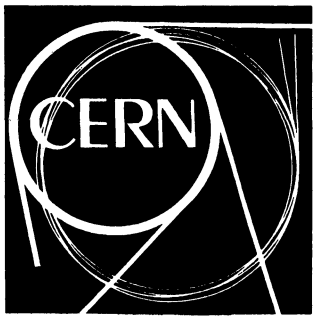


COMPUTER CENTRE

CDC 7600 USERS GUIDE

For further help,
Programming Enquiry Office,
Building 513 {8-935 or 4952}

DD-US-1
January 1977

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

IF YOU NEED HELP...

COMMUNICATION OPERATOR		513/R	4927
OPERATIONS Group - J. Ferguson		513/R-037	4935
SOFTWARE Group - T. Bloch		31/2-028	4949
USER SUPPORT Group - C. Jones		513/1-007	4884
COMPUTER COORDINATOR Rudy Bock until February 1977			*8-786
PROGRAMMING ENQUIRY OFFICE (PEO) Computer Center 9.00 - 12.30 &	14.00 - 17.30	513/1-014	4952
TC RIOS	9.00 - 10.30	13/1-019	3533
OC RIOS	11.00 - 12.30	2/1-034	2377
DOCUMENTATION OFFICE F. Morice - J-L. Penaud		513/1-011	2371
TAPE RECEPTION	8.30 - 17.00	513/R	4939
REGISTRATION New users - Divisional Representative (see list in Chapter 1) INTERCOM users - A. Oude-Moleman		513/1-003	5029
PROGRAM LIBRARY Distribution of Library Material D. Dupraz - G. Berger		513/1-015	4951
Help with Library programs F. James - T. Lindelof		513/1-017	4959
Mathematics and Computing Service Advisory Service - B. Schorr		31/3-024	4120
PHYSICS DATA HANDLING ALGORITHMS K. Gieselmann		27/2-020	4861
COMPUTER SCIENCE LIBRARY - J. Megies		513/1-024	2379
GD3 GRAPHICS - M. Howie		31/2-007	2993
MICROFICHE - J-C. Juvet		513/R-037	4935
FOCUS DATA LINK - I. McLaren		513/1-009	5010
TERMINALS - M. Sheehan		513/R-034	3348

Note - An up-to-date version of this page is monthly given in the Computer Newsletter.

1.1 INTRODUCTION

The CERN Central Computing Facilities, situated in the Computer Centre, Building 513, offer a variety of services based on a group of CONTROL DATA computers, the largest of which is a CDC 7600. A further set of services based on an IBM 370/168 will start their evolution in January 1977. The IBM services will be covered in an IBM Users Guide (DD/US/4) to be made available in 1977. This 7600 Computer Users' Guide refers only to the CDC based facilities and in particular to usage of the 7600.

In this chapter we have grouped not only introductory material but also some organisational information that is not directly mentioned elsewhere in the guide. The first two sections contain brief overviews of the computer configuration and of the services offered. These are intended for the completely new user and certainly do not tell the whole story. Certain keywords are highlighted and may point to further information in this guide via an index search. The section WHO, WHEN AND WHERE summarizes some of the organisational information of who to contact or where to go for what service. This information has a habit of changing and for the most up-to-date information, consult the back pages of the latest **Computer Newsletter** or give a call to the **Program** Enquiry Office - tel 4952 or bleep 8-935.

Finally in the introductory Chapter, we include a description of the architecture and features of the CDC 7600 Computer.

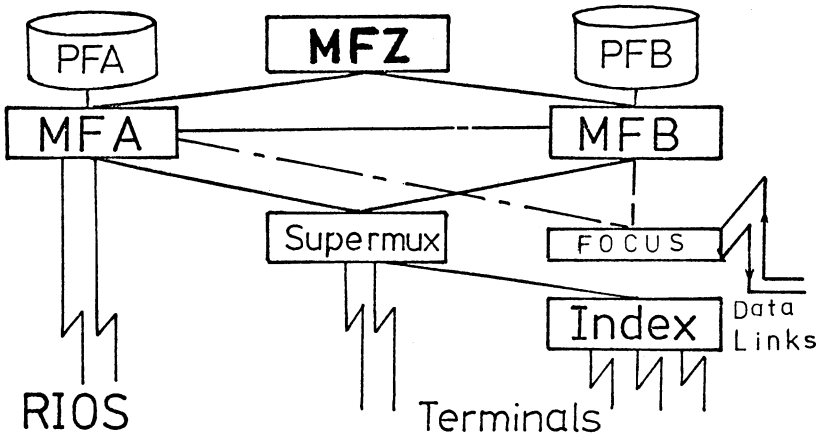
1.2 MACHINE CONFIGURATION

The configuration of the CDC Computers is shown in the figure below. The 7600, which is also known as MFZ (mainframe Z), is an extremely fast, 60-bit wordlength computer oriented towards scientific floating-point calculation. Operated at CERN under SCOPE 2.1.2 system, it runs only batch jobs and has no peripherals other than its large and fast working disks. For further description of the 7600 and its architecture see Section 1.5 of this Chapter.

The jobs and data necessary to keep the 7600 at work are 'staged' to and from the 7600 working disks by two 'front-end' CDC 6000 computers, known also as MFA and MFB. At present MFA is usually the CDC 6400 and MFB is the dual-processor CDC 6500. (The 7600 is roughly ten times more powerful than a 6400). These front-end computers control the two **separate** permanent file bases where users may store programs and very limited data samples for testing (see Chapter 8 for rules). They also control fifteen 7 and 9 track tape units (up to 1600 B.P.I.). Jobs in the 7600 which require files from disk or tape actually generate jobs (SPun - Off Tasks, SPOTs) in the 6000's, which transfer this data via the STATION software to the 7600. Files may also be transferred between the two 6000's by a similar technique.

The operating system of the two 6000's is SCOPE 3.4.3. Running under this system, the INTERCOM time sharing system provides Remote Access Facilities for both Terminals and Remote Input Output Stations (RIOS) distributed all over the site.

Finally in this brief configuration overview there are a pair of CDC 3000 computers which via the FOCUS system provide data links to the user data acquisition mini computers on the experimental floor.



1.3 SUMMARY OF SERVICES

The aim of this section is to provide new or inexperienced users with an overview of some of the more important features of the service offered on the CDC computers. For the sake of continuity none of these features are described in detail in this section. Further information may be found elsewhere in this guide via the index.

The CDC 7600 provides a powerful central batch processing service mainly for user-written FORTRAN programs, although other compilers and assemblers also exist, e.g. COMPASS, ALGOL, COBOL, MORTRAN, PASCAL, BCPL. A comprehensive set of applications programs may be found in the CERN Program Library (PROGLIB) and in a growing library of physics code known as PHLIB.

Jobs are automatically put into **Job Classes** depending on the user's requests for time and/or magnetic tapes on the job card. About twenty user jobs are run concurrently in execution environments referred to as 'control points', and the **scheduler** has a certain flexibility in the number of these control points given to each job class. For example, in the daytime the scheduler aims to give a fast turnaround for express and short jobs at the expense of medium and long jobs. In addition to these normal jobs there are four types of **PRIORITY** jobs available to users - P3 priority for urgent feedback to running experiments, P2 for other urgent work, P1 for guaranteed production turnround and P0 'half-price' computing for work that can wait until all normal work is completed. The cost in Swiss francs is calculated for each job on the basis of time, memory, I/O and tapes used. Although no real money is involved the amount is deducted from the user's budget. These budgets are allocated at the group level in the EP Division (who use 80% of the computer time) and at the divisional level otherwise via a committee known as **COCOTIME**. All users must **register** in order to use the central computers.

The 7600 has large working disks on which users may put programs or data samples in either of two forms, neither of which should be regarded as permanent since no guarantees are given. **FIND files** stay on the disk until demand for new space causes them to be purged. They usually stay for some hours thus providing a very useful debugging service, saving tape mounting and speeding up turnround. The ***P files** are a privilege granted to production users, saving them multiple tape merging problems by allowing them to accumulate up to a tape of production output on the 7600 disks. The limited space available requires that the *P files are closely rationed and controlled in order not to affect the general users' FIND service. Allocations for *P file space are granted by the **Computer Coordinator**.

SUMMARY OF SERVICES

The two 6000 computers act as front-ends to the 7600, controlling peripheral devices such as tape units, disks, line-printers, card readers, etc.., staging this data to and from the 7600 working disks. Besides keeping the 7600 well fed with work the front ends provide remote working facilities for both Terminals and Remote Input-Output Stations (RIOS).

The tape units, both 7 and 9 track, can read and write tape densities up to 1600 B.P.I. Services such as allocation of new tapes, cleaning tapes and introduction or removal of tapes from the Centre are handled via the **Tape Reception** in the Computer Centre. A **'closed-shop'** service giving a controlled improved performance is provided for 9-track tapes in the 60000 to 69999 number range. These tapes belong to the Computer Centre and are never allowed to leave the building. There are three 7 track tape units and twelve 9 track tape units, of which 4 are assigned to the closed-shop service.

The two **separate Permanent File bases** (one for each front-end) provide space where users may store programs and even short data samples. These file bases are budgetted and controlled daily. The rules which you must follow if you want your files to be really permanent are given in Chapter 8. It is possible to transfer files between the two file bases but there should be little need for most people to use more than one file base.

Although the Computer Centre provides a central user working area in Building 513, **remote working facilities** are provided all over the CERN site by means of more than 70 **terminals** and 10 **Remote Input-Output Stations** (RIOS). **A Delivery Service** for large outputs, microfiches, cards, magnetic tapes, plots, etc.... helps to minimize the number of visits which are necessary to the Centre.

TERMINAL locations

All these terminals (as well as the RIOS and their associated job enquiry terminals) are connected to the **INTERCOM time sharing** system which runs in the two front-end 6000's. INTERCOM provides facilities ranging from editing and job submission (to the 7600) to fully interactive programming and debugging in the 6000. Many terminals are connected via **INDEX** (INdependant Digital EXchange) and most are connected through the CERN developed concentrator SUPERMUX. INDEX, which acts in a similar way to a telephone exchange, allows many more terminals to be scattered around the CERN site than can be connected at any one time. The SUPERMUX concentrator allows the user to choose to which 6000 computer he should be connected. The service on MFA is aimed at the 7600 user who wishes to edit source files, submit jobs to the 7600 and inspect his output at the terminal. The service on MFB is aimed more at those users who wish to do interactive computing in the 6000. For further information on INTERCOM, see the pocket card INTERCOM at CERN (DD/US/2).

1.4 WHO, WHEN AND WHERE

This section attemps to summarize to whom one should go for various services offered and as such, is hopefully correct at the time of printing (December 1976). Inevitably these things change but we try to publish the current telephone number of essential services and most changes in the monthly Computer Newsletter which is received by all registered users.

REGISTRATION TO BECOME A USER

In order to use the Central Computers you must become a registered user. This is usually done by contacting your Divisional Representative for Computing (see list below) who will arrange for the necessary formalities. If you are in any doubt who to contact, call or write directly to Ans Oude-Moleman, DD Division, in the Computer Centre (tel 5029). Any user who wishes to use the time sharing service INTERCOM must supply a password in addition to the above, also by calling the above number.

THE 7600 COMPUTER USER'S GUIDE

This is the manual you are reading. Every registered user automatically receives the new issue of the 7600 Computer User's Guide (about every 8 months). It is constantly updated and the newest version is always available under the MFB machine using the macro WRITEUP described below. With this guide it is intended to give the Computer user 'up-to-date' technical and practical details about computer service, software, job applications, etc... mainly using examples of complete jobs. The first page summarizes the different services related to the Computer with corresponding phone numbers. Chapters 9, 10 and 11 of this guide are to be used as the computing first aid, helping the user to find errors occuring most frequently. The last chapters contain the catalog of the 7600 Program Library and a glossary of abbreviations and terms commonly used around the Computer Centre and inside this guide. At the very end, one can find a keyword index which allows quick retrieval of information.

Copies of this guide can be obtained from the Computer Documentation office (Building 513, 1-019).

WHO, WHEN AND WHERE

THE COMPUTER NEWSLETTER

The Computer Newsletter (CNL) is the Computer Centre's principal means of communication with computer users. All registered users receive it. It contains news of changes to the system and service, descriptions of new programs and facilities as well as an up-to-date list of people to contact for specific services and problems. Users are strongly recommended to read it.

DOCUMENTATION - MANUALS AND NOTES

The Computer Documentation Office, Building 513, 1-019, holds copies of all the CDC manuals and the Program Library manual and provides updates to them. Other documentation described below is available from the Self-Service Room, Building 513, 1-023.

CDC Computer manuals

Although the 7600 User's Guide provides enough information for most 7600 users, more detailed information can be obtained from the CDC manuals which are available in the RIOS and Terminal Rooms. The titles of the more commonly used manuals are listed below.

Specialists should refer to the CDC publication catalog in the Computer Science Library or the Documentation Office. Some manuals have recently changed their name and reference numbers. In the list below, old reference numbers are given in brackets. Each manual has an eight digit reference number followed by a letter - X below - indicating the revision. New updates are announced in the Computer Newsletter. In order to receive them, please contact F. Morice or J-L. Penaud, tel 2371.

FORTRAN users will find more information in -

FORTRAN Extended (Version 4) Manual	60497800X (60305600)
7600 SCOPE 2.1.2 Reference Manual	60342600X (60342600)

People new to the 7600 are recommended to read the SCOPE 2.1.2 User's Guide, which has good descriptions of file and tape handling -

SCOPE 2.1.2 User's Guide	60372600X (60372600)

and for more details -

LOADER Reference Manual	60429800X (60344200)
Record Manager Reference Manual	60495700X (60307300)
Record Manager User's Guide	60359600X (60359600)
7600 Reference Manual	60258200X (60258200)

WHO, WHEN AND WHERE

COMPASS Reference Manual 60492600X (60360900)

If UPDATE is chosen to maintain source programs, refer to -

UPDATE Reference Manual 60449900X (60342500)

In order to use the INTERCOM system -

Interactive INTERCOM Guide 60359700X (60359700)
INTERCOM 4.3 Reference Manual 60307100X (60307100)
6000 SCOPE 3.4.3 Reference Manual 60307200X (60307200)

CERN Program Library Manual

The Program Library manual containing a short write-up of all the routines in the CERN Program Library (see catalog at the end of this guide), is also distributed by the Documentation Office. Please contact F. Morice or J-L. Penaud, tel 2371.

