the problem is brought to our attention as soon as possible.

Users should apply for this type of refund through the Computer Information and Assistance Office or, after hours, the Shift Supervisor. In either case, the user should be prepared to surrender the output in question as this output is very useful in the investigation of the problem.

Such refunds will not normally be granted for jobs aborting because of tape parity or other conditions which should be handled under program control.

Normally, no more than 60 ccu is refunded per job; users whose jobs are lengthy are expected to include restart procedures.

8.0 CDC 6600 HARDWARE AND OPERATING SYSTEM CHARACTERISTICS*

8.1 HARDWARE

The CDC 6600 computer consists of a central processor (or CP) and ten peripheral processors (or PPs) which share a common central memory. Each peripheral processor possesses its own additional "private" memory, inaccessible to the other peripheral processors or to the central processor.

The central memory consists of 65,536 words (or locations) each containing 60 binary digits. It is a core memory with a read/write cycle time of one microsecond, and is divided into sixteen banks with independent read/write cycles, so that the eleven processors could, if ideally timed, transfer information to and from central memory at the rate of sixteen 60-bit words per microsecond. In actual practice, because 60-bit words are transferred at a rate of one per minor cycle (100 nanoseconds), only ten words can be transferred per microsecond. Each peripheral processor has a memory unit consisting of 4,096 words of 12 bits each.

Each peripheral processor may read or write any location of central memory. However, a program running in the central processor is restricted to a contiguous block of storage beginning at a given point known as the Reference Address (RA), and extending for a given number of locations known as the Field Length (FL). All addressing of central memory by the central processor is relative to the reference address; thus a central processor program may be loaded into any part of central memory and will run properly, provided that the RA and FL are properly set. The RA and FL may only be modified by a PP program, specifically the system monitor.

The central processor has eight 60-bit arithmetic registers (X0...X7) where most calculation is done, eight 18-bit address registers (A0...A7) that control data transfer between memory and the arithmetic registers, and eight 18-bit increment registers (B0...B7), used for address calculation, loop counting, and other purposes. (B0 is special; it always contains zero). In addition, there are the program address register (P) which contains the address of the next instruction to be executed, the reference address register (RA), the field length register (FL), and the exit mode register (EM). The exit mode register allows the trapping of arithmetic overflow and bounds protection errors.

Central processor instructions are packed two, three or four to a 60-bit word. Instructions are fetched from a stack of instruction registers eight words long, which is replenished automatically. Consequently a

*This chapter has been adapted from a corresponding Smithsonian Astrophysical Observatory Guide.

program loop which fits into the stack will not require any instruction fetches from central memory beyond the initial ones.

The central processor does not have any input or output instructions and cannot directly communicate with any I/O device, or even with the computer operator.

The peripheral processors share twelve data channels, to which are connected all I/O devices, including the operator's console. Thus, the responsibility for all input and output, including operator communication, rests with the peripheral processors.

In addition, a peripheral processor has the ability to perform a special operation called "exchange jump." This consists of exchanging the contents of the central processor registers with the contents of a specified area of central memory. Prior to doing this, the peripheral processor must, of course, see to it that the specified area contains the proper information to start or continue some program stored in central memory. By the exchange jump, the peripheral processor interrupts the program presently using the central processor, saves all information necessary to continue that program exactly where it left off, and sets some other program (stored elsewhere in central memory) into operation. A peripheral processor can thus switch the central processor from one task to another in "midstream" without losing its place in any of the tasks. This makes it possible to implement a multiprogramming operation system. The central processor is interrupted for only two microseconds while the exchange jump is performed. For details as to how the operating system uses this feature, see 8.2.

User programs run only in the CP, never in a PP.

The memory of each of Brookhaven's CDC 6600's is supplemented by Extended Core Storage (ECS). This device consists of one million 60-bit words of storage which may be accessed through four ports. It is currently attached to both CDC 6600's and to the Brooknet hardware, leaving one port available for future expansion. ECS can be run in two modes: Split mode whereby each computer has access to half of the available ECS, and shared, each computer having access to any portion of ECS.

Instructions cannot be executed out of ECS. Instead it is used as a temporary storage and swapping device. This makes use of the extremely high data rates between ECS and the CM of either 6600. The transfer rate is ten 60-bit words per microsecond, after an initial delay of 3.2 microseconds.

8.2 SCOPE 3 OPERATING SYSTEM

8.2.1 Multiprogramming

Clearly a great deal of parallel operation is possible in the CDC 6600 computer. The ten peripheral processors, the central processor, and all input and output devices could be in operation simultaneously, provided they had tasks to perform. To assign tasks to the various components of the computer with a high degree of overlap, the SCOPE 3 Operating System employs the technique of multiprogramming.

Briefly, this means the computer works on more than one user program at a time. Only one user can occupy the central processor, but when the program using the CP requests a task of a peripheral processor, the CP can be turned over to a second user program while the first waits for the PP operation to be completed. Similarly, only one of the peripheral processors can use a given data channel at a time, but they can take turns driving the different devices on that channel.

8.2.2 Files

The Scope 3 Operating System handles all information in the form of files. A file is a collection of data, programs, or other information in a form which may be read by, or is produced by, the computer. Each file has a name assigned by either the user or the system. This name consists of 1-7 alphanumeric characters, with the first character alphabetic. The operating system recognizes two arrangements of file -- sequential and random -access.

A sequential file is one in which information is read or written in sequential order, as on a magnetic tape. Such a file has a beginning and an end, and is positioned at a given point at any time, ready to read or write the next record. When a sequential file is written on, all information following the record just written is lost.

A random-access file is one in which any record may be read or rewritten at any time, without explicit positioning. Also, new records may be added at any time. The records in such a file may be indexed by number (in order of creation) or by names assigned to each record by the user or by relative physical position.

A sequential file may reside on magnetic tape, disk (or disk pack) storage, ECS or in some special cases be assigned to a card reader, card punch, line printer, or other device. The great majority of users will be concerned only with tape and disk files. Random-access files may reside only on disk storage or ECS.

A file consists of a sequence of logical records terminated by an End-of-Information mark (EOI). Logical records can have level numbers, 00 to 15 (octal); a logical record of level 17 is usually called a file (within a file). Records of level 17 terminate with an End-of-File (EOF), others terminate with an End-of-Record (EOR).

A file (whether sequential or random access) may be either binary (also called odd-parity) or coded (also called BCD or even-parity). Binary files and coded random-access files are usually read or written one or more logical records at a time. However, the logical records of coded sequential files are normally subdivided into unit records, and such a file may be read or written a unit record at a time. A unit record may be thought of as representing a punched card or a printer line image, and is usually restricted to 140 characters or less. Logical records are restricted only by the capacities of the tape or disk, and in some cases by central memory buffer sizes allotted.

In addition to the above classifications, a file will be of one of the following <u>types</u>: input, output, local, or common. Local and output files may have disposition codes as well.

When the user's job deck is read by a card reader, it is given a file name (the so-called Job Name), made into a file of type <u>input</u>, and stored in disk storage. There may be many such user input files waiting in disk storage for execution. The files make up the <u>input queue</u>.

When the job gains access to the central processor, the control cards, which constitute the first logical record, are copied into a special buffer, and the rest of the input file is assigned <u>local</u> type and given the name INPUT. While the job runs, it may create other local files. If it writes on the standard output file, a local file called OUTPUT is created. There are also standard Hollerith and binary card punch files (PUNCH and PUNCHB respectively). Tape files and scratch files on disk or in ECS are also local.

A user may request that a disk based local file be made <u>common</u> or may assign a disposition code to the file. When his job terminates, the following events happen. All common files are saved on disk storage and may be subsequently called by other users. Files that have disposition codes (including OUTPUT, PUNCH, and PUNCHB, which are assigned disposition codes automatically) are given type <u>output</u> and eventually processed according to the disposition code and then destroyed. All other files are destroyed immediately. Tapes cannot be commoned or disposed; they are unloaded at the end of the job and the tape position is lost.

The files of type <u>output</u> make up the <u>output queue</u>. These files will be printed on the line printer, or punched in Hollerith or binary on the card punch (depending on the disposition code) as the devices become available. Access order is determined from a priority generated by the system, depending on the length of the file and other factors.

In addition, there are several special files defined by the system that are handled differently from the above description. The user need not be concerned about the structure of these files since their use is provided automatically or through control cards. These special files include:

1) The System Library

   This file contains the major compilers, assemblers, utility routines, and all programs and overlays of the operating system.

2) The User Dayfiles

   Each program, while occupying central memory, has a dayfile into which messages are written during the program run. These may be error messages, accounting information (amount of computer time used, time of day, date, etc.), or messages generated by the user himself. This dayfile becomes the last part of his printed output when the job terminates.

3) The System Dayfile

   This contains all messages put into user dayfiles, plus some additional record-keeping messages put in by the system. This file is never seen by users, but is used for Computing Center record-keeping and accounting.

8.2.3 Basic Structure of Scope 3

Under Scope 3, only two of the peripheral processors (PPO and PP9) always contain fixed programs. PPO contains a monitor program called MTR, whose responsibility is assigning tasks to the central processor and to PP1 through PP8, as well as keeping track of all system times, dates, and use of central memory. PP9 contains a program called DSD, which drives the console display scopes, interprets operator type-ins, and handles all processing of the user and system dayfiles.

The operating system can keep track of up to seven jobs requiring central memory at once. Each such job is said to be assigned to a control point. Associated with each control point is a contiguous region of memory called the field length which the job can use as it wishes. There is also another central memory area called the control point area associated with each control point. This area contains information about the job presently at the control point for the use of the operating system, and is not accessible to the user.

The user indicates the maximum central memory field length his job may require on his JOB Card. His job will be brought to a control point only if that much memory or more is available and his turn is next. If the field lengths of other control points must be moved around to make a contiguous area of the proper length, this will be done. The job is then brought to the available control point, and the requested field length is assigned. The user may request that his field length be increased or decreased at any time. The request will be honored by MTR as soon as possible. If the request is for an increase, the waiting time could be very large, of course.

In addition to central memory assignments, one or more files may be assigned to a control point, as well as devices (such as tape units). Each local file is assigned to some control point. Common files are assigned to a control point when they are in use. Files of type input and output are not associated with control points.

Control points may be assigned to certain system tasks as well as to user tasks. For example, the system programs which create input files and process output files are assigned to control points. BROOKNET also runs at a control point.

8.2.4 Priority

The selection of jobs from the input queue to be brought to control points is governed by a priority assigned by the system. When the job enters the machine via card reader or remote station, the system assigns a priority based on the Job Card priority, CP time limit, and field length. Jobs requesting larger amounts of time or memory receive lower priorities. Furthermore, the priority is periodically incremented for all waiting jobs, so that the priority increases with waiting time. As a general rule, whenever a control point and some central memory are available, the input queue is searched for a job that will fit in the available space and whose priority is at the highest level of the jobs in the queue. If such a job is found, it will be brought to the available control point and executed. If the queue includes a Class 17, no other job will be brought up ahead of that task.