Self-Service Documentation - Write-ups and Notes

In the Self-Service Room, Building 513, 1-023, one can find the following documentation -

-CERN Program Library long & short write-ups of individual routines
-CERN Computer Newsletter
-User Support Group publications (DD/US notes) including -

INTERCOM Pocket card	DD/US/2
Comparison between FTN and FORTRAN H Extended	DD/US/3
FICHE User's Guide	DD/US/6
MACRO User's Guide	DD/US/7
GD3 Graphic Display User's Guide	DD/US/8
Paper Tape input/output Terminal	DD/US/9
PEJOB ASCII Printing User's Guide	DD/US/12
FLOP FORTRAN Oriented Parser	DD/US/13
BCPL User's Guide	DD/US/14

-HBOOK Histogram Booking (CERN Library Y250)
-HPLOT Histogram Plotting (CERN Library Y251)

-Physics Data Handling Algorithms and Notes
(The Documentation Office only holds updates to this documentation. In order to receive it regularly, contact K. Gieselmann, tel 4861.)

WRITEUP macro

If one wishes to get any of the guides listed below and referenced as User Support Group Notes (DD/US/), as a Computer output, one should use the WRITEUP macro on an MFB terminal -

WHO, WHEN AND WHERE

```
ATTACH,WRITEUP
WRITEUP,guide,NAME=username
```

Only one guide can be requested at a time. It will be printed in Building 513 using the first five characters of 'username' as a banner page. At present, it can be any member of the following list -

CDC (7600 Computer User's Guide)	DD/US/1
INTERCOM (pocket card)	DD/US/2
FICHE (Microfiche)	DD/US/6
MACRO (Macros Facility)	DD/US/7
GD3 (Graphic Display)	DD/US/8
PTAPE (Paper Tape reading/punching)	DD/US/9
PEJOB (ASCII file printing User's Guide)	DD/US/12
HBOOK (CERN Library Write-up Y250)	
HPLOT (CERN Library Write-up Y251)	
FLOP (FORTRAN Language Oriented Parser)	DD/US/13
BCPL (BCPL User's Guide)	DD/US/14

COMPUTER SCIENCE LIBRARY

The Computer Science Library is situated in Building 513, Room 1-024, and open all the time. The receptionist is present from 8.30 to 12.30, tel 2379.

PROGRAMMING ENQUIRY OFFICE - PEO

Users who need advice or assistance with computing problems can make use of the Programming Enquiry Offices. They should bring with them any relevant outputs, dayfiles and so on.

The main Programming Enquiry Office is in the Computer Centre, Building 513 - Room 1-014. It is open from 9.00 to 12.30 and from 14.00 to 17.30. The telephone number is 4952 and the bleeper number 8-935.

Members of the PEO try to visit the Old Center and TC RIOS as regularly as possible. The PEO in TC, Building 13, 1-019, telephone 3533, is open from 9.00 to 10.30, and the PEO in OC, Building 2, 1-034, telephone 2377, is open from 11.00 to 12.30.

However,for advice or problems of a purely mathematical nature, help may be obtained from the Mathematics and Computing Section . Contact Benno Schorr, who will put you in touch with the appropriate person.

WHO, WHEN AND WHERE

TIME TABLE

The 7600/RIOS (MFZ/MFA) system is available to the users from Monday 9.45 to next Monday 6.30 with two interruptions during the week, on Tuesday night from 19.00 until Wednesday 01.00, and from Thursday night 21.00 until Friday 01.00.

The MFB machine is available every day with interruption from 6.30 to 9.45 on Monday, from 7.00 to 8.30 on Tuesday, Wednesday, Thursday and Friday, and from 6.30 to 12.30 on Sunday. It is available all day Saturday, but it is clear that one has no access to the 7600 through MFB during the MFZ/MFA interruptions.

COMMUNICATION OPERATOR - SPECIAL JOBS

The Communications Operator is the person to contact (tel 4927) if your job requires any special intervention. Many special jobs are best run overnight in which case the job together with its 'Special Job Form' may be given to the Communication Operator on duty in the Computer Centre. You should also contact the Communication Operator if you have a problem with equipment such as the RIOS, terminals or hard copy device.

CERN PROGRAM LIBRARY

The CERN Program Library consists of a large number of routines covering any kind of calculation from numerical integration to character manipulation. A list of all the routines available is contained in Chapter 14. The Program Library Manual consists of a short one page write-up of all the routines and longer write-ups, where necessary, can be obtained from the Self-Service Documentation Office.

PHYSICS DATA HANDLING ALGORITHMS AND NOTES

Data Handling Algorithms and Notes are intended as means of communication for physicists and programmers working on problems of data analysis. The pool of algorithms should be regarded as an informal extension of the CERN Program Library sections on Application Programs and High Energy Physics Programs. No guarantees are made for the validity or maintenance of these programs. A list of abstracts and a description of how to access the material is available from the

Self-Service Documentation Office or from Karin Gieselmann (tel 4861).

MAINTENANCE OF SOURCE PROGRAMS

Two utility programs, **UPDATE** and **PATCHY** are available for the maintenance of program source code. UPDATE is a CDC utility available on all large CDC machines. PATCHY is a FORTRAN program and has been installed on several CDC, IBM 360/370, PDP 10 and UNIVAC 1100 machines in High Energy Physics Laboratories. For further details on both of these utilities, see Chapter 4.

GRAPHICS FACILITIES

The main graphics package available is called GD3. It creates an output file that can be plotted on the line printer, **CALCOMP** plotter or visual display terminal according to the users wishes. For further information, see the GD3 User's Guide (DD/US/8) and the GD3 Long Write-up (CERN Program Library J510).

At present, December 1976, plotting is done on-line on simple Computer Instrumentation plotters, but in 1977 an off-line plotting service will be introduced using a CALCOMP 936-S plotter with the possibilty of having wide (90 cm) paper and coloured pens.

PAPER TAPE

A self-service paper tape reading and punching facility is available on the ground floor of the Computer Centre. For a description of how to use it, see Paper Tape Input/Output Terminal , (DD/US/9).

MICROFICHE

A microfiche is a sheet of film, 15 by 10 cm, on which printout is recorded by an electron beam. Normally 207 pages of computer paper output can be written on one fiche, so it provides a very compact way of storing output. See Chapter 5 for details of how to use it.

WHO, WHEN AND WHERE

MACROS

The macro facility on the CDC machine is roughly equivalent to the catalogued procedures of IBM. It allows the user to define a set of control cards to which parameters can be passed. These parameters may be keyword or positional, and can be used in a substitutional or conditional manner. For further details, see Chapter 5.

DIVISIONAL REPRESENTATIVES IN CUAC

Your Divisional Representative may well be a very useful person with whom to make contact, since many special arrangements, particular to your division are often handled by them. Your representatives can also bring your wishes, suggestions, complaints, etc..... to the notice of the Computer Users' Advisory Committee (CUAC) which is appointed to advise the computer centre in matters arising around the service offered. This can vary from the positioning of terminals or RIOS to the assigning of priorities amongst a list of user requests, etc...

The current list of formal Divisional Representatives is the following -

		Room	Extension	Bleep
EP	Mrs L. Griffiths	4/1-072	2660	*8-968
EF	H. Wenninger	36/3-016	4097,4270	*8-361
DD	H. Grote	31/3-008	4960,4961	*8-568
FI	J-D. Mandica	5/R-	2823	
PS	P. Skarek	26/1-003	2213	
ISR	B. Zotter	30/6-008	3034	
HS	M. Hoefert	24/1-023	4602,3893	
PE	M. Baboulaz	5/1-018	4484,4126	
SB	A. Lecomte	54/1-023	3273	
SPS	C. Iselin	3/1-028	3657	
TH	F. Schrempp	4/2-064	2445	
DG/DI	G. Lindecker	60/4-008	5886	

REMOTE INPUT/OUTPUT STATIONS

At present there are ten RIOS stations (Remote Input/ Output Stations) for the 7600. The RIOS stations (in order, roughly, of usage) are located as follows-

WHO, WHEN AND WHERE

	Building	Room no.	Tel.	Delivery	Dispose
OC, O2	-2-	1-032	4263	X1	CCPIOC, CCPIO2
TC, T2	-13-	1-018	4244	X3	CCPITC, CCPIT2
TCL	-17-	S-027	4060	X7	CCPITL
ISR	-112-	1-013	4208	X4	CCPIIR
West Hall	-582-	R-017	4997	X6	CCPIWH
SPS	300 Gev	C-01	5262	X5	CCPISP
NP	-22-	1-031	3991	X8	CCPINP
SB	-4-	S-025	2284	X2	CCPISB
	-54-	R-035		X9	

The first two are double RIOS. Building 54 is only a delivery point.

Each of these Remote Input/Output Stations is equipped with a card reader and line printer, and a job enquiry terminal. Users may submit their jobs from any of these sites, as well as at the Computer Centre itself.

Jobs punching, plotting or using the PE-ASCII upper/lower case printer should use one of the above delivery codes as the first two characters of the jobname, so that their punched, plotted or printed outputs will be delivered automatically to them at an appropriate RIOS. INTERCOM users should note that files processed by the BATCH command are automatically prefixed by 'I', but that IX1, IX2, IX3 and so on are also interpreted as delivery codes.

DATA LINKS

The Computer Center provides for data links connections between the 7600 and experiments, via the FOCUS System which runs in one of a pair of CDC 3000 computers. CAMAC modules and mini-computers software at the experiment end of the link are available for certain computers. The CERN Network, which is under active development, is foreseen to replace the FOCUS System in 1978. For advice concerning new or existing data links, contact Ian McLaren (tel 5010) or the Programming Enquiry Office (tel 4952, 8-935).

MAGNETIC TAPE ALLOCATION, ROUTING AND SERVICES

All tapes are allocated by the computer receptionist and the following procedure should be followed-

WHO, WHEN AND WHERE

A) A stores user's card should be obtained from your division. This constitutes the authority to debit the appropriate budget account number - if in doubt ask your group leader which card/account to use.

B) Take your card to the Computer Centre receptionist in the Computer Centre user area - building 513- and ask for the quantity of tapes you require. The receptionist will need to know if your tapes are for use outside the Computer Centre.

C) The receptionist will then give you the Volume Serial Numbers or VSNs of your tapes. This will form part of the ANSI tape label if your tape is labelled at CERN. It is also the same as the VID or Visual IDentifier, used to locate the physical tape reel in the tape store. Note, however, that if you are registering a tape brought from outside which is already labelled, its VSN is already defined and the number you are given will be the VID only. See the TAPEIN/TAPEOUT control card for details of how to read and write such external tapes.

D) Your tapes may need preparation, and might not be immediately available for you to use or take away.

A receptionist is on duty from 8.30 to 17.00 Monday to Friday. For emergency allocation of a few tapes outside of these hours see the computer room supervisor. Standard tapes are half an inch wide, 2400 feet long with one load point (reflective beginning-of-tape marker). Unless requested otherwise they will be prelabelled with a standard ANSI label recording the division and group of the owner of the tape. By special request to the receptionist, shorter tapes can be allocated. The tapes are bulk-purchased and sample-tested by DD division. All tapes should be kept in the Computer Centre tape store whenever possible, to ensure optimum storage conditions.

Whenever you need to send a tape to or from the Computer Centre, you must obtain a TAPE ROUTING SLIP. This is a yellow card obtainable from the computer reception, RIOS or delivery point. With it, you can request-

-Delivery to you before or after processing.
-Delivery to the Computer Centre for storage.
-Cleaning. (Data on the tape is preserved)
-Degaussing. (All data and labels on the tape are erased)
-Pre-labelling. (Needed before using unlabelled tape as labelled)
-De-labelling. (Needed before using labelled tape as unlabelled)
-Re-numbering.

Pre- and de- labelling destroy existing data on a tape.

1.5 THE ARCHITECTURE OF THE CDC 7600

The CDC 7600 is a very fast computer executing typically 10 million instructions per second (Mips) with peaks of 27 millions in certain highly coded loops. The user program on which the central processing unit (CPU) is busy at any one moment is fully (except for a possible overlay structure) resident in the Small Core Memory (SCM) of 65536 words, of which about 59000 words are available to the user program(s). As long as the user program does not require any input/output this works fine with a basic machine cycle of 27.5 ns and a memory cycle time of 275 ns but arranged in 32 independant banks.

Input/Output for a user program takes place uniquely to and from the large disk files (7638 and 844) on the 7600. These disk files have extremely high transfer rates (3.5 Mcharacters/second) but are very slow in positioning the arms carrying the heads (107.5 ms average positioning plus rotation delay). The effect of this is that one must transfer a lot of data each time one has a request for a read or a write in order to achieve the high average data rates necessary for a 10 Mips Computer. Hence transfers of data to and from disk must use very large buffers in order to achieve overall satisfactory performance (5120 words). The CDC 7600 uses the Large Core Memory (LCM) of 512000 words for this purpose - user programs resident in SCM get their I/O buffers in LCM. The data is then moved between LCM and the user program in SCM in rather small blocks at very high speed in a memory-to-memory transfer - 360 Mcharacter/second. If a user program in SCM does not find its input data already available in the LCM buffers it is itself swapped out to LCM (in about 1ms for an average program) and, whilst the I/O operation is performed, another user program is brought into SCM. A further level of multi-programming is achieved by moving user programs out to the disks when waiting for operator action or whilst a stage operation is in progress.

Thus LCM becomes a very precious resource, critical to input/output and effective multiprogramming since the system overlays are kept there. Users programs can request that arrays be put in LCM if they do not fit within SCM. (at CERN this is rather common!)

The transfer of files between the CDC 6000 front-ends and the 7638 disk files is handled by the operating system in a spooling fashion (using large LCM I/O buffers!) in parallel with the execution of user programs. (The speed over a 6000-7600 link is about 600 Kcharacters/second at peak load).

HOW TO SET UP A JOB FOR THE 7600

This Chapter is intended as a very first approach to the 7600/6000 Computer System, for any user who wishes to execute simple tasks on the 7600 at CERN. A few examples of control cards display the way of

-Running a FORTRAN program
-Reading magnetic tapes
-Using permanent files

Some information about the job enquiry and the way of getting one's output, are also given, but one should refer to the following Chapters which go into more details about every control card described here.

CONTROL CARDS SUGGESTED TO EXECUTE A FORTRAN PROGRAM.

2.1 CONTROL CARDS SUGGESTED TO EXECUTE A FORTRAN PROGRAM.

The following cards are recommended to compile and execute a FORTRAN program. It may call routines in the CERN Program Library, read data cards and also print results. A short description of the function of each of the control cards is given-

```
abcdefg.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
FTN,L.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
      PROGRAM mine(INPUT,OUTPUT)
.
      READ  10,X
      PRINT 20,Y
.
      END
end-of-section
.
. Data cards, if any
.
end-of-information
```

JOB card

The first card of the deck is known as the JOB card. The first 5 characters of 'abcdefg' are the only ones used by the system, which appends 2 characters - XY, say - to these first 5 characters forming a unique system job name, 'abcdeXY'. This is normally printed on the banner page of the job's output. However, if the job attempts to print more than is allowed at a RIOS, it is printed centrally and the banner page displays the RIOS delivery code, followed by 'abcXY'. Note also that punching M9 or M6 in columns 79-80 of the job card allows the reading of decks punched in a mixture of 026 and 029 codes through a RIOS card reader. Also, the default time limit of 10B seconds can be overriden with the Tn parameter, where n is a user-requested time limit in octal seconds.