Similarly, the selection of files in the output queue for printing or punching is governed by a priority based mainly on waiting time and the size of the file. (Shorter files get higher priorities.)

Jobs at control points obtain use of the central processor by a priority system as well. The control point priority algorithm is designed to maximize the effective use of the CP.

9.0 PROGRAMMING PRACTICES AND TIPS

Good programming practices have certain goals. Among these are efficiency in terms of both computer resources and human resources. The former aims to reduce the amount of central memory and processor time so as to reduce the dollar cost of a job. The latter is a more nebulous concept; it includes program clarity and intelligibility and those "tricks of the trade" which allow one to accomplish more with less effort. Sections on tips for FOCUS and BROOKNET users and on debugging and the avoidance of pitfalls are included in this chapter.

9.1 Computer Efficiency

9.1.1. CM reduction

A sizable part of the charge for computer usage is (CM in use) x (real time), generally known as Kiloword-hours. Consequently it is in the user's interest to reduce the amount of CM used. Furthermore, reduction of a job's CM makes more CM available for other jobs, and thus enhances the multiprogramming capabilities of the machine. Below we list some of the ways in which CM requirements can be reduced.

a) Overlays

If a portion of a program is not required for the full duration of the job, the area it occupied may be given to another portion of the job, and then reloaded when needed. A description of segmentation and overlay features may be found in Chapter 8 of Fortran Reference Manual, CDC Publication No. 60174900.

b) COMMON

The loader requires a certain amount of table space, and therefore uses more CM than the job it is loading. However, if an array of size equal or greater to the loader tables (usually about 2K) is assigned to blank COMMON (the area used for loader tables) no extra space will be needed.

c) Extended Core Storage

ECS may be used as secondary storage in two ways. A block of it may be assigned to a job and accessed directly via ECREAD and ECWRITE subroutines, or a scratch file may be assigned to ECS.

Direct access to ECS is the more efficient method. Up to 400000_8 words may be assigned to the job by simply punching the number of octal thousands of words required in columns 31-33 of the job card. It is often advisable to use ECS for large blocks of data which will be used repeatedly, and in a systematic manner (such as row-by-row access of a matrix).

ECS has an access time of 3.2 microseconds and a transfer rate of one word every 100 nanoseconds.* This means that it takes about 103.2μ to transfer 1000 words to or from Central Memory (CM) - disk and tape I/O take hundreds of times longer, and must wait for the monitor to respond to a request. Accessing single words from unrelated locations is not as economical, as it costs 3.2μ per word.

The subroutines to read and write ECS are called as follows:

```
CALL READEC (A,L,N)
CALL WRITEC(A,L,N)
```

where:

A is a variable in CM;
L is the address in ECS. Under the RUN compiler ECS addresses are numbers from zero to the amount of ECS assigned minus one. Under FTN, L must be a variable defined to be in ECS by a type ECS statement and assigned to an ECS COMMON block. For example

```
RUN                          FTN

REAL A(100)                  REAL A(100)
.                            ECS L(1000)
.                            CØMMON/ECSBL/L
.                            .
.                            .
.                            .
CALL WRITEC(A,500,100)       CALL WRITEC(A,L(501),100)
```

N is the number of words to transmit

If it is desired to move large arrays to ECS, but without extensive recoding, the following trick may be used. Since it greatly increases the overhead (by about 10μsec per word), it should be used only for parts of the program which are not executed very often, but where the ECS variable appears many times in the source code.

If A(i,j,k) is the array to be moved to ECS, removing its dimension statement will make every reference to it on the right side of an arithmetic replacement. (i.e. after the equal sign) into an external function call. The subprogram below, having the same name, will examine the subscripts used, read ECS if necessary, and return the proper value. Uses of A in other contexts must all be replaced, but most will produce Fortran errors when the dimension statement is removed. (RUN reference option lists all uses of A.) When an element of the ECS array is altered and stored in ECS, the variable KNOW (which should be in some labeled common) must be set to zero.

*If the other 6600 happens to be accessing ECS at exactly the same time, these rates degrade by 50-75%. If an exchange jump interrupts an ECS transfer, it will be restarted from the beginning. However, these conflicts occur very, very infrequently.

```
      FUNCTION A(I,J,K)
      DIMENSION A(imax,jmax,kmax),AK(imax,jmax)
      COMMON/ECS/KNOW - - - this is not an ECS COMMON block
      DATA KNOW /0/
      DATA LECS/0/                      start address of A in ECS
      DATA IMAX,JMAX,KMAX/imax,jmax,kmax/
      LUMP=IMAX*JMAX
      KK = (K-1)*LUMP
C IF PLANE K NOT IN AK READ IT FROM ECS
      IF (K.NE.KNOW)  CALL READEC(AK,LECS+KK,LUMP)
      KNOW=K
      A=AK(I,J)
      RETURN
      END
```

d) Scratch Files

The skillful use of scratch files assigned to disk, disk-packs, tapes, or ECS can greatly reduce the amount of CM needed by a job. However, care must be taken so that the extra I/O time does not nullify any gains.

e) Buffer size

Each file used by a program has a buffer assigned to it. The system default size for such buffers is 1010_8 words. However, there are times when this is more than is necessary. In particular, the BUFFER IN and BUFFER OUT statements generally do not need a system buffer at all since data can flow directly between the program array and the I/O device. Modification of the buffer size can be done on the PROGRAM card by specifying FILENAM=nn where nn is the number of octal words desired. For example:

```
PROGRAM XYZ(INPUT,OUTPUT,TAPE1=200,TAPE2=0)
```

Parameters may also be supplied on the LGO card which will override the ones on the PROGRAM card.