ACCOUNT card

This card identifies registered users. For details see the summary of control cards, Chapter 3.

CERN Program Library routines

Most of the CERN library routines are contained in their binary form on a single permanent file called 7600LIBRARY. This file must be

CONTROL CARDS SUGGESTED TO EXECUTE A FORTRAN PROGRAM.

attached to the job before any of the routines can be loaded- FIND makes it available. A list of the contents of 7600LIBRARY can be found in Chapter 9, or in the Program Library Manual, or on RIOS noticeboards. For more details of the library routines see the Chapter on FORTRAN programs.

FIND card

FIND makes permanent files accessible to the user's job- here two files, 7600LIBRARY (containing the CERN Program Library) and MAN, (containing the routines needed by the MANTRAP post-mortem processor) are made accessible to the user job with FIND. Of course, ATTACH could be used to copy the CCP 6000 disk file onto the 7600 disks (7638 or 844), but for heavily used files this is wasteful. FIND looks first to see if the file is already on the 7600 disks.

LIBRARY card

This card declares that the permanent file 7600LIBRARY attached to the job by FIND under the logical file name ,lib, as well as the post-mortem processor MAN, form the global library set. The loader will try to satisfy all the user-program unsatisfied external references from this set. The files of the global set are searched for unsatisfied external references in the order declared on the LIBRARY card- the global set is NOT searched circularly. For more details of the declaration of libraries, and of how to create your own private libraries, see the Chapter on control cards and the section on LIBEDT.

FTN card

FTN is the FORTRAN Extended compiler. This call to FTN will compile the FORTRAN cards found after the first end-of-section card up to the next end-of-section card. The binary compiled version of the program is written to a file called LGO. The parameter L on the FTN cards requests a source code listing on file OUTPUT. For more details of the FTN card parameters, see the section on FORTRAN programs or the summary of control cards Chapter.

LDSET card

This card is used to instruct the Loader to write the Loading Map of the job to the file ZZZZMP. If the user program aborts, then the post-mortem processor MANTRAP is called automatically (this is why MAN is declared in the LIBRARY card) and traces back from the error with the aid of this Loading Map and the FTN Symbolic Reference Map data. The printout from MANTRAP is in readable English, and gives the user the values of local variables and SUBROUTINE arguments in his trace-back chain for all routines he has compiled.

CONTROL CARDS SUGGESTED TO EXECUTE A FORTRAN PROGRAM.

Note that the operation of MANTRAP is independent of any user request for Symbolic Reference Map, listings (FTN card) or Loading Map (MAP card- see Chapter on control cards).

MANTRAP is described in more detail in Chapter 9, but its output is self-explanatory in the great majority of cases.

LGO CARD

This card loads and executes the contents of the file LGO (the compiled version of the FORTRAN program).

End-of-section card

The end-of-section card consists of the numbers 7,8 and 9 multipunched in column 1 of the card. These end-of-section cards are used to separate the input card deck into the job control cards, FORTRAN deck, data cards, and so on. Normally every control card such as FTN which tries to read data cards from the input stream will have a corresponding section in the input card deck.

FORTRAN program

The user FORTRAN program can call routines in the 7600LIBRARY. Users who are accustomed to other machines should read the Chapter 4 on FORTRAN programs for non-standard features of CDC Fortran.

Data cards

When the FORTRAN routines are executed and data is to be read from cards (the file INPUT) these cards are read.

End-of-information card

The end-of-information cards are prepunched and can be found near each RIOS card reader. They are white cards with a purple stripe at the top of each card and have the numbers 6,7,8 and 9 multipunched in column 1 . This card signals to the system that an entire job has now been read.

Output from the job

The printout from the user job above starts with a banner page, displaying the system jobname. This is followed by the FTN compilation listings and the user printing (if he does any). Always, the last item is the job dayfile. This dayfile contains information about the processing of the job by the system, and possibly diagnostic messages. All the control cards processed are listed, and if a file is staged - (read from the 6000) the number of words transferred is indicated.

2.2 JOBS USING MAGNETIC TAPES OR PERMANENT FILES

The use of magnetic tapes is controlled by using the TAPEIN and TAPEOUT cards (or possibly FILE, LABEL, VSN, FIND and STAGE cards). The parameters for these cards are described in the Chapter on control cards. For more details on tape processing, see the Chapter on tapes. The following control cards can be used to execute a FORTRAN program which reads data from a 7-track (MT) unlabelled (UL) tape written on a small experimental data-taking computer (RT=U). The TP1 parameter on the job card announces to the system the intention of the job to use a tape-

```
abcde,TP1.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
FTN.
TAPEIN,TAPE1,MT,UL,RT=U,VSN=12345A.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
.
. FORTRAN program
.
end-of-information
```

Permanent files are created by use of the CATALOG control card, and subsequently used via the ATTACH or FIND cards. In some cases (if the file is to be used both under INTERCOM and on the 7600) a FILE card may be necessary. The parameters of these cards are discussed in the Chapters on control cards and permanent files. In the example below, TAPE3 is created and CATALOGed on the CCP 6000 (ST=CCP) for use in a subsequent job- Its retention period is 120 days (RP=120)

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
FTN.
LDSET,MAP=B/ZZZZMP.
LGO.
CATALOG,TAPE3,myfile,ID=userid,ST=CCP,RP=120.
end-of-section
.
. FORTRAN program
.
end-of-information
```

JOBS USING MAGNETIC TAPES OR PERMANENT FILES

This permanent file, myfile, ID=userid, is ATTACHed and read by the job shown below-

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
FTN.
ATTACH,TAPE1,myfile,ID=userid,ST=CCP.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
.
. FORTRAN program
.
end-of-information
```

2.3 JOB ENQUIRY AND JOB STATUS

Two commands are available to users to enquire about the status of their jobs from an INTERCOM terminal.

The **Q command** is a means of printing or displaying complete queues on an INTERCOM terminal. The full form of the command is-

Q,x,yyy,OURS

Here x is one of I,O,P,E or A. These correspond to requests for information on INPUT, OUTPUT, PUNCH, EXECUTE and ALL queues. If A (ALL) is used, a list of jobnames in the mentioned queues is given. If none of the possible x is specified, the number of jobs in each of the possible queues is given.

A mainframe identifier may be given by yyy. If it is omitted, the host mainframe on which the Q command is being given is assumed. The current identifiers can be displayed by the command 'Q,ID'. Each mainframe can have more than one identifier but users should use for the RIOS front-end, MFA, and for the SUPERMUX front-end, MFB. The 7600 is known as MFZ. MFA,MFB, and MFZ are the physical identifiers of the mainframes.

The presence of 'OURS' restricts the output of queue information to jobs initiated from the host mainframe on which the user is enquiring.

For example, an enquiry to list all executing jobs on the 7600 submitted via the MFB 6000 could be entered on a SUPERMUX terminal as follows-

Q,E,MFZ,OURS

The **JOBS command** has the form

JOBS,abcdefg

This command is used to enquire about a specific job, 'abcdeXY' say. The queues are searched for all jobs whose system job names begin with the string 'abcdeXY'. Up to 7 characters may be given. Note that XY is added to the user jobname first 5 characters by the system. The answers shows, the LOC ation ,the NAME , the STATUS , the SIZE and the DEST ination of the job. The location gives the place of the job in the machine it is in (MFA, MFB or MFZ). For example, A (65) means position 65 in MFA. If the user submitted his job at a RIOS, the system job name 'abcdeXY' is printed on the RIOS teletype after his job has been successfully read. Users submitting their jobs centrally must enter a 'JOBS,abcde' enquiry and decide which of the

listed jobs 'abcdeXY' (there may be more than one) is the one they are interested in. The system job name 'abcdeXY' is not listed on any user accessible device when a job is read in centrally.

Status explanations for MFA and MFB 6000 jobs.

W-SWAP	Waiting to be swapped.
EXECUTNG	Executing at a control point.
W-SCHED	Waiting for scheduler activity.
W-MEMORY	Waiting for central memory.
W-P.F.	Waiting for permanent file.
W-TAPE	Waiting for magnetic tape.
W-DEVICE	Waiting for a device, such as a tape unit.
W-OPR	Waiting for operator action.
W-INTRCM	Waiting for INTERCOM user action.

Status explanations for 7600 jobs.

EXECUTNG	Executing
W.CPUSCM	Job in SCM waiting for CPU to become available.
W.CPULCM	Job swapped to LCM waiting to be swapped into SCM.
W.CPURMS	Job rolled out to disk waiting for swap into LCM.
W-P.F.	Waiting for permanent file.
W.RECALL	I/O recall (rolled out to disk waiting for staging of 6000 permanent files or tapes).
W-TAPE	Waiting for magnetic tape.
W.MEMORY	Waiting for more SCM/LCM.
W.EVENT	Waiting for an event (e.g. 7600 permanent file accessed by another job without multiread access). This situation is abnormal if it lasts.
SUSPNDED	Job execution suspended by the console operator. Ask the communications operator why- tel 4927.

Requesting the dropping of a job

It sometimes happens that users realise that a job they have submitted is going to fail, or have some undesired result. If they want to drop the job, they should contact the communication operator (tel 4927) and give him the LOCation identifier of the job, as well as the job name. The location identifier is the entry under 'LOC' obtained in response to a JOBS enquiry. The reason for requesting this information is firstly to ensure that the correct job is dropped, and secondly so that the operators do not have to look for the job, perhaps on all three mainframes.

INTRODUCTION

This Chapter attempts to describe the most commonly used 7600 control cards. A brief description of the function of the control card is given, together with its most useful options or parameters. For more detailed information, users should consult the SCOPE 2.1.2 Reference Manual.

ACCOUNT CARD

The ACCOUNT card identifies a user to the system, allowing the rejection of jobs submitted with exhausted budgets or by un-registered users.

ACCOUNT,name,group,accno.

name — This is the user registered name (up to 7 characters).

group — This is the group or division code. There are different specifications for this parameter. Note that the user registered name must correlate with the group or division code, or the job will be aborted.

accno — A six character alphanumeric string of the form ABBBCC, where A and CC are normally at the user disposal. However, BBB must represent a valid group, or a valid project within the overall group, or the job will abort. Phone up your Divisional Representative - see list in Chapter 8 - in case of any doubt.

ATTACH CARD

The ATTACH card is used to make permanent files available to a user job. The files can be resident on the 7600 disks, or either of the 6000 front-end machines. Permanent files are heavily used for storage of programs, in source or binary form, and also for data storage. For more details , see Chapter on permanent files.

ATTACH,lfn,pfn,ID=userid,options.

lfn — 1-7 alphanumeric character logical file name. The first character must alphabetic.

pfn — 1-40 alphanumeric character permanent file name.

ID=userid — 1-9 alphanumeric character owner identifier. See Chapter 8 for the rules for the format.

Options.

PW=list Password list of passwords PW=A,B,...,X established on initial catalog (see below). Only relevant to allow a PURGE function on a file initially cataloged with passwords, or in order to access a file which has had a READ password established on initial catalog.

CY=cy Cycle number.

ST=st Station parameter. If absent, a 7600 file will be ATTACHed. ST=CCP requests the ATTACH of a CCP 6000 file, ST=CCQ the ATTACH of a CCQ 6000 file.

LC=1 attaches the lowest existing cycle

For more details see the Chapter 8 on permanent files.

BUFFERS CARD

The BUFFERS control card will produce a dayfile output giving the number of unreleased I/O buffers, specified by file name, for non-returned files. (Note that one LCM buffer is about 1000 words).

BUFFERS.

Files can be returned either by use of the RETURN control card or by a FORTRAN call to the routine RETRNF (CERN Program Library K510).

CATALOG CARD

The CATALOG card is used to make a user file into a permanent file. Any file created by a user job can be CATALOGed. The 6000 disks have a limited capacity, and space is allocated to users on a group or divisional basis - See the Chapter on permanent files. Really large files can be CATALOGed on the 7600 disks, but they are temporary and their lifetime unpredictable. The shorter the file is and the more heavily it is used, the longer it will last. A 7600 job must use a REQUEST card to gain permission for a CATALOG on the 7600 disks - (REQUEST,lfn,*PF.).

CATALOG,lfn,pfn,ID=userid,options.

The options associated with the CATALOG control card are the same as for the ATTACH card, except for -

RP=rp Retention period of file in days. Default is 14 days. The maximum is 999 days (infinite). This parameter has no meaning for 7600 disk files.

PW=list Password list on other than an initial catalog. Written in the form PW=A,B,C,..,X where A,B,C,..X are parameters established at initial catalog. This is only relevant to catalog a new cycle of a file established with passwords. See the section on passwords for details, if necessary.

For more details see the Chapter 8 on permanent files.

DISPOSE CARD

The DISPOSE card is used to route user files to particular devices. A DISPOSE card with a null option can be used to prevent the output processing of such files as PUNCH, PLOT and so on.

DISPOSE,lfn,option.

lfn	Logical file name of file to be sent to a device.
Option	This indicates the device to which the file is to be sent. All devices are at the Computer Centre except the RIOS printers.

Option	Device
PR	Any printer at the location from which the job originated
PR=CUP,ST=CCP	User area printer, building 513
PR,ST=CCPIrr	RIOS printer, rr is the RIOS identifier - see list in Chapter 1 - for example OC, TC, NP and so on
PU,ST=CCP	Card punch, 029 code
P9,ST=CCP	Card punch, 029 code
P6,ST=CCP	Card punch, 026 code
P8,ST=CCP	Card punch, for FREE FORM BINARY DATA only
PB,ST=CCP	Card punch, binary cards
PT,ST=CCP	Any available plotter
PT=CS1,ST=CCP	Plotter, squared paper
PT=CP1,ST=CCP	Plotter, plain paper

Files named PUNCH, PLOT and OUTPUT are all automatically disposed to the correct devices at the location from which the job was submitted. Since RIOS stations do not have punches and plotters, a job submitted from a RIOS must include a DISPOSE card with ST=CCP for the relevent file. For example -

DISPOSE,PUNCH,*PU,ST=CCP.

To eliminate a file such as PUNCH or PLOT from a job, use -

DISPOSE,lfn.

This ensures that lfn will not be processed by the system. RETURN, lfn. is not sufficient. The file will still be sent to the appropriate device. REWIND,OUTPUT. eliminates the file OUTPUT.