9.1.2 CP time reduction

a) Optimization of object code

The easiest way to optimize CP code is to use Fortran Extended (FTN) with either OPT=1 or OPT=2. This is particularly useful in optimizing DO loops. Occasionally it may be desirable to hand-code highly used inner loops, but it must be realized that the debugging of Compass code is much more difficult than Fortran code. In fact, one should first ask whether optimization is worthwhile at all. Unless the program is to be run often and for long periods, the effort will probably not pay.

Some general principles should be mentioned. If an array is passed to a subroutine via COMMON instead of through the calling sequence, address computation is reduced. Division is slow, so, if the same divisor is to be used frequently, it is better to form its reciprocal and use it as a multiplier. Short loops, which fit into the eight-word instruction stack, are to be preferred to longer ones. Jumps, which interrupt the in-stack processing, are also expensive, and their use should be minimized.

Once it has been decided that it is worth improving the efficiency of the object code, it is recommended that debugging proceed using FTN(OPT=0), and that the final version of the program be compiled using FTN(OPT=1). The following table gives some very rough figures on relative execution speeds for a "typical" job, and for one which uses a considerable amount of DO loops and indexing (computation of eigenvalues) and is compute-bound. The figures are given relative to FTN(OPT=0), assumed to be 1.00.

	"Typical" job	Matrix job	Compile times
FTN(OPT=0)	1.00	1.00	1.0
FTN(OPT=1)	.67	.20	1.3
FTN(OPT=2)	.66	.19	1.6
RUN	.83	.55	.8

Note that there are reasons for using FTN other than optimization. These include better diagnostics, and upward compatibility (and CDC support) with future systems.

b) Reduction of wait states

A Wait state occurs whenever the CP has to await the completion of an I/O operation being performed by a PP. Not every Fortran I/O statement results in an I/O operation; in fact the operation is only done if the buffer is full, or the operation involves an end of record. It follows that short binary records are very expensive. BCD records are not as costly because each BCD line ends on a zero byte which is used to separate internal (e.g. print lines) records. Thus many of these internal records can be blocked into one physical record on either disk or SCOPE "Y" tapes. "X" tapes are therefore to be avoided unless the tape is used to communicate with a non-6600 computer system. A small record is one of less than 512 words for tape and less than 2000 words for disk. BCD IO is buffered into blocks of 128 words before transmission.

c) Reduction of jumps

The use of functions which insert in-line code, such as the shift functions LEFT and ALEFT and the compare function KOMPARE, are to be preferred to external functions which require a return jump to go to the function, and another jump to get back.

The use of array IO is to be preferred over an indexed list. The array form:

```
          READ(5,2)  A
```

is considerably faster than the implied DO loop in:

```
          READ(5,2)  (A(I),I=1,100)
```

assuming A has been dimensioned as 100. The latter uses much more CP time as it requires 99 return jumps to the subroutine INPUTC. This is true for binary and BCD, input and output. Furthermore, the statement:

```
          PRINT66,  (A(I),I=21,30)
```

could be speeded up by equivalencing:

```
          DIMENSION A(100),AA(10)
          EQUIVALENCE (AA,A(21))
                  .
                  .
                  .
          PRINT66, AA
```

Lists such as (A(I),B(I),C(I),I=1,100) should be rearranged as A,B,C where possible.

9.2 Tips and Tricks

9.2.1 Generation of fancy output

Computer paper is large, and, to the reader, generally unattractive, particularly if it comes in a thick stack. In order to keep the output compact it is sometimes desirable to divide a page into more than one column, but, it can happen that an item destined to the top of column 2 becomes available only after an item for half-way down column 1. Since it is not possible to backspace the printer, one can proceed as follows. A page-image is set up in memory, and the Fortran ENCODE statement is used to write output to the page-image, which is only printed when complete.

For example, suppose we want a page to consist of two columns of 50 characters each, separated by 10 spaces, with 55 lines per page. Consider:

```
          INTEGER PAGE(5,2,55), BLANK
          DATA BLANK/1H /
                  .
                  .
          DO 10 I=1,550
       10 PAGE(I)=BLANK                (Clear page-image to all blanks)
                  .
                  .
```

```
        ENCODE(50,101,PAGE(1,1,L1)) L1,X       (Write into line L1 of
                                                Column 1)
    101 FORMAT(* WRITING X ON*I3,*-TH LINE OF COL 1, X=*F6.2)
             .
             .
        ENCODE(50,102,PAGE(1,2,L2)) Y,L2       (Write into line L2 of
                                                Column 2)
    102 FORMAT(* Y=*F10.5,* ON LINE NO.*I3,* OF COL 2*)
             .
             .
       PRINT202,PAGE                   (Finally print page)
   202 FORMAT(1X,5A10,10X,5A10)
             .
             .
```

Alphanumeric output must be done while taking into account that only 10 characters can be stored in a word. However, if a given field contains 11 to 20 characters, one can define the variable to be double precision. For example:

```
        DOUBLE TEXT(2)
        DATA TEXT/20H ACCOUNT IS OVERDUE   ,20HACCOUNT IS PAID UP   /
             .
             .
        PRINT 33, TEXT(I)              (Print one message or the other,
   33   FORMAT(5X, 2A10)               depending on whether I=1 or I=2)
```

9.2.2 A Method for Testing for Negative Zero in Fortran

When a variable is read from a blank field with an "I" or "F" Fortran format specification, the variable is given the value -0. It is often desirable to detect a negative zero, but most Fortran IF statements will not distinguish between -0 and +0. For example, the expression in the statement

```
        IF(X.EQ.-0)  GO TO 20
```

is assigned the value "TRUE" for both X = -0 and X = +0, and the branch to 20 occurs in either case.