All device codes can be preceded by *, for example-

```
DISPOSE,OUTPUT,*PR,ST=CCPIOC.
```

This * implies that the DISPOSE is to be made at the end of the job, and not at the point where the DISPOSE card is processed.

DMP CARD

Users can at any time request a dump of any part of their SCM field length. This is sometimes useful for error-finding, but MANTRAP, the FORTRAN post-mortem processor, is much more efficient at scanning the user's field length in the event of a failure of his job. (See Chapter 9). If you want to ask for a dump, however, the format is-

```
DMP,word1,word2.
```

Limits of the core dump are 'word1' (lower) and 'word2' (higher). These two numbers must be octal. Note that an error message appears when 'word1' is beyond the user's SCM field or if 'word2' is smaller than 'word1'.

EXIT CARD

A user can almost always regain control over his program if it aborts during execution, by using the error recovery procedures described in Chapter 11. He can then decide within his own program what he wishes to do. However, this is not always convenient or necessary. The EXIT control card allows a simple way to control the execution of the user's control cards if his job aborts. The two common forms of EXIT are -

```
EXIT.          basic form.
EXIT,U.        unconditional processing option.
```

If no errors occur, the job terminates at 'EXIT', but continues through an 'EXIT,U'. If errors occur, control transfers to the control cards following either 'EXIT' or 'EXIT,U'. (If a job is dropped by an operator, EXIT cards are treated in the same way as for any other job abort).

Two other forms of EXIT exist- EXIT,C and EXIT,S. For details, see the SCOPE 2.1.2 Reference Manual.

FILE CARD

The FILE card informs the system about the logical structure of data in a file. Most users of the 7600 never need FILE cards, since they use the system defaults (RT=W, BT=I for tapes and RT=W unblocked for disk files). However, FILE cards are necessary when using files created on the 6000. The form of the FILE card is -

FILE,lfn,options.

Once again a considerable number of options exist for the FILE card describing the file lfn . The majority are concerned with the record type, RT=rt, and the block type, BT=bt of the file concerned. The more common values are given below. For more details, see Chapter 6 on record types.

Record type	Description
RT=W	Control word records
RT=U	Undefined records - 1 block= 1 logical record
RT=S	SCOPE 3.4.3 standard records, 6000 machines
RT=Z	Zero byte terminated records - INTERCOM files
RT=X	Short block terminated records - CERNSCOPE tapes
RT=F	Fixed length records
FL=fl	Length of record in decimal characters, F and Z records

Block type	Description
BT=I	Internally blocked files
BT=C	Character count blocked files
BT=K	Record count blocked files
RB=rb	Decimal number of blocks per record for K-format files. The default is RB=1
MRL=mrl	Maximum LOGICAL Record Length in characters mrl is a decimal number. The default is 5120
MBL=mbl	Maximum Block Length in characters mrb is a decimal number. The default is 5120

MRL and MBL particularly apply to RT=U, F and S. See Chapter 6 on record types.

These record and block types occur in the vast majority of cases in the following combinations, and are the DEFAULTS chosen by the system as indicated.

File type	Use of file
RT=W,BT=I	Tape files on the 7600, default FORTRAN unformatted READ/WRITE, default, SCOPE 3.4.3
RT=W	Disk files on the 7600, default, SCOPE 2.1.2
RT=S,BT=C	Utility and FORTRAN BUFFER default, SCOPE 3.4.3
RT=Z,BT=C,FL=80	INTERCOM files, default Formatted FORTRAN READ/WRITE on 6000
RT=U,BT=K	Tape files from data-taking computers
RT=F,BT=K,FL=80	Card image files on tape

The user can also specify on the FILE card what he wants to do if an unrecoverable parity error occurs when the file concerned is being processed. The possible options are as follows, and are entered as another FILE card parameter.

EO=T	Terminate job . Default for the COPY control cards.
EO=D	Records with parity errors dropped.
EO=A	Records with parity errors accepted, FORTRAN default.

For both EO=D and EO=A the parity error flag is set when an attempt is made to read the bad data. The only difference between them is that with EO=D the bad data is not transfered to users input buffer while for EO=A it is and the user can, if he wishes try to make some sense of it.

If TD, DD or AD is used instead of T, D or A a listing of the record in error is produced and printed separately at the end of job.

It is possible to deal with code conversion involving BCD, EBCDIC or ASCII codes. By default, both 7- and 9-track tapes are written in binary mode with odd parity. To obtain code conversions, add the following parameters to the FILE and STAGE cards.

FILE card	STAGE card	Data mode
CM=YES	nothing	7-track tape, BCD, even parity
CM=YES	nothing	9-track tape, EBCDIC, odd parity
CM=YES	EB	9-track tape, EBCDIC, odd parity
CM=YES	US	9-track tape, ASCII, odd parity

FIND CARD

The FIND card is very similar in its function to ATTACH. It is used to try to reduce system activity for heavily used 6000 permanent files. It will first of all look for the user's file on the 7600 disks, and only transfer or stage it from the 6000 disks if it is not found on the 7600 disk. Each time the 6000 file is transferred, it will be CATALOGed on the 7600 disks and (hopefully) will not have to be transferred if a FIND is attempted in a job run shortly afterwards. Users should, for example, always, FIND the 7600LIBRARY.

FIND,lfn,pfn,ID=userid,options.

lfn	Logical file name 1-7 alphanumeric characters.
pfn	Permanent file name 1-40 alphanumeric characters.
ID=userid	Owner identifier 1-9 alphanumeric characters.
LV=1	Default parameter value, meaning that if a 7600 permanent file is not found, a copy of the 6000 permanent file with the same permanent file name is staged and catalogued as a 7600 permanent file.
LV=2	If a 7600 permanent file is not found, the magnetic tape described in the STAGE card which follows the FIND card, is staged from the 6000 and catalogued as a 7600 permanent file with permanent file name and owner identifier as supplied on the FIND card.
CY=cy	If omitted, the 7600 file with the highest cycle number is attached. If no file is found on the 7600, the CCP 6000 permanent file with the highest cycle number or the magnetic tape (depending on the LV parameter) is staged and catalogued as CY=511 on the 7600. If cy is specified, cycle cy of the 7600 permanent file is attached. If this cycle is not present, the magnetic tape or cycle cy of the CCP 6000 permanent file, is catalogued as cycle cy of the 7600 permanent file - (unless ST=CCQ is specified, giving CCQ file).
PU=YES	FIND purges the 7600 permanent file if found and then catalogs the 6000 permanent file or magnetic tape as a 7600 permanent file, depending on the level parameter.
PU=ONLY	FIND purges the 7600 permanent file, if found.
ST=CCQ	By default, FIND searches the CCP 6000 filebase. Use this to search the CCQ 6000 filebase instead.

FTN CARD

The FTN card is used to invoke to the FORTRAN Extended Compiler. This is the most heavily used compiler at CERN, and this means that not only must it generate efficient code for execution, but also that it must be capable of efficiently running the great number of short jobs executed during the day. Improved versions of the compiler are normally submitted as soon as possible, with the result that this list of parameters and recommendations may not be correct for ever.

It is, in fact, expected that the present default compiler, FTN 4.1, will be replaced by the improved FTN 4.6, before this guide is re-issued. Check the Computer Newsletter for notification of the change.

The principal advantages of FTN 4.6 are -

A) FTN 4.6 contains as an overlay, a fast one-pass compiler, the so-called Time-Sharing (TS) compiler. It saves CPU time as well as real time in compilation and the execution times are comparable with FTN,OPT=n.

B) FTN,OPT=2 Version 4.6 produces even more optimized code than Version 4.1.

C) FTN 4.6 accepts 116-character strings on input (for PATCHY users). These strings are fully printed by FTN or FTN,OPT=1., whereas the TS version truncates the input string.

Users are strongly recommended to try this new compiler. See the Section below for details of how to use it.

FTN 4.1 control card options

Exit parameter

A	Compilation terminates and branches to an EXIT,S control card if fatal errors occur during compilation. If there is no EXIT,S. card the job terminates. By default, a job continues after compilation errors.

Binary object file- default is LGO

B=lfn	Generated object code file is lfn.

Source Input- default is INPUT

I	Source code is on file COMPILE.
I=lfn	Source code is on file lfn.

Listing controls- default is NO list

L	List source code and diagnostics on OUTPUT.
L=lfn	List source code and diagnostics on file lfn.
L,R=1	List source code, diagnostics and reference map (symbols, addresses and properties) on OUTPUT. Exactly the same as the obsolete LR.
L,R=2	As R=1 plus DO-Loop map and line number references.
L,R=3	As R=2 plus EQUIVALENCE groups and COMMON block members.
LX=lfn	List source code, non-ANSI usage and diagnostics on file lfn.
LO=lfn	List source code, generated object code and diagnostics on file lfn.
LN=lfn	List source code and fatal diagnostics on file lfn. Informative diagnostics suppressed.
LRX	List source code, reference map and non-ANSI usage on OUTPUT.

Print limit- default is PL=1000

PL=pl

Upper limit on the number, pl, of lines the user program is allowed to write to the file OUTPUT (not including compilation listings, etc.). It is also possible to control the OUTPUT file line limit by adding the PL parameter as the last option of the LGO or EXECUTE cards. The first parameter on the EXECUTE card is the program ENTRY point, and must not be overwritten, so the two forms are-

```
LGO,PL=pl.
EXECUTE,,PL=pl.
```

If file names are being changed with the LGO or EXECUTE cards, at execution time, the PL parameter must be the last to appear.

Rounded Arithmetic Switch

ROUND=r

r may be any combination of the FORTRAN arithmetic operators +,-,* and /. Single precision REAL and COMPLEX floating point arithmetic operations are performed using the hardware rounding features (7600 Computer Sytems Reference Manual). By default, arithmetic computations are not rounded.

Error traceback

T

Calls to basic external functions are made with call-by-name sequence. Full error traceback occurs if an error is detected, and maximum error checking takes place (such as invalid argument for TAN). By default, call-by-value linkages are generated and no checks are made for invalid arguments.

Debug

D — This requests the compiler to include debugging code in the user's generated object code. Directives are expected to be found on the file INPUT, unless D=lfn is used. See Chapter 10 for details of the use of Debug facilities.

Optimisation

OPT=0 — Fast compilation. No optimisation of object code. Use for program debugging.

OPT=1 — Default.

OPT=2 — Fast execution but slow compilation. Use for tested production jobs.

Program syntax checking mode

Q — Performs full syntax checking of the source code, but does not produce any object code. Gives a fast check for compilation errors, but program cannot be loaded.

Post- Mortem Processor MANTRAP

When using the FTN 4.1 compiler, the Post-Mortem processor MANTRAP which produces an interpretation of the FORTRAN dumps, (see Chapter 9 for details) can be requested without any special parameter in the FTN card. Note - Do not use the U parameter on the FTN control card.

```
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,man,MAN,ID=OPSYSTEM.
FIND,mylib,mylib,ID=ddgggxxxx.
LIBRARY,mylib,lib,man.
FTN.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
      FORTRAN program
end-of-information
```

FTN 4.6

Control card differences

The control card differences concern the listing parameters, the error traceback parameter, the use of the Post-Mortem Processor MANTRAP and the use of the TS version of the compiler. All other parameters described above, under FTN 4.1 remain valid.

Listing controls- default is NO list

L	List source code and diagnostics on OUTPUT.
L=lfn	List source code and diagnostics on file lfn.
L,R	List source code, diagnostics and reference map (symbols, addresses and properties) on OUTPUT.
L,R=1	Exactly the same as L,R.
L,R=2	As R=1 plus DO-Loop map and line number references.
L,R=3	As R=2 plus EQUIVALENCE groups, COMMON block members.
L,EL=A	List source code, non-ANSI usage and diagnostics on OUTPUT.
L,OL	List source code and generated object code on OUTPUT.

Error Traceback

The error traceback option described above for FTN Version 4.1 (see FTN control card options) comes automatically with FTN Version 4.6 without specifying the T parameter on the FTN control card.

Time-Sharing compiler

TS	Selects the Time-Sharing version of the compiler. This is recommended to users not wishing to use FORTRAN Debug facilities (D parameter) or the Post-Mortem Processor MANTRAP (which is incompatible with it). Warning- TS is not compatible with the D,OPT=n or Q parameters.

Post-Mortem Processor request

U	Selects the Optimize compiler and prepares the necessary information for use with the post-mortem processor. Users who are not familiar with the program they are using, or who are unfamiliar with tracing execution errors or reading dumps, or who are having any problems with their programs ARE STRONGLY ADVISED TO USE THE POST-MORTEM PROCESSOR MANTRAP. See Chapter 9 for more details, or any of the control card examples of jobs in this User's Guide.

Access to the compiler

The compiler is stored in a Library known as NEWPRODUCTS, ID=OPSYSTEM. To get access to it, users must insert the control cards

```
FIND,new,NEWPRODUCTS,ID=OPSYSTEM.
LIBRARY,new.        or   LIBRARY,mylib,....,new.
```

before the FTN control card.

A new version of the Post-Mortem analyzer MANTRAP, is also necessary. This is stored in the Library file NEWMAN,ID=OPSYSTEM. Note also that the U parameter must be specified on the FTN control card in order for MANTRAP to work correctly.

Example 1

Use the TS option to obtain fast compilation, produce a listing and cross reference map -

```
FIND,new,NEWPRODUCTS,ID=OPSYSTEM.
LIBRARY,new.
FTN,L,R,TS.
FIND,mylib,mylib,ID=ddgggxxxx.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,mylib,lib,new.
LGO.
end-of-section
    FORTRAN program
end-of-information
```

Example 2

Produce highly optimized code using OPT=2, use MANTRAP and produce a listing -

```
FIND,new,NEWPRODUCTS,ID=OPSYSTEM.
LIBRARY,new.
FTN,L,U,OPT=2.
FIND,mylib,mylib,ID=ddgggxxxx.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,man,NEWMAN,ID=OPSYSTEM.
LIBRARY,mylib,man,lib,new.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
    FORTRAN program
end-of-information
```

Note that the Library NEWMAN is only necessary at execution time not at compilation time. If neither own libraries nor 7600 LIBRARY are needed, the LIBRARY control statement before execution is -

```
LIBRARY,man,new.
```

JOB CARD

The JOB card is used to tell the system the user's jobname, and also what time limit he wants, whether it is a tape job, whether he wants to pay for priority time, whether he wants to run on the 7000 machine or on one of the 6000 machines, and so on. It is also used to indicate a mixed code (026/029) card deck by punching M6 or M9 in cols. 79-80,but this only works for jobs read through a RIOS.