The statement

```
        IF(.NOT. X) 10,20
```

will cause a branch to 20 if X = -0 or a branch to 10 if X has any other value, including +0. The variable used in the expression may be either real or integer. This statement distinguishes -0 from all other numbers for the following reason. Fortran has the convention that a logical expression is false if it has the value +0, true if it has any other value. The .NOT. operation complements the value of the,variable, i.e., changes its sign.

Thus only the value -0 could make .NOT.X be false (+0). (Note that since "-X" is an arithmetic expression and is not in logical mode, the statement "IF(-X) 10,20" will not work.)

9.2.3 Free Format Cards

It may sometimes be necessary to read cards containing data which does not consist of BCD characters. Binary cards must have a 7-9 punch in column 1, so it may not be possible to use a binary read either. However, a "free format" mode is available. This is accomplished by sandwiching the free format cards between two cards each containing a 7777B in columns 1 and 2 and nothing else. Each of the free format cards is read in as 16 words.

Example:

```
          PROGRAM FREE(INPUT,TAPE5=INPUT)
          INTEGER XXX(32)
               .
               .
          READ(5) XXX
               .
               .
          END
     7-8-9 Record Separator
     Card with all 12 positions (=7777B) punched in columns 1 and 2
     Free format card
     Free format card                  (Two cards = 32 words)
     Card with all 12 positions (=7777B) punched in columns 1 and 2
     6-7-8-9 End of File
```

Further details may be found in the SCOPE Reference Manual, CDC Publication No. 60189400, page E-3.

9.3 Pitfalls to Avoid

9.3.1 Mixed Mode Expression

Both of our Fortran compilers (RUN and FTN) allow the use of mixed-mode expressions such as: X + J.

Within any level of parentheses, if there is an element (variable, constant, or sub-expression) of real mode, then all integer elements therein are converted to real modes before the arithmetic or relational operations are performed. Many other systems (and USASI Standard Fortran) do not provide mixed mode, and some which allow it handle it differently. Therefore, Fortran programs using it may perform differently or cause compiler errors on other machines.

Furthermore, mixed-mode expressions may cause unintended results, particularly when used within IF statements. For example, under RUN, the following statements will cause a jump to 3:

```
      J = 1HA
      E = 2.0
      IF (E.GT.3.0) GO TO 3
      IF (J.EQ.1HA)  GO TO 3
```

The following will not:

```
      J = 1HA
      E = 2.0
      IF (E.GT.3.0 .OR. J.EQ.1HA)  GO TO 3
```

The reason is that, since E and J are at the same level of parenthesization, J is converted to floating-point, before comparing it to the typeless hollerith constant. (This is handled differently by FTN.)

The LX option causes FTN to flag all non-USASI code, including mixed-mode.

9.3.2 Round-off errors

Like all finite word-length digital computers, the 6600 represents floating-point numbers with finite precision. This means that most rational numbers cannot be stored exactly - however, the round-off error is only about one part in 10^{14} (one part in 10^{29} for double-precision). This seemingly trivial error does effect results in certain cases. In a result which is the difference between two very close numbers, the small errors in those numbers may be large compared to the result. Similarly for quotients, tangents, etc. Even very small errors may effect a program, if decision statements are carelessly written:

```
          T = 1.0 / 3.0
          X = 3.0 * T
          IF (X.EQ.1.0) GO TO 20
```
(This code will not go to 20.)

Repetitive incrementation of a floating point number can result in an error of one bit in the low-order position (of the mantissa), for each incrementation. In some cases, such as differential equations the problem is critical. The code

```
          X = X + DX
```
is sometimes better as:
```
          I = I + 1
          X = XZERO + FLØAT(I)*DX
```

As there is no way in principle for a system to determine if precision has been lost, preventing errors due to floating-point round-off is always the responsibility of the user.

Note furthermore, that there is no guarantee that the code which converts data read from cards or tape will give identical results with the compiler code which converts Fortran constants. Thus if one variable, X, is set to a real (floating-point) number with an arithmetic assignment or a DATA statement, and another variable, Y, is set by a READ of a number having the same <u>decimal</u> representation, the <u>binary</u> representations may well differ in a low order bit, and a statement such as:

```
IF(X.EQ.Y)GO TO 25
```

may fail to go to statement 25. In such a case a better test would be:

```
IF(ABS(X-Y).LT.EPSIL)GO TO 25
```

where EPSIL is a suitably chosen small number.

9.3.3 Job Reruns

When designing the way in which a job is to be run, one must remember that at any stage during the execution of the job it may get stopped and restarted from the very beginning of the job. This can occur for several reasons:-

1. The entire operating system may have been stopped and restarted because a job met a bug in the system and stopped it.

2. The job may be RERUN, i.e., pushed back into the input queue by the operator because resources are needed for a higher priority (usually rush) job.

It, therefore, follows that a job which reads from a tape and writes back to the same tape may fail because of a restart. For example, control card sequences such as the following are definitely not suitable for rerunning. They cannot be detected by software and will eventually cause one a large amount of trouble trying to find out what went wrong.

Example of bad control cards:-

```
JOB Card
REQUEST,LISPPL,5,Y,W,UL9999.PERMIT
UPDATE(P=LISPPL,N=CLAM)
REWIND(CLAM,LISPPL)
COPYBF(CLAM,LISPPL)
```

In this example, the COPYBF card overwrites the tape which was read by the UPDATE card so if the job is restarted for any reason during the COPYBF operation, then the first part of the job will fail second time around. The only safe technique for this type of job is to use two tapes, one for the file LISPPL and one for the file CLAM.

9.3.4 Field Length and REQUEST Cards

The REQUEST control card reduces field length (FL) to 100, and restores the original FL when the next card is other than a REQUEST or COMMENT. To avoid unnecessary storage moves and delays, tape positioning and other operations should not interrupt a series of REQUEST cards, but should follow the last one. For example:

Good:	Very Inefficient:
REQUEST,TAPE1,...	REQUEST
REQUEST,TAPE2,...	SKIP
REQUEST,TAPE3,...	REQUEST
REQUEST,TAPE4,...	COPY
SKIP(TAPE1,2)	REQUEST
COPYCR(TAPE2,NULL,7)	REWIND
COPYBR(INPUT,DATA)	*QUIXOTE
REWIND(DATA)	REQUEST
*QUIXOTE.	

9.4 Hints for Remote Users

9.4.1 FOCUS Users

Use of the FOCInnn file. In the SEND command, when one specifies Remote Files (i.e. those to be returned to the FOCUS user), one may specify a single octal digit, usually one. This is made to correspond to FOCInn0, where nn is the job's sequence number. Furthermore, a FOCUS user who sends a Fortran job for execution on the 6600, will frequently want to have returned to his terminal either the output from his job if it ran successfully, or his Fortran diagnostics.

A program called PEANUTS searches through RUN output for diagnostics which it writes to a file called FOCUS0 (Focus-zero). This file is returned to the remote under the name FOCInn0.

The following control cards will return diagnostics, if any, in FOCInn0; the OUTPUT file can also be remoted.

```
RUN(S)                   Compile program
LGO.                     Execute if compilation OK
CXIT.                    Here if compilation error
PEANUTS.                 Retrieve diagnostics
```

The SEND command takes the following form:

```
********COMMAND SEND CC FORT.1.
```

where CC is the file of control cards, FORT is the Fortran program, and 1 specifies that FOCInn0 is to be returned.

A user program can be coded to write its output to FOCUSO. A simple but clumsy way is:

```
RUN(S)
REWIND(OUTPUT)                 Overwrite compiler output if no errors
LGO.
REWIND(OUTPUT)
COPYBF(OUTPUT, FOCUSO)         Copy job output to FOCUSO
REWIND(OUTPUT)
COPYBF(NULL,OUTPUT)            Destroy OUTPUT so as to avoid
                               printing it
```

Another way is to specify FOCUSO on the PROGRAM card:

```
PROGRAM XYZ(FOCUSO,TAPE1 FOCUSO)
     .
     .
     .
WRITE(1,123) ...
```

OUTPUT can be redefined to be FOCUSO via the LGO or EXECUTE card: if the PROGRAM card is:

```
PROGRAM ABC(OUTPUT, .... )
```

then the control card:

```
LGO(OUTPUT)    will cause all PRINT statements to write FOCUSO.
```

If a MAP(ERR) control card has been used, the storage map can be returned to the remote in the even of an execution error:

```
RUN(S)
MAP(ERR)                       Write storage map to file MAP
LGO(FOCUSO)                    Substitute FOCUSO for first file
                                 on PROGRAM card
CXIT.
PEANUTS.
EXIT.                          Here if execution error
REWIND(MAP)
COPYBF(MAP,FOCUSO)
```

9.4.2 BROOKNET Users

In the event of a BROOKNET crash, all files attached to BROOKNET are returned to the catalogue so that their contents are saved. However, the next time any such file is referenced, it will be re-attached by BROOKNET in a rewound position. Thus if a remote continuously writes to a file, and, somewhere along the way BROOKNET crashes and starts up again, the beginning of the file could be inadvertently overwritten. Remotes should be aware of this danger, and take steps to test for this condition.

First though, it should be noted that in the event of a system crash, the contents of the file are saved as of the last time the directory was updated. If a remote writes continuously to a file, it should, periodically, either perform an UPDATE (Function 15) or a WRITE END-OF-FILE (Function 11).

In order to detect whether a file has been rewound, a remote can, periodically, (or before every write if the data is critical) perform a SKIP BACKWARD of one record followed by a READ. Naturally this assumes that the records are sequentially numbered or identifiable in some way. Some remotes combine the update and the rewind check by performing a WRITE END-OF-FILE, a SKIP BACKWARDS of two records, and a READ. This positions the file ready to overwrite the End-of-File and keep going. It may sometimes be more convenient to perform the rewind test on a file other than the one being continually written. In fact a special file can be created for just this purpose. Note, however, that every catalogue update is very, very time consuming. Therefore each remote user must decide if insuring against rare loss of data is worth the greatly increased overhead.