Users who want to run on the 6000 should add the CP60 (or STCCP for the CCP machine, or STCCQ for the CCQ machine) parameter to their job card, and also note that if they want n 9-track tape units and m

7-track tape units then the parameters NTn, MTm are obligatory. Note however, that for the 6000 machines, STCCP and STMFA are equivalent, and so are STCCQ and STMFB. STMFZ is default and causes the job to be run on the 7600 machine.

abcdefg,TPn,Tn,Pn,Rn,Dnn. Code,cols 79-80

abcdefg 1-7 alphanumeric character job name. Must start with a letter. Only the 5 first are used by the System. Optional parameters are as follows -

TPn A CERN parameter for 7600 jobs. This parameter distinguishes tape jobs from non-tape jobs at job scheduling time. TP0 is default and indicates a non-tape job. TP1 indicates a tape job. Whenever a TP0 job attempts to stage an input tape, it is rerun as a TP1 job unless the job was submitted with the R0 parameter on the job card in which case the job is aborted. The automatic rerun facility is intended to help users of FIND with LV=2 who may not know if their tapes are cataloged on the 7600 disks or not.

Tn n is a 1-5 digit octal CPU time limit in seconds . The default is 10B seconds. Jobs are assigned to classes on the basis of their time request and their TP parameters. This class is given by the response to the JOBS enquiry. The definition of the classes is as follows-

Type	TP0 class	TP1 class	Time
express	0	2	up to 10B
short	1	2	11B to 40B
medium	3	4	41B to 300B
long	5	6	301B to 2000B
very long	5	6	over 2000B
SPY jobs	7	7	anything

Pn A priority code. Users can request an allocation of priority time from their Divisional representative (see list in Chapter 1) or the Computer Coordinator. The types of priority available and their cost proportional to a normal non-priority batch job are as follows-
P3 -(X 3.0) Very high priority for running experiments
P2 -(X 2.0) High priority for urgent short or medium jobs
P1 -(X 1.3) High overnight priority for scheduled production jobs
P0 -(X 0.5) A job class for low priority jobs which will only be run at times when the machine is idle, or during requested block time.

Rn The R parameter on the JOB card allows the user to specify the number of times, n, his job can be rerun under any conditions. The default is 77B, so if the user wishes to inhibit rerun, he should specify R0.

Dnn Dependency specification. This allows a user to run a set of jobs in a predetermined sequence. See the SCOPE 2.1.2 Reference Manual. Note, however, that dependency flags are lost if the 7600 goes down.

code This parameter punched in columns 79,80 of the job card specifies the card punch code of the job deck. If it is omitted or 29 it is assumed that the deck is in 029 ANSI code. If it is 26 it is assumed to be in 026 code. At any RIOS, the codes M6 and M9 can be used, and specify a mixed-code 026-029 card deck. Any non-FORTRAN characters are assumed to be in 026 or 029 code respectively.

LABEL CARD

The LABEL card is used to give details of a tape's ANSI label, if any. The label can be checked or re-written. Any error in label checking will abort the staging of the file.

LABEL,lfn,options.

lfn Logical file name.
R Read (check) label.
W Write label.

L=label 1-17 character file identifier. Identifiers containing special characters must be enclosed by $ signs.

O=ddgggxxxx Owner identifier. See below section on TAPEOUT card.

T=dddd retention period in days. If T is not given, the retention date is set to infinite.

The parameters R and W take precedence over the E (Existing label) or N (New label to be written) parameters of the STAGE card, if they are present.

See Chapter 7 for special considerations for the L and T fields when moving tapes between the CDC and IBM.

LDSET CARD

The LDSET card is used to give loader commands. One common use is to direct the writing of the Loading Map to ZZZZMP by MANTRAP post-mortem processor users (LDSET=B/ZZZZMP). Another common use is to allow more than five user libraries to be searched. The LIBRARY

card can name up to five user libraries but up to five more can be declared as a local library set on an LDSET card. This LDSET card must occur within a loading sequence, and the local library set is only available for this loading sequence. For example-

```
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
FTN.
LIBRARY,lib1,lib2,lib3,lib4,lib5.
LDSET,LIB=lib6/lib7/lib8/lib/MAN,MAP=B/ZZZZMP.
LGO.
```

The library files are searched non-circularly in the order given on the LDSET card in order to try and satisfy the user's unsatisfied externals. Note that if both a global library set (LIBRARY card) and a local library set (LDSET card) have been declared for a loading sequence, the global set is searched first to try to satisfy unsatisfied external references. The full list of LDSET options can be found in the Loader Reference Manual.

LGO CARD

The LGO card is used to initiate execution of the relocatable binary file which FTN produces. Of course, the user can rename the file LGO with the 'B=name' parameter of the FTN card, and he should then change LGO to 'name' in what follows. The user can change the logical file name (lfn) of one or more of the files to be processed by his program by using the LGO card. A new lfn can be entered as a positional parameter on the LGO card, renaming the file declared in that position in the user's FORTRAN PROGRAM card. For example-

```
PROGRAM mine(INPUT,OUTPUT,TAPE1,TAPE2)
```

allows the user to rename TAPE1 as TAPE11 as follows with the LGO

```
LGO,,,TAPE11.
```

The user can also set a new print limit on the LGO card, by giving the parameter PL=pl. If he is renaming files, this should be the last parameter given.

```
LGO,,,TAPE11,PL=5000.
```

LIBEDT CARD

The LIBEDT utility under SCOPE 2.1.2 allows the user to create his own library files from his LGO files. They are modified to allow fast scanning by the loader when it is trying to satisfy the user's

unsatisfied externals. To create a library file 'mylib', for example-

```
abcde.
ACCOUNT,name,group,accno.
FTN.
LIBEDT,M.
CATALOG,mylib,mylib,ID=userid,ST=CCP.
end-of-section
.
.  FORTRAN deck
.
end-of-section
LIBRARY(mylib,NEW=2000)
REWIND(LGO)
ADD(*,LGO)
FINISH.
LISTLIB(*,mylib)
end-of-information
```

The FORTRAN deck is compiled as LGO. The LIBEDT directives state first that the library is to be called 'mylib', reserving 2000 words for the library directory. The name 'mylib' can be up to 7 characters, and the 2000 word directory is normally adequate. However, up to 9000 words can be requested. The ADD directive requests the addition of all of LGO to mylib, and FINISH signals the end of the editing sequence. LISTLIB gives a short informative list of the routines in mylib. **Note that the parentheses are obligatory in LIBEDT directive syntax.** The 'M' parameter in the LIBEDT card allows LIBEDT to print informative messages about its operations.

To modify the file mylib in a subsequent LIBEDT operation, the user might run a job like the following . Here, two other users' LGO files (lgo1 and lgo2) are added to (or update) the library, and the user himself provides replacements on LGO of some of his original routines-

```
abcde.
ACCOUNT,name,group,accno.
FIND,mylib,mylib,ID=userid.
FIND,lgo1,lgo1,ID=userid1.
FIND,lgo2,lgo2,ID=userid2.
FTN.
LIBEDT,M.
CATALOG,newlib,newlib,ID=userid,ST=CCP.
end-of-section
.
. FORTRAN replacements of user
.
end-of-section
LIBRARY(newlib,NEW=2000)
OLDLIB(mylib)
REWIND(LGO)
REPLACE(*,LGO)
REPLACE(*,lgo1)
REPLACE(*,lgo2)
DELETE(badnam)
```

```
FINISH.
LISTLIB(*,newlib)
end-of-information
```

The OLDLIB directive states that the library file 'mylib' is to be modified, the LIBRARY directive has declared that the new library is called 'newlib'. The REPLACE directives add all new routines, deleting old routines (if any) with the same name. The DELETE directive deletes the named routine 'badnam'. Note that after many modifications of a library with LIBEDT, there is often a large amount of wasted space in the library file. This can be avoided by occasional use of COPYLB- see the SCOPE 2.1.2 Reference Manual.

LIBLOAD CARD

Users of private libraries who want to execute a main program 'main' on one of their libraries, while also compiling and loading some routines, must use the following control card sequence-

```
LIBRARY,mylib,lib,MAN.
FTN.
LDSET,MAP=B/ZZZZMP.
LOAD,LGO.
LIBLOAD,mylib,main.
EXECUTE,main,PL=5000.
```

The PL parameter, if required, must be placed on the EXECUTE card.

LIBRARY CARD

The LIBRARY card allows the user to specify up to 5 library-format files (see LIBEDT) as forming his global library set. Until a subsequent LIBRARY card redefines this set, the loader will try to satisfy the user's unsatisfied externals from this set. If the user has declared a local library set with the LDSET card, the local set is searched after the global set. Note, however, that the search of library sets is non-circular, so some care is needed over the order of declaration on the LIBRARY card. Normally the user's private libraries come first, and the 7600LIBRARY and MAN (MANTRAP) last. A library file can be declared more than once on the LIBRARY card if need be to avoid problems with ordering.

LOAD CARD

The LOAD control card specifies the files to be loaded, but not executed.

```
LOAD,lgo1,lgo2,...,lgon.
```

lgoi Name of the files to be loaded in the order specified on the card

The EXECUTE card completes the load if necessary and initiates execution. The two following cards -

```
LOAD,lfn.
EXECUTE.
```

are the equivalent of -

```
lfn.
```

REQUEST CARD

The REQUEST card is used to obtain permission to assign a file to a permanent file device belonging to the machine where the user's job is running. At present a 7600 job can CATALOG files on the 6000 without a REQUEST. The form of the REQUEST card is-

```
REQUEST,lfn,*PF.
```

RETURN CARD

The RETURN card removes a file from a user's job, releasing any disk space and buffers associated with it. RETURN will not prevent the post-staging of a file which has already been explicitly or implicitly marked for post-staging (to do this, use the UNSTAGE control card). After a RETURN of lfn, the user can re-use the name lfn in any way he wishes. No information about the previous uses of lfn is kept. In order to release LCM buffer space and disk space that is no longer needed - it is strongly recommended to RETURN files as soon as possible.

To RETURN a library file, it is first necessary to detach it from the declared library set by putting in a new 'LIBRARY.' Card before the RETURN.

```
LIBRARY.
```

gets rid of all currently declared libraries.

```
RETURN,lfn1,lfn2,....,lfnn.
```

lfni — Name of a logical file to be returned to the system (minimum of one lfn required). Files return to the system for system processing according to normal disposition.

REWIND CARD

The REWIND card is used to position files to their beginning-of-information. In copying operations, this is very often essential.

REWIND,lfn1,lfn2,....,lfnn.

lfni — Name of a logical file to be rewound (at least one required).

STAGE CARD

The STAGE card directs the reading and writing of tapes. It provides the system with information on the physical properties of the tape, and possibly on whether code conversion is required or whether only part of the data is wanted (on reading only). The STAGE card, and by implication the TAPEOUT card, makes no provision for extending a tape file. You may not skip files on a tape and then write to it.

STAGE,lfn,options.

Staging direction.

PRE — Default, meaning that tape is to be read only.
POST — Means that tape is to be written only.

Note — A tape cannot both be PRE and POST staged. The last parameter supplied takes precedence.

Tape specification, 7-track or 9-track.

NT — Default, tape is 9-track.
MT — Tape is 7-track.

Tape recording densities.

9-track	7-track	description
PE	-	1600 BPI, default for 9-track, phase encoded.
HD	HY	800 BPI, default for 7-track.
-	HI	556 BPI.
-	LO	200 BPI. For reading only.

Conversion codes.

It is possible to request code conversion involving BCD, EBCDIC and ASCII codes by means of parameters on the STAGE and possibly FILE cards. The possibilities are noted below.

FILE card	STAGE card	Data mode
CM=YES	nothing	7-track tape, BCD, even parity
CM=YES	nothing	9-track tape, EBCDIC, odd parity
CM=YES	EB	9-track tape, EBCDIC, odd parity
CM=YES	US	9-track tape, ASCII, odd parity

The standard character sets are displayed in Chapter 13.

Tape number or VSN specification.

This parameter specifies the Volume Serial Number of the requested tape. If your tape is a CERN labelled tape, this forms the VSN field of the ANSI tape label and is also the same as the VID or Visual IDentifier on the tape reel. If your tape has been labelled outside CERN, the VID and VSN of the tape will be different. See the Sections on TAPEIN/TAPEOUT and VSN control cards for details of how to read and write such tapes. In its simplest form, for requesting the single tape 12345A it appears as -

```
VSN=12345A
```

If the user wishes to have continuation tapes 12346B,12347C, mounted then the form is -

```
VSN=12345A/12346B/12347C
```

The user can also specify that he wants only certain partitions or certain blocks from a tape. To skip the first n partitions and read the next m partitions, the form of the parameter is -

```
VSN=$12345A/F/n/m$
```

Similarly, to skip the first n blocks and read the next m, the parameter appears as -

```
VSN=$12345A/B/n/m$
```

This is known as PARTIAL staging. If end-of-information is reached, staging is terminated for both file and block staging. If an end-of-file is reached, block staging is also terminated. Partial staging does NOT work for all record types . Check in the Chapter 7

for details.

Users who stage many tapes may find that their jobs exceed the overall default limit for the mass storage assignable to a single job, resulting in the message- 'JOB MASS STORAGE LIMIT EXCEEDED'. If this happens, they should add the following control card to their job control card deck, prior to their STAGE or TAPEIN cards, to grant the maximum allowed mass storage to the job

```
LIMIT,0.
```

Staging multi-volume files

If the user wishes, he may stage all the volumes of a multi-volume labelled file at once by specifying the parameter SF on his STAGE card.

Operator intervention and parity errors

If the user has the parameter 'NR' on his STAGE card, the operator is not informed of parity errors on tape reading. Bad data is processed according to the user's EO parameter on his FILE card. If no NR parameter is specified, the operator must intervene to permit the staging of the tape to continue, but if he does give a 'GO' the bad data is passed on 7600 without any indication of their being in parity error.

The NR parameter is ignored for 'closed shop tapes' (those with VSNs in the range 60000-69999), which offer lower error rates due to their more restricted use- only as 9-track, 1600 BPI labelled tapes kept permanently in the Computer Centre.

Labelling parameter

N	Indicates that a new label is to be written. This can only be done if the tape is already labelled or pre-labelled.
E	Indicates that a label already exists.
Omitted	Implies E if a LABEL card is present, or an unlabelled tape if no LABEL card exists.

TAPEIN/TAPEOUT CARD

The physical and logical properties of a tape are most conveniently specified by using the CERN control cards TAPEIN (reading) and TAPEOUT (writing), which generate the SCOPE 2.1.2 control cards FILE, VSN, LABEL and STAGE. TAPEIN also can generate calls to the FIND utility for the temporary storage of tape data on the 7600 disks. Most options apply both to TAPEIN and TAPEOUT - those that are different will be indicated. The corresponding SCOPE 2.1.2 control

card (FILE, VSN, LABEL, STAGE or FIND) will be indicated for each parameter.

Required parameters.

lfn	Logical file name.

VSN parameter. (VSN, STAGE).

VSN=vsn	Volume Serial Number. vsn is a 5-digit tape number, plus checkletter. (multiple VSNs cannot be specified with TAPEIN)
VSN=$vsn/F/n/m$ VSN=$vsn/B/n/m$	Partial staging. Skip n files (F) or blocks (B), and then read the next m files or blocks. A full description of partial staging appears in the Chapter 7. Example- VSN=$12345A/B/0/10$.

Optional parameters.

Note that * indicates a default parameter, which can be omitted.

VID parameter. (VSN).

VID=vid	The Visual IDentifier of the tape. This should be used when a labelled tape not written at CERN is to be read. The VSN parameter contains the VSN magnetically recorded in the tape label. The VID parameter gives the number allocated by the Tape Receptionist at CERN and it informs the operators which reel to mount.

Tape specification 9- or 7- track (STAGE).

NT *	Nine track, default.
MT	Seven track.

Recording densities (STAGE).

PE *	9-track, 1600 BPI, default for NT.
HD	9-track, 800 BPI.
HY *	7-track, 800 BPI, default for MT.
HI	7-track, 556 BPI.
LO	7-track, 200 BPI.

Label specifications (LABEL).

SL *	Standard label, default.
UL	Unlabelled.
L=label	Label contains 1-17 alphanumeric characters.
L=$label$	Label contains 1-17 alphanumeric and special characters. L=DSTJAN75 and L=$DST JAN 75$ are examples.

O=ddgggxxxx * Owner identifier, where dd and ggg are the owner's division and group code, and xxxx are up to 4 optional characters chosen by the user. If not given, the O parameter will be constructed from the division and group code of the ACCOUNT card followed by 4 blanks. See Chapter on magnetic tapes for details.

Record types - Block types - (FILE).

RT=W * Control word records, 7600 standard, default. Implies BT=I - internal blocking - for tape files.

RT=U Undefined records- 1 block= 1 logical record. Implies BT=K.

RT=X Short block terminated records- CERNSCOPE. Implies BT=C.

RT=S SCOPE 3.4.3 standard records, MFA and MFB 6000. Implies BT=C.

RT=Z Zero byte terminated records- INTERCOM files. Implies BT=C.

RT=F Fixed length records. Implies BT=K.

Record length for F and Z records (FILE).

FL=fl fl (decimal) characters per record. For example, FL=80. Default field length is 0. RT=F or Z and CM=YES parameter. It is 0 otherwise.

Blocking factor (FILE).

RB=rb For fixed length records, rb is the number of logical records per block. Default is RB=1 .

Maximum record length - Maximum block length (FILE).

MRL=mrl For C blocking, maximum record length, in decimal number of characters. Default is MRL=5120.

MBL=mbl For K blocking, maximum block length, in decimal number of characters. Default is MBL=5120.

Character code conversion (FILE, STAGE).

CM=YES 7-track tape, BCD conversion.
CM=YES 9-track tape, EBCDIC conversion.
CM=YES,US 9-track tape, ASCII conversion.

Error options for unrecoverable parity errors (FILE)- TAPEIN only.

EO=DD *	Drop bad record and list its contents- default.
EO=D	Drop bad record, don't list.
EO=T	Terminate job.
EO=TD	Terminate job, but list bad record.
EO=A	Accept bad records. No indication of error.
EO=AD	Accept and list bad records.

Jobs with many tapes may exceed the overall default mass storage allocation for a job, resulting in the message- 'JOB MASS STORAGE LIMIT EXCEEDED'. Users should add the following control card to their jobs before the TAPEIN and TAPEOUT cards- it requests the maximum permitted mass storage to be allocated to the job

```
LIMIT,0.
```

Temporary disk files (FIND)- TAPEIN only.

Creating temporary disk copies of a tape is recommended when the same tape is likely to be used again within a short period of time. For example, when a user is debugging a program. The job card parameter TPn indicating a tape job should be omitted to gain better turnaround if this option is being used.

ID=userid	userid is a 1-9 character name to identify the disk copy of the tape in subsequent jobs.
CY=cy	cycle number, normally omitted, 511 by default.

Purge 7600 FIND files (FIND)- TAPEIN only.

PU=YES	Purge the disk file with ID=userid and read the tape, creating a new disk file with the same identifier. Use this parameter with caution, especially if more than one job will be requesting access to the same file.
PU=ONLY	Simply purge the FIND file. Recommended method.

Expiration Date (LABEL) - TAPEOUT only

T=ddd	retention period in days. If T is not given the expiration date is set to infinite.

UNSTAGE CARD

The function of UNSTAGE is to inhibit staging out of a file, for example in the case of FORTRAN execution error -

```
UNSTAGE,lfn.
```

will inhibit staging out of lfn.

VSN CARD

The VSN card allows the user to read a labelled tape written elsewhere. The difficulty to be surmounted is that upon arrival at CERN, the tape is given a Visual IDentifier or VID. When such a tape is requested, the system must be given both the VID (so the tape reel may be located) and also the VSN (so that system label checking, which does not by default distinguish between VSN and VID, won't abort the job). The VSN card allows the specification of alternate VSN's, one of which will normally be the CERN VID and the other the tape label VSN field. The form of the VSN card is-

```
VSN,lfn,vid=vsn.
```

The VSN parameter can be omitted from the STAGE card if a VSN card is in use, and TAPEIN accepts both VSN and VID as parameters, generating a VSN card if required. For example-

```
FILE,TAPE1,RT=U,BT=K,MBL=5120,EO=DD.
VSN,TAPE1,12345A=RHEL66.
STAGE,TAPE1,NT,PE,NR.
```

The tape label contains the VSN 'RHEL66', and the CERN VID is 12345A. The form of TAPEIN for this tape would be-

```
TAPEIN,TAPE1,RT=U,VSN=RHEL66,VID=12345A.
```

Labelled multi-reel processing.

When the data is on more than one reel, TAPEIN's VID and VSN parameter cannot be used, but the VSN card (together with FILE, LABEL and STAGE as necessary) can be used. Its form becomes-

```
VSN,lfn,vid1=vsn1/vid2=vsn2/vid3=vsn3/.../vidi=vsni.
```

XPLOT CARD

The CERN control card XPLOT allows the user to get his files converted to paper plot format and disposed to plain or squared paper plotter.

According to its options, it generates control cards PLOTLIM, CPPLOT and DISPOSE.

All calls to XPLOT resulting in a call to the CERN Library conversion program CPPLOT will require the presence of the 7600LIBRARY.

```
XPLOT,lfn,options.
```

lfn File to be plotted

Disp	PLAIN (default) or SQUARED
T=plottim	Plot time limit in minutes - default is 20
CPPLOT=yes CPPLOT=no	lfn is first to be converted - default - lfn is ready to be disposed to a plotter
I=lfndir	lfndir contains the directives to CPPLOT If directives are on file INPUT, just specify I If no directives (normal case), omit this parameter

```
XPLOT,TAPE1,T=30,SQUARED.
```

Plot TAPE1 on SQUARED paper, plot time limit 30 minutes and CPPLOT conversion without directives before plotting.

4.1 INTRODUCTION

The FORTRAN compiler in CDC Scope is called FORTRAN Extended or FTN. Below is a list of the principal differences between it and other common FORTRAN compilers - the list, however, is by no means complete. FTN has many features which are not standard ANSI FORTRAN. In the CDC FORTRAN Extended Manual, descriptions of these non-ANSI features are shaded in grey. If you want to be sure that your programs can be easily moved from one make of machine to another, DON'T USE THEM. A comparison of CDC's FTN and IBM FORTRAN H Extended compiler (DD/US/3) may be obtained from the Self-Service Documentation Room.

PROGRAM card

FTN requires that the main program is headed by a PROGRAM card. This must declare all the logical filenames used by the user's program. The filenames should always include INPUT and OUTPUT to allow reading of cards and printing of output. FORTRAN statements such as READ (N,M), READ(N), WRITE (N,M) or WRITE(N) where N is a unit number require that the PROGRAM card contains the filename TAPEN. CDC FORTRAN uses the names INPUT and OUTPUT to refer to the card reader and line printer. READ and PRINT statements of the form below can be used for reading cards and printing results -

```
      READ 100,  X, Y, Z
      PRINT 100, X, Y, Z
  100 FORMAT (3F10.3)
```

If a program reads cards and prints results with statements of the form -

```
      READ (5,100)  X, Y, Z
      PRINT (6,100) X, Y, Z
  100 FORMAT (3F10.3)
```

then, the PROGRAM card must equivalence the files TAPE5 and TAPE6 to INPUT and OUTPUT -

```
      PROGRAM mine(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
```

If you use the PUNCH statement, you should also include PUNCH as a filename if the punching is formatted, and PUNCHB if it is not. It is possible to change the name of a file to be processed by a program- suppose a user reads from TAPE20 which is the fourth file on his PROGRAM card-

```
      PROGRAM mine(INPUT,OUTPUT,TAPE10,TAPE20,PUNCH)
```

He may decide to call this PKBAR in his control card deck, and he can inform his program of this change via the LGO card-

```
      LGO,,,,PKBAR.
```

Only the fourth file is 'renamed' here. Any FORTRAN references to logical unit 20 (TAPE20 in the PROGRAM card) will reference file PKBAR instead.

INTEGER and REAL specifications

IBM statements such as REAL*8 and INTEGER*2 are not accepted. Users who have worked on IBM machines should note that the 60-bit word of the CDC 7600 gives 15 significant figures for floating point without a DOUBLE PRECISION declaration, and that the Program Library contains routines BUNCH and BLOW (M436, M426) to pack and unpack small integers into 60-bit word arrays. The Library also contains subroutines to convert the floating-point number formats of the following machines to CDC 60-bit word format-

Machine	Routine	Library code
IBM-360	CVT360	M219
CII, SDS Sigma-7	CVTCII	M225
Hewlett-Packard	CNVTHP	M221
IBM-7090	CNVTZF	M201

Hollerith strings

Hollerith strings are delimited by the 'not equal to' or 'double quotes' character. The CDC manuals use the 'not equal to' character. The 029 punches, teletypes, VDU keyboards and most printers use the 'double quotes' character. For example, one might use -

```
      MONTH="JANUARY"
      CALL TITLE(N," I CANT THINK OF A GOOD TITLE")
```

In a FORMAT statement only, a Hollerith string may be delimited by the character * instead, for example -

```
20    FORMAT(* END OF FILE FOUND*)
```

DATA and NAMELIST statements

In an expression or a DATA statement, a Hollerith constant is limited to 10 characters. Thus to initialise an array with the letters of the alphabet, one would use either -

```
      DIMENSION ICHAR(3)
      DATA ICHAR/10HABCDEFGHIJ,10HKLMNOPQRST,6HUVWXYZ/
```

or -

```
      DATA ICHAR/"ABCDEFGHIJ","KLMNOPQRST","UVWXYZ"/
```

DATA and NAMELIST statements must be the last non-executable statements. NAMELIST will not accept Hollerith constants.

INTRODUCTION

RETURN and END in an OVERLAY

A RETURN statement may be used at the end of the main program of an OVERLAY. When FTN encounters END or a RETURN statement, control returns to the statement following the CALL OVERLAY statement.

SPYing on user jobs

Users who are concerned about the efficiency of their programs can use the utility SPY, described in the CERN Program Library Manual under L300, to find out how much time is spent in each part of their program.

4.2 LARGE CORE MEMORY

The maximum amount of space available to a user in the fast SMALL CORE MEMORY or SCM on the 7600 is about 140,000B or 49,000 (Decimal) words. However, the FORTRAN programmer has access to the slower LARGE CORE MEMORY or LCM via the LEVEL statement. Variables held in a COMMON block may be stored in LCM simply by declaring them to be LEVEL 2 or 3 variables, for example -

```
COMMON/C1/X(10000),Y(10000),Z(10000)
LEVEL 2,X,Y,Z
```

Variables held in LEVEL 2 are used in essentially the same way as ordinary variables stored in SCM. Variables stored in LEVEL 3 are accessed using a MOVLEV subroutine call (see the FTN manual). Note, however, that if a variable held in LEVEL 2 is used as an argument in a subroutine CALL, it must be declared to be in LEVEL 2 in both the calling program and the called subprogram. **For this reason, variables held in LEVEL 2 CANNOT be used as arguments in calls to CERN Program Library routines.** The amount of space available to the user in LCM is 400,000B or 131,072 (Decimal) words. Note, however, that every file requested by the user's job - not just those on his PROGRAM card, but 7600LIBRARY and so on (see BUFFERS control card) - will need anything up to 10,000B words of LCM buffer space. To economise on use of LCM, jobs should RETURN all files no longer needed by the job.

Dynamic allocation of LCM

Since users are charged for LCM usage on a 'kiloword-second' basis, they should always consider whether it is worth their while to dynamically request LCM. This can be done using the CERN Program Library routine INCLCM (Z026). The program below shows its use. Here A is occupying, nominally, 1000 words. However, after dynamically increasing the LCM field length to 10000 words, A can be treated as A(10000). Similarly, the allocation of LCM used can be reduced again by 10000 words whenever we wish.
This is the most efficient way to use LCM.

LARGE CORE MEMORY

```
      PROGRAM TEST (INPUT,OUTPUT,... )
      COMMON / SPACE / A( 1000 )
      LEVEL 2, A
C
C     DEFINE A - LCM ARRAY
C
      .
      . Process now needs 10000 words for A -
      .
      N=9000
      CALL INCLCM (N,NTOT,IERR)
C
C     N IS INCREMENT/DECREMENT - CHECK WE GOT ENOUGH
C     IERR IS THE DIFFERENCE BETWEEN N AND THE ACTUAL
C     CHANGE PRODUCED BY MEMORY
C
      IF (IERR.NE.0) CALL ENDRUN
      .
      . Suppose process now needs only 1000 words -
      .
      N=-9000
      CALL INCLCM (N,NTOT,IERR)
      .
      . and so on
      .
C
C     CHECK LCM QUANTITY ALLOCATED, N=0
C
      N=0
      CALL INCLCM (N,NTOT,IERR)
C
      PRINT 10, NTOT
   10 FORMAT (10H NTOT =    ,I7)
C
      .
      .
      .
```

4.3 MASS STORAGE ROUTINES

It is possible that even LCM will not provide enough storage space. Almost indefinite amounts of storage can be obtained on disk, and this is useful for large blocks of data when we only use one block at a time. FORTRAN supplies routines OPENMS, READMS , WRITMS and CLOSMS for efficient maintenance of mass storage files. In the example, TAPE12 is created on mass storage with 500 blocks of 1000 words. These may be accessed after creation, and possibly updated 'in place', by calls to READMS and WRITMS.