The fact that BROOKNET catalogues all its files opens some further pitfalls for the unwary. Under normal operations, BROOKNET does not return a file to the catalogue unless explicitly requested to do so. Thus a user who tries to ATTACH a BROOKNET file with a permission other than Read will hang indefinitely on "File Busy", unless he uses the BATACH control card or the BFATACH subroutine which cause BROOKNET to return the file and so make it available to others. Here too, care must be taken to specify the file exactly as it is known to BROOKNET. In particular, if a cycle number is specified, it must be the correct one. If no cycle is specified, (or a zero cycle number), then BROOKNET will return the first file matching on PFN and LFN.

If the BATACH specifies the remote number instead of the PFN, the file will not be found if it is no longer attached to BROOKNET or if BROOKNET is in the other machine. For these reasons, it is best to always specify both PFN and LFN.

9.5 Program Debugging

Errors fall into three main groups:

- Wrong answers, including loops resulting in TIME LIMIT or LINE LIMIT
- ARITH ERRORS
- Errors detected by system software and processed by subroutine SYSTEM.

It is difficult to formulate general rules for the discovery of the first of these error types. Detection of the others can be aided by error message, MAP and DUMP.

Subroutine SYSTEM outputs a message when it detects an error. Some errors are non-fatal and allow execution to continue. It is possible to modify the error procedure and make fatal errors non-fatal, suppress error print-outs, etc. This is done through a call to SYSTEMC, as described in Computer Center write up No. 1044.

The system default is not to produce a storage map at load time. If one is desired, the loading sequence must be preceded by either a MAP (PART) or a MAP(ON) control card. The former, which is usually adequate, lists the load point of the main program, and all subroutines, both user and system supplied. The full map also includes a list of every entry point and all references to it. A useful option is to use the control card MAP(ERR). This generates a full storage map on a file named MAP. This file can then be copied to the output file if and only if an error occurs.

For example:

```
JOB Card
RUN(S)
MAP(ERR)
LGO.
EXIT.
REWIND(MAP)
COPYCF(MAP,OUTPUT)
```

An example of a partial storage map is given in figure 9-1.

The dump produced by an error prints the contents of all registers, and the area of memory where the problem was detected. DMP control cards can be used to print other areas of memory.

ARITH ERRORS

These errors are detected by the system hardware on either a reference to an address outside of the job's field length (MODE 1), or the use (in an arithmetic operation) of an infinite operand (MODE 2) or an indefinite operand (MODE 4).

If an ARITH ERROR - MODE 1 occurs, the first place to check is in the storage map under "UNSATISFIED EXTERNALS". These are routines called for, but not provided. A subscripted variable omitted from a DIMENSION statement results in an unsatisfied external. Another possible cause is the use of a subscript whose value exceeds the size of the array as specified in the DIMENSION statement.

If ARITH ERROR - MODE 2 or 4 occurs, the output should first be checked for non-fatal error messages. Certain subroutines return an indefinite result on an "impossible" condition; e.g. SQRT(negative number).

To determine where in the program an error occurred, one should proceed as follows: The absolute location of the error is given either in the dayfile in the case of an ARITH ERROR, or in the P-register as shown in the dump. From the storage map, determine in which subprogram this address lies, and then subtract from the address of the error the load address of the subprogram. This result gives the approximate location of the error within the offending routine. For example, referring to figure 9-1, if a MODE 2 error occurred at location 3050, one computes 3050 - 3021 = 27 (in octal arithmetic) Thus one surmises that the error occurred in the vicinity of location 27 of subroutine MATMUL.

```
CORE MAP  14.35.11.  NORMAL                 CONTROL
      ---TIME---LOAD MODE --L1--L2----TYPE---------------USER---+---CALL--------/
      FWA LOADER  034167  FWA TABLES  033651
      -PROGRAM----ADDRESS-                         --LABELED---COMMON--
      MATRIX       000162                                  CNAME       000100
      MATMUL       003021
      PRNTMAT      003075                                  CNAME       000100
      BNSINV       003152
      SYSTEM       003710                                  SCOPE2      003710
      RANF         005022
      OUTPTC       005035
      SQRT         005132
      SIO$         005175
      REL1         006533
      KODER        006610
      GETBA        010127
      ---UNSATISFIED EXTERNALS-----                          REFERENCES

                                                 000100    010766    010146    000620
                                          /------FWA LOAD--LWA LOAD--BLNK COMN--LENGTH--
```

Notes:

1) Loading starts at location 100 with the labelled COMMON block CNAME, 50 (=62_8) words long.

2) Loading ends at location 10766.

3) Blank COMMON starts at 10146 and is 400 (=620_8) words long.

4) MATRIX is the main program, MATMUL, PRNTMAT, and BNSINV are user supplied subroutines. All others are system supplied.

5) The labelled COMMON block, CNAME is shared by the main program and subroutine PRNTMAT

6) There are no unsatisfied externals.