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
FTN,L,R=1.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
      PROGRAM MASS (INPUT,OUTPUT,TAPE12)
      DIMENSION INDEX (501), DATA (1000)
C
C     THE DIMENSION OF INDEX IS 1 + THE NUMBER
C     OF RECORDS IN TAPE12
C
      NWORDS=1000
      CALL OPENMS (12,INDEX,501,0)
      DO 10 I=1,500
C
C     CREATE DATA SET ON TAPE12
C
      CALL WRITMS (12,DATA,NWORDS,I)
   10 CONTINUE
C
C     CAN NOW OBTAIN ANY RECORD IN TAPE12
C
      .
      .
      NREC=60
      CALL READMS (12,DATA,NWORDS,NREC)
      .
      .
C
C     AND ALSO REWRITE IN PLACE ANY RECORD
C
      NREC=70
      CALL READMS (12,DATA,NWORDS,NREC)
      .
      .
C
```

```
C      MODIFY RECORD NOW IN DATA - SAY, MAY BE
C
       DATA (101)= 0.
c
C      - AND REWRITE IT IN PLACE
C
       CALL WRITMS (12,DATA,NWORDS,NREC,1)
       .
       .
       .
       END
end-of-information
```

Use of permanent mass storage files

Mass storage files, if not too large, can be catalogued as 6000 disk files. (Refer to the Chapter on permanent files for details). However, if a 7600 job wishes to update this file, the procedure is slightly awkward, since the 7600 cannot directly 'rewrite-in-place' a 6000 disk file. Therefore, the user must ATTACH the original file, use it and update it, then COPY it to a duplicate file to re-CATALOG on the 6000.
(One cannot ATTACH and CATALOG the same file in the same job). FILE cards are needed for the COPY, but should appear after all FORTRAN processing is complete.

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
ATTACH,TAPE12,massfile,ID=userid,ST=CCP.
.
.
FTN.
LDSET,MAP=B/ZZZZMP.
LGO.
.
.
FILE,TAPE12,RT=U,FO=WA.
FILE,TAPE13,RT=U,FO=WA.
REWIND,TAPE12.
COPY,TAPE12,TAPE13.
CATALOG,TAPE13,massfile,ID=userid,ST=CCP.
RETURN,TAPE12,TAPE13.
PURGE,TAPE12,massfile,ID=userid,ST=CCP,LC=1.
```

Repeated successful executions of this job will leave the user with one file whith the cycle number incremented by 1 for each run. If the job goes wrong, the file may be corrupted and valuable data and computer time lost. Dayfiles and outputs should be carefully checked after each run for unusual conditions. If in doubt, consult the PEO.

4.4 INTRODUCTION TO ERROR RECOVERY IN FORTRAN

Recovery from input/output errors

Users are advised to use the CERN X-package for Input/Output whenever possible, as it gives the best input error recovery facilities for the current system. One call to XSETIO is necessary to initialise recovery. Records should be read using -

```
CALL XREAD(lun,ipar,vec,m,n)
```

where 'lun' is the unit number, 11 for example if the data is on TAPE11. ipar is 1 for a binary read and 0 for a coded read (A10). Data is read into the array 'vec' from element 'vec(m)' to 'vec(n)'. A maximum of (n-m+1) words are transferred. If the input record contains fewer words it is not fatal, but the number of times this occurs will be listed under ERROR 89 in the error summary at the end of the FORTRAN output.

End-of-file and parity error status should be checked after each read using the logical functions XEOF and XRDIN, which must be declared as LOGICAL at the beginning of the routine.

XEOF(lun) - TRUE if end-of-partition or end-of-information was found on unit 'lun', or end-of-section on file INPUT.

XRDIN(lun) - TRUE if a parity error was found on unit 'lun'. CDC real function EOF(lun) and integer function IOCHEC(lun) are roughly equivalent. End-of-information can only be distinguished by successive end-of-file conditions with no data in between. The Library routine IXFPZL can detect end-of-record, section, partition or information. (See 7600LIBRARY write-up Z202).

Example of FORTRAN use of X-package

```
      PROGRAM TEST(INPUT,OUTPUT,TAPE11)
      DIMENSION B(100)
      LOGICAL XEOF,XRDIN
      DATA LUN/11/
      CALL XSETIO
      .
      .
   20 CALL XREAD(LUN,1,B,1,100)
      IF(XEOF(LUN)) STOP
      IF(XRDIN(LUN)) GO TO 20
      .
      .
```

If 'End-of-file', then stop reading. If 'Parity error', then skip the record, and read another.

For more details, see Z200 in the CERN Program Library Manual.

INTRODUCTION TO ERROR RECOVERY IN FORTRAN

The effect of the T option

The T option in the FTN call card gives full error traceback from library functions and user routines at the expense of execution time. Argument checking is provided unconditionally for the following functions-

EXP,ALOG,ALOG10,SIN,COS,SQRT,ATAN,ATAN2,ASIN,ACOS,CABS,SINH,COSH

The absence of the T parameter prevents some library functions , (for example, DMOD, TAN) from giving any indication an error has occured. Incorrect results (e.g. Indefinites, Infinites) are returned without notification. If T is specified on the FTN card, the library mathematical functions will return control to the user after printing an error message on the output file. The OPT=0 or D (DEBUG) parameters of the FTN card automatically activate T.

User program error recovery procedures

There are two classes of errors, firstly FORTRAN execution-time errors, and secondly system-detected errors, such as UNDERFLOW and OVERFLOW, for which the user can specify a recovery procedure within his programs. A special routine called SYSTEMC must be called to permit recovery from any of the FORTRAN errors listed in the FORTRAN Extended manual. Examples of its use may be found in Chapter 11. To recover from the system-detected errors, calls to a routine REPRIV are needed. Once again, examples of their use can be found in Chapter 11.

Note that use of MANTRAP excludes calls to SYSTEMC or REPRIV. If this is done by the user, MANTRAP will not give the correct answers.

Loading map

The Loading Map is provided according to the

MAP,option.

statement. The options are OFF (default), PART and ON. The PART option gives a short loading map of routines and COMMON blocks, without external references.

FTN listing and Symbolic Reference Map

The column of numbers appearing on the left hand side of the compiler listing contains line numbers, not a word count. Diagnostics are collected at the end of program units and refer to these line numbers. Execution time traceback messages also refer to line numbers and not to word addresses. There are four levels of Symbolic Reference Map, controlled by the R parameter of the FTN card.

R=0	is default, giving no map.
R=1 or R	gives symbol names, locations and types.
R=2	adds to this, line number references and a DO-loop map.
R=3	adds on COMMON block maps and EQUIVALENCE-pairs.

Dump

The Scope 2.1.2 dump is explained in detail in Chapter 9 to which users are referred. The FORTRAN post-mortem processor MANTRAP will interpret the dump and give an explanation in readable English, as far as is possible. If users wish to request a dump themselves, they may add the control card-

DMP,word1,word2.

The limits of the dump are words word1 and word2 in the SCM. These parameters must be octal numbers.

4.5 CARRIAGE CONTROL

Up to 65 lines can be printed per page if 6 lines are printed per inch (normal mode), however, with automatic page eject, only lines 4 to 64 are used, thus allowing for 61 lines. If the printer is switched to print 8 lines per inch (either manually or by the carriage control character T) 88 lines can be put onto a page, but only lines 5 to 85 are used if pages are skipped automatically. Please note that a print line can contain 136 characters plus a carriage control character. The following table lists the effect of all characters appearing in position 1 of a print line. Line numbers, however, apply to 6 lines/inch. They are also used for microfiche except for Q, R, S, T.

Character	CDC Scope effect
A	single space, post-print page eject
B,C	single space, post-print skip to line 64
D	single space, skip to line 4 or 34
E	single space, skip to line 4, 24 or 44
F	single space, skip to line 4, 19, 34 or 49
G	single space, skip to line 4
H	single space, post-print skip to line 2
I,J	single space, post-print page eject
K	single space, post-print skip to line 64
L	single space, post-print skip to line 1
M,N,O,P	single space - like blank
Q	clear auto page eject, suppress remainder of line
R	select auto page eject, suppress remainder of line
S	select 6 lines/inch, suppress remainder of line
T	select 8 lines/inch, suppress remainder of line
blank	single space
U,V,W	single space - like blank
X	page eject
Y	skip to line 64
Z	skip to line 1
0	double space
1	page eject
2,3	skip to line 64
4	skip to line 4 or 34
5	skip to line 4 ,24 or 44
6	skip to line 4, 29, 34 or 49
7	skip to line 4
8	skip to line 2
9	skip to line 4
+	no spacing
-	treble space

All special characters act like blanks, when placed in position 1 of a print line.

4.6 CERN PROGRAM LIBRARY ROUTINES- BINARY AND SOURCE FORM

BINARY FORM

The Scope 2.1.2 system provides the facility of user libraries, i.e. files which contain, in the general case, relocatable binary routines and which are created by LIBEDT. (See Chapter 3). They are structured such that they can be scanned extremely quickly by the loader when searching for unsatisfied calls to external routines. The CERN program library is stored on a permanent file called 7600LIBRARY, ID=PROGLIB. To gain access to the library, the following control cards must precede program loading -

```
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
```

Jobs running on the front-end MFA and MFB 6000 machines have access to all programs in the CERN Program Library which are not 7600-dependent. Batch jobs should replace the 7600LIBRARY of the above control cards with 6400LIBRARY. A list of routines contained in these files is available in Chapter 14.

SOURCE FORM

The source form (FORTRAN or Compass) of all library routines is available using the program GETPROG. This extracts all program packages requested by their Library code or package name, writing them to COMPILE. This file can be listed (via COPYSP) or punched, as in this example for routine BLOW, code M426 -

```
abcde.
ACCOUNT,name,group,accno.
FIND,GETPROG,GETPROG.
GETPROG,M426.   ( Alternatively, GETPROG,BLOW. ).
COPYSP,COMPILE,OUTPUT.
REWIND,COMPILE.
DISPOSE,COMPILE,PU=C.
end-of-information
```

More than one package can be requested at once- just list their Library codes or names, separated by commas, on the GETPROG card. Up to 25 items are accepted in the GETPROG parameter list. If a routine has more than one version - perhaps Compass or 6000- the default you get is 7600 Compass . FORTRAN or 6000 version can be requested by adding FORTRAN or 6000 to the GETPROG parameter list. Requests for 6000 or FORTRAN versions which do not exist are not fatal- the Compass or 7600 version will be produced instead. GETPROG can also be used to transfer the source code of 7600LIBRARY routines to magnetic tape for transfer to other laboratories. Parameters accepted by GETPROG concerned with tape output are listed in the table below. The tape format, however, is fixed-length records (RT=F,FL=80) with 20 records per block (RB=20 parameter for RT=F files). Users are always

required to specify the VSN parameter if they want tape output, but the other tape parameters are optional. Should the user wish to transfer the whole 7600LIBRARY to tape, he should also use the FULL parameter. Full details of GETPROG are to be found in the Program Library Manual.

Parameter	Description
VSN=vsn	Output to 7-track tape, unlabelled, 800 BPI in BCD
NT	This implies VSN=vsn is 9-track, unlabelled It is in EBCDIC code, 1600 BPI
US	If NT is specified, code will be ASCII
SL	VSN=vsn is a labelled tape

CATALOG as a GETPROG parameter, gives the Short Library Catalog.

4.7 MAINTENANCE AND UPDATING OF USER PROGRAMS

Two updating programs are available to users -

UPDATE - a CDC product (CDC UPDATE Reference Manual 60449900)

PATCHY - a program written at CERN and which has been installed on various other computers such as IBM, PDP 10 and UNIVAC.

The users choice of program will depend to a large extent on whether he is likely to be using more than one make of computer. For reasonably straight forward applications running only on CDC, UPDATE is probably simpler to use.

UPDATE

Users of UPDATE should note that UPDATE concerns itself with five files. These are called by default INPUT, OLDPL, COMPILE, NEWPL and SOURCE. UPDATE always reads directives to it from INPUT (or from lfn, if I=lfn is specified on the UPDATE card). These directives control the actions of UPDATE in creation or correction runs. The information to be corrected or maintained is normally on the file OLDPL (or lfn1 if P=lfn1 is present on the UPDATE card). This is a specially formatted and compacted file containing basically card images, together with information on the history of the card images- whether they are currently 'deleted', what 'correction set' they belong to, etc. The file OLDPL is initially made by an UPDATE CREATION RUN- see below for an example. UPDATE scans the file OLDPL unless it is a creation run, and, according to its directives, makes a file called COMPILE. This contains all the card images asked for by the user, with an identifier in cols. 74-90, if neither the D nor the 8 options are specified in the UPDATE control card (see below Section on common UPDATE control card parameters). The identifier consists of the user's card sequencing and UPDATE's identifier. The COMPILE file can be used as input to FTN, and renamed by using C=lfn2 on the UPDATE card.

Users can ask for a new UPDATE 'OLDPL' to be created with the N or N=lfn3 parameter. The contents of lfn3 (NEWPL by default) are determined by the user's directives and his UPDATE parameters. Lastly, the file SOURCE can be requested by the parameter S or S=lfn4. This file will contain all cards required to recreate the NEWPL.