Figure 9-1: Storage MAP(PART)

APPENDIX A

CHARACTER CODES

CDC 63-CHARACTER SET

Display Code	Character	Hollerith (026)	Hollerith (029)	External BCD	Display Code	Character	Hollerith (026)	Hollerith (029)	External BCD
00	(none)†			16	40	5	5	5	05
01	A	12-1	12-1	61	41	6	6	6	06
02	B	12-2	12-2	62	42	7	7	7	07
03	C	12-3	12-3	63	43	8	8	8	10
04	D	12-4	12-4	64	44	9	9	9	11
05	E	12-5	12-5	65	45	+	12	12-8-6	60
06	F	12-6	12-6	66	46	-	11	11	40
07	G	12-7	12-7	67	47	•	11-8-4	11-8-4	54
10	H	12-8	12-8	70	50	/	0-1	0-1	21
11	I	12-9	12-9	71	51	(	0-8-4	12-8-5	34
12	J	11-1	11-1	41	52	)	12-8-4	11-8-5	74
13	K	11-2	11-2	42	53	$	11-8-3	11-8-3	53
14	L	11-3	11-3	43	54	=	8-3	8-6	13
15	M	11-4	11-4	44	55	blank	no punch	no punch	20
16	N	11-5	11-5	45	56	, (comma)	0-8-3	0-8-3	33
17	O	11-6	11-6	46	57	. (period)	12-8-3	12-8-3	73
20	P	11-7	11-7	47	60	≡	0-8-6	8-3	36
21	Q	11-8	11-8	50	61	[	8-7	8-5	17
22	R	11-9	11-9	51	62	]	0-8-2	12-8-7	32
23	S	0-2	0-2	22	63	:(colon)†	8-2	8-2	00*
24	T	0-3	0-3	23	64	≠	8-4	8-7	14
25	U	0-4	0-4	24	65	→	0-8-5	0-8-5	35
26	V	0-5	0-5	25	66	∨	11-0 or	11-0 or	52
27	W	0-6	0-6	26			11-8-2	11-8-2	
30	X	0-7	0-7	27	67	∧	0-8-7	12	37
31	Y	0-8	0-8	30	70	↑	11-8-5	8-4	55
32	Z	0-9	0-9	31	71	↓	11-8-6	0-8-7	56
33	0	0	0	12	72	<	12-0 or	12-0 or	72
34	1	1	1	01			12-8-2	12-8-2	
35	2	2	2	02	73	>	11-8-7	0-8-6	57
36	3	3	3	03	74	≤	8-5	12-8-4	15
37	4	4	4	04	75	≥	12-8-5	0-8-2	75
					76	¬	12-8-6	11-8-7	76
					77	;(semicolon)	12-8-7	11-8-6	77

†When the 63-Character Set is used, the punch code 8-2 is associated with display code 63, the colon. Display code 00_8 is not included in the 63-Character Set and is not associated with any card punch. The 8-6 card punch (026 keypunch) and the 0-8-4 card punch (029 keypunch) in the 63-Character Set are treated as blank on input.

*Since 00 cannot be represented on magnetic tape, it is converted to BCD 12. On input, it will be translated to display code 33 (number zero).

APPENDIX B

PRINT CONTROL CHARACTERS

The leftmost character of a print line is not printed, but is interpreted by the print routines as a control character. The following table describes the effect of the permissible control characters:

Character (Column 1)	Action Before Printing	Action After Printing
A	Space 1	Eject to top of next page
B	Space 1	Skip to last line of page
1	Eject to top of next page	No Space
2	Skip of last line of page	No Space
+	No Space	No Space
0	Space 2	No Space
-	Space 3	No Space
blank	Space 1	No Space

When the bottom of a page is reached, the print routines eject to the top of the next page, thus skipping four lines. This may be undesirable if the printer is being used to produce a continuous graph. It is possible to suppress all printer controls, including the bottom of page eject. To turn off the printer control, either the letter Q must be placed in column 1 of the print line, or the letters PF in columns 1 & 2. To restore print control, the letter R or the letters PN are to be used. When Q or R is used an eject occurs before the control is put into effect; this does not happen for PF/PN. In either case, the remaining content of the line containing either the Q/R of the PF/PN is not printed.

Note that the FOCUS printer on the CDC 3200 recognizes only control characters 1, 0, blank and +.

APPENDIX C

DISK AND TAPE HARDWARE CHARACTERISTICS

Listed below is the storage capacity for the various disk units used in our system. (The conversion from megacharacters to physical record unit is 640 characters per PRU.)

Disk Unit	Capacity Megacharacter	KPRU
6603	74.70	116.72
841	35.60	55.63
844-2	118.66	185.41

With data recorded at different densities and block sizes, the storage capacity for a 2400 ft. magnetic tape is listed below:

Density BPI	Block Size Characters	Capacity Megacharacters	KPRU
200	80	2.00	25.00
556	80	2.58	32.22
800	80	2.71	33.88
200	136	2.74	20.14
556	136	3.94	28.95
800	136	4.26	31.30
200	800	4.85	6.06
556	800	10.53	13.16
800	800	13.12	16.46
200	1280	5.16	4.03
556	1280	32.08	9.44
800	1280	15.69	12.26
200	5120	5.58	1.09
556	5120	14.80	2.89
800	5120	20.63	4.03

Figures for the access time and maximum transfer rate are given below. For the disk units, the access time is computed on the basis of average seek time and rotational delay.

Device	Access time msec.	Max. Transfer Rate kilocharacters/sec.
6603	160	1250
841	85	420
844-2	40	6800
ECS	0.0032	100000
200 BPI, 80 char. block	5	10
800 BPI, 5120 char. block	5	100

INDEX

<u>COMMENT SHEET</u>

Title: BROOKHAVEN COMPUTER USER'S GUIDE

The Applied Mathematics Department solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual:

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference)

General comments:

FROM: NAME:

DEPARTMENT:

Please staple or scotch tape this sheet and send to the address on the back.

Librarian
Central Scientific Computing Center
Applied Mathematics Department
Brookhaven National Laboratory
Upton, New York 11973