UPDATE creation run

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
```

MAINTENANCE AND UPDATING OF USER PROGRAMS

```
COPYS,INPUT,TAPE10.
REWIND,TAPE10.
UPDTSC.
REWIND,TAPE11.
UPDATE,I=TAPE11,N,W,L=1234.
CATALOG,NEWPL,myupdatefile,ID=userid,ST=CCP,RP=120.
end-of-section
.
. FORTRAN deck
.
end-of-section
.
. UPDTSC data, needed if 2 routines have the same name- see below
. Otherwise, just two end-of-section cards
end-of-information
```

The program UPDTSC (see short write-up L410) reads one section of data from the file INPUT. Normally this is an empty section, but it is necessary sometimes to differentiate between routines with the same name in the FORTRAN decks to be scanned by UPDTSC. After reading these data cards (if any), UPDTSC scans the rest of the TAPE10 file and generates *DECK cards to identify separate SUBROUTINE, FUNCTION and OVERLAY sections of the user's deck. The prepared file is called 'TAPE11' above. It is then read by UPDATE, which produces a specially formatted file called NEWPL. This file contains all the user's cards, uniquely identified by an UPDATE identifier of the form **deckname.n** where n is the card number. The deckname is the same as the routine name. These identified cards can be replaced or temporarily deleted by subsequent UPDATE runs, as shown below.

UPDATE correction run

```
abcde.
ACCOUNT,name,group,accno.
ATTACH,OLDPL,myupdatefile,ID=userid,ST=CCP.
UPDATE,L=1234.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
FTN,I=COMPILE.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
.
. UPDATE directives - insertions etc.
.
end-of-information
```

UPDATE expects as input, under the name OLDPL, the specially formatted file created in the first job. Normally it produces an output file called COMPILE containing card images ready for compilation. If an N is present as a parameter on the UPDATE control

card, a file called NEWPL is produced. It has the same format as OLDPL, but contains the corrections that have been made. If this file is kept by cataloging, it can be used as an OLDPL in subsequent runs.

The contents of the file COMPILE depend on the form of the UPDATE being made. For UPDATE,Q only those decks named on *COMPILE directives are written to the file COMPILE. Without the Q parameter, those decks changed in this run plus all those decks named in *COMPILE directives are written to the COMPILE file. UPDATE,F writes all decks to COMPILE.

Common UPDATE directives

*IDENT name	identifies a set of corrections
*INSERT identifier	insert the following FORTRAN cards after the identified card
*DELETE identifierl,identifier2	delete the cards from identifierl to identifier2 inclusive. If the *DELETE is followed by normal FORTRAN cards, these will be inserted in place of the deleted cards. If only a single identifier is given, only that single card will be deleted
*COMPILE decknamel,deckname2,..	write the named decks to COMPILE

EXAMPLES

*IDENT FEB77	identify corrections by FEB77
*INSERT FCN.35 J=0 I=I+1	insert cards J=0 and I=I+1 after card 35 of routine FCN
*DELETE TCPG.10,TCPG.23 CALL SUBX	replace lines 10-23 of TCPG by CALL SUBX

Consult the UPDATE Reference Manual, 60449900, for details.

Common UPDATE control card parameters

Q	quick mode- only decks requested by *COMPILE directives are considered
F	full mode- the entire OLDPL is transferred to COMPILE. For the 7600LIBRARYPL,ID=PROGLIB which contains the 7600 library source, this is more than 100,000 card images
8	COMPILE contains 80-column card images for punching

MAINTENANCE AND UPDATING OF USER PROGRAMS

D	Data width -80 columns instead of 72 if no D. D implies 8
L=1234	lists the UPDATE DIRECTIVES encountered, and any errors
R	no rewind is made of any of the UPDATE files before or after the UPDATE run, and an end-of-partition is written. Default is to rewind files before and after the UPDATE run
N	UPDATE is requested to produce the file NEWPL, for use in subsequent UPDATE runs as an OLDPL. This file NEWPL will contain the results of the new corrections
W	UPDATE is requested to make NEWPL a SEQUENTIAL file. The default is RANDOM, and this cannot for example be written to tape or used on the MFA 6000/MFB 6000, where UPDATE expects RT=S sequential files.

INTRODUCTION TO PATCHY ON THE 7600

The PATCHY family of programs is designed to handle libraries of source-code. Its specifications are described in the PATCHY reference Manual available from the Program Documentation Oficce. PATCHY version 4 has been written for speed and machine independance and it has been installed on several CDC, PDP 10, UNIVAC 1100, IBM 360/370 machines in high energy physics laboratories.

Preparing a PAM-file

A **PAM-file** is a sequential string of cards-images, divided into **'patches'** by the PATCHY control cards **+PATCH,pname.** Any patch may, but need not, be further divided into **'decks'** by cards **+DECK,dname.** Cards on the file before the first +PATCH or +DECK make up the **'title-deck'**, its very first card is the title card of the file, the first word on this card is the **'PAM-identifier'**. Usually each subroutine is one deck, and several subroutines which belong together are grouped into a patch. The whole PAM-file may well be a single patch. The skeleton of a PAM-file might look thus -

```
ABCD   /1          title-card with PAM-id ABCD, version-level /1
+PATCH, A.
 .
 .   CDE-definitions   see below for 'sequences'
 .
+DECK, MAIN.
 .
 .   Main prograà
 .
+DECK, RA.
```

```
      .
      .   Subroutine RA
      .
     +DECK, RB.
      .
      .   Subroutine RB
      .
     +DECK, RC.
      .
      .   Subroutine RC
      .
```

Frequently, identical COMMON-DIMENSION-EQUIVALENCE statements (CDE for short) have to be placed into several or all the subroutines of a FORTRAN program. PATCHY allows one to give such cards once at the beginning as a **'sequence'** with **+KEEP,sname.** and to call for inclusion of the sequence-material anywhere with **+CDE,sname.** For example -

```
+PATCH,A.
+KEEP, QBITS.
      COMMON /QBITS/IQDROP,IQMARK,IQCRIT,IQUEST(300)
      DIMENSION QUEST(300)
      EQUIVALENCE (QUEST(1),IQUEST(1))
+KEEP, BM.
      COMMON /BM/LINKS(12),BM(9,11),BMM(9,11,3)
```

defines the sequences 'QBITS' of 3 cards and 'BM' of 1 card. They may then be called in this context -

```
+DECK, RB.
      SUBROUTINE RB
+CDE,QBITS,BM.
      DIMENSION F(124)
      ...
```

requesting of YPATCHY to replace the card +CDE,QBITS,BM. by the material of the sequences QBITS and BM in this order. If the same sequence is defined twice, the first definitation over-rules any subsequent definition.

Having punched all the cards of a PAM file one can compact them with the program YTOBIN, and put them on disk ready for use by YPATCHY, for example with this job.

```
abcde.
ACCOUNT,name,group,accno.
FIND,ulib,ULIB,ID=PHLIB.      Library with all PATCHY programs.
LIBRARY,ulib.
COPYS,INPUT,CARDS.
YTOBIN,CARDS,PAM.             Convert CARDS into PAM file.
CATALOG,PAM,filename,ID=userid,ST=CCP,RP=nnn.
```

```
YLIST,PAM.                      List PAM file.
YEDIT,PAM,AWAY,,,EOF.           Print index of PAM file.
end-of-section
.
.  PAM file on cards
.
end-of-information
```

The PATCHY auxiliary program YTOBIN converts a card file CARDS to a compact binary file PAM.

The program YLIST may be called to get a straight card-for-card listing of the PAM-file, separated into decks, showing the card numbers needed when addressing any card.

Running YPATCHY

With YPATCHY the user can fetch from the PAM-file exactly those subroutines which he wants to, modify them as he pleases, and write the results to the 'ASM-file' ready to input to the compiler.

YPATCHY uses 4 files -

PAM - the user's previously created PAM file
ASM - the file to receive the desired subroutines ready for compilation
INPUT- the so-called 'cradle' containing the control-cards by which the user drives the YPATCHY run, program modifications requested and even new decks or patches. Cards in the cradle are considered to belong to the 'blank deck' of P=CRA*, until deck or patch name are changed by +DECK,dname or +PATCH,pname in the cradle.
OUTPUT- the file to receive messages and listings written by YPATCHY.

A call to YPATCHY ,showing the default file names ,has the form

```
YPATCHY,PAM,ASM,INPUT,OUTPUT.
```

Program modifications

The resulting compiler material derived by YPATCHY from a PAM file may be modified by action cards. There are four action cards available to add new material before +ADB or after +ADD a specified card, to replace +REPL a specified card or group of cards by new material and to delete +DEL a specified card or group of cards as shown in these examples -

MAINTENANCE AND UPDATING OF USER PROGRAMS

```
+ADB, P=A, D=RB, C=12.        explicit parameter-keys
     .
     .  New material
     .
+ADD, A, RB, 24.              short-hand
     .
     .  New material
     .
+REPL,,, 42-73.               patch/deck as previous
     .
     .  New material
     .
+DEL, C=109.
```

Cards are addressed by giving patch-name,deck-name and card-number, groups are specified by giving first and last card-number separated by 'minus'. Sequences can easily be changed by giving a re-definition in the cradle.

Select program version and processing modes

The wanted program version is selected by +USE -cards giving the name of all patches which should be taken into the making of the program. +USE. without parameters selects everything.

The routines to be compiled must be selected by +EXE -cards in the cradle. This may be done in 3 different ways.

A) global — a card +EXE. without parameters, given together with +USE. causes the whole program to be transferred to the ASM-file.

B) direct — a card +EXE,pname. or +EXE,pname,D=dname selects all routines of P=pname or the routine contained in D=dname of P=pname for compilation (plus routines elsewhere which may be modified from the selected patch or deck, by indirect selection)

c) indirect — a card +EXE,CRA* given in the 'blank deck' of the cradle-patch P=CRA* selects all routines which are modified from CRA* (plus routines contained in P=CRA* by direct selection). This allows to re-compile just the modified routines.

The decks to be listed by YPATCHY may be selected by +LIST -cards in the cradle in exact analogy to +EXE. To get the cradle listed but not routines modified from the cradle one needs +LIST,CRA*,T=ONLY.

YPATCHY example

The following is the schema of a cradle -

MAINTENANCE AND UPDATING OF USER PROGRAMS

```
        +USE.                       single version program, accept
                                    everything
this    +EXE.                       compile whole prograà
or      +EXE, CRA*.                 compile modified routines
or      nothing                     no compilation, listing only

this    +LIST.                      list everything
or      +LIST, CRA*, T=ONLY.        list CRA* without propagation and
        +LIST, CRA*, D=.            list decks modified from blank deck
or      nothing                     no listing (except blank deck of CRA*)

        +ADD, ...                   program modifications (with list, if
                                    +LIST,CRA*,D=.)
        .
        +DECK,WITHOUT.              following cards in deck WITHOUT, not
                                    blank deck,so....
        +ADD, ...                   program modifications (without list,if
                                    +LIST,CRA*,D=.)

        +OPTION, MAPASM.            print map of decks transferred to ASM
        +PAM.                       switch YPATCHY input to PAM-file

        +DECK, dname.               New routine, CDE's from PAM now
                                    available
        .
        . New routine
        .
        end-of-section
```

The following is the schema of a job to run YPATCHY-compile-load-go

```
abcde.
ACCOUNT,name,group,accno.
FIND,mypam,....
FIND,mylib,....
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,ulib,ULIB,ID=PHLIB.
LIBRARY,ulib.
YPATCHY,mypam,asm.
RETURN,ulib.
FTN,I=asm.
LIBRARY,mylib,lib.
LGO.
end-of-section
.
.  YPATCHY cradle
.
end-of-section
.
.  user data
.
end-of-information
```

5.1 COPY UTILITIES

The COPY utilities, except for COPY, COPYL and COPYSP, all have the form-

```
COPYx,lfn1,lfn2,n.
```

File lfn1 is copied to file lfn2, and n decimal RECORDS, SECTIONS or PARTITIONS are copied depending on the suffix x. For all copies except COPYL, the default input and output logical file names are INPUT and OUTPUT. The default number of records, sections or partitions, n, to be copied is 1. Files are not rewound either before or after a COPY operation. The delimiters recognised by the COPY routines are end-of-record, end-of-section, end-of-partition and end-of-information.

An **end-of-record** is detected after a logical record. A FORTRAN WRITE statement writes a logical record.

An **end-of-section** in an input deck is a 7/8/9 card (7,8,9 multipunched in column 1). There is no way of writing an end-of-section from FORTRAN. Only W records (blocked and unblocked) and Z records with C blocking can be grouped into sections. The Library routine WEOR (K402) writes an end-of -section.

An **end-of-partition** is the same as the term end-of-file as commonly used in FORTRAN.

An **end-of-information** is detected at the very end of the information in a file. On a tape, two hardware tapemarks with no data in between or an EOF label specifyes end-of-information. For a card deck, it is signified by a card punched with 6/7/8/9 in column 1.

Parity errors

The action of the copy utilities on finding a parity error on an input file depends on the error option (the parameter EO) specified on the corresponding FILE or TAPEIN card. For TAPEIN, the error option defaults to DD in which case the bad data is not transferred, and the COPY continues. To terminate the COPY on a parity error, put EO=T on the TAPEIN card. If the user supplies his own FILE and STAGE cards, the COPY assumes EO=T and will terminate on a parity error. To drop the bad data and continue use EO=DD. If it is wished to copy the bad data, EO=AD should be put on the TAPEIN or FILE card.

Maximum record length

When using the COPY utilities to convert a file from record type X or S to record type W, it is necessary to specify the maximum record length, MRL=mrl, where mrl is the maximum number of 6-bit characters per record. This parameter should be put on the FILE, TAPEIN or TAPEOUT card of the file as appropriate. The maximum record length

that can be handled by the COPY utilities is 30000B words (i.e. 12288 decimal) words.

COPY

File lfnl is copied to file lfn2, each from its current position up to the end-of-information on lfnl. The files are not rewound after the copy. If COPY detects an end-of-section or an end-of-partition on the input file, an equivalent delimiter is written on the output file. At end-of-information, COPY writes an end-of-partition on the output file if the last operation was anything other than to 'write an end-of-partition'.

COPYP

This utility copies PARTITIONS from the current position of lfnl. Note that lfn2 is not rewound before the copying. The COPYP terminates when the PARTITION count is exhausted or when end-of- information is encountered on the input file. Upon completion of the copy, an end-of-partition is written on the output file if the last operation was anything other than 'write an end-of-partition'.

COPYS

This utility copies SECTIONS from the current position of file lfnl. Note lfn2 is not rewound before the copying. If COPYS encounters an end-of-section on the input file,an end-of-section is written on the output file. COPYS terminates when n sections have been read from the input file and written to the output file. If COPYS encounters an end-of-partition on the input file,an end-of-partition is written on the output file and COPYS terminates. COPYS also terminates when end-of-information is encountered on the input file. In this case,COPYS writes an end-of-section on the output file. Note that SECTIONS are only defined for the following record types -W records blocked or unblocked and Z records with C blocking. S records with C blocking are also treated as SECTIONS by the utilities. In a file containing relocatable binary routines, each routine forms a separate section.

COPYR

COPYR copies RECORDS from lfnl to lfn2 from the current position on file lfnl. This utility may be used to copy both binary records and coded records (such as cards images) from a file. If an end-of-section or an end-of-partition is encountered on the input file, an equivalent delimiter is written on the output file, and COPYR terminates.

COPY UTILITIES

COPYSP

This utility prepares a PARTITION for printing by inserting a blank character at the start of each print record. The first character of the original record is not therefore, used as a carriage control character. Only one partition is read, and a page throw (a single record with 1 as the first character) is inserted first of all into lfn2. COPYSP only works for RT=W unblocked (disk) files.

COPYL

This utility is not quite the same as any of the other COPY routines. It is used to perform sequential copy with replacement on the user's LGO binary files. This can be useful when you are making changes to one or two routines of a large, already compiled and cataloged program. The control card has the form -

```
COPYL,lfn1,lfn2,lfn3.
```

A) - lfn1 is the previously created LGO file - default OLDLIB
B) - lfn2 is the newly compiled LGO file containing the routines to be changed - default BINARY
C) - lfn3 is the updated version of OLDLIB - default NEWLIB

The utility copies lfn1 onto lfn3, but replaces routines in lfn1 if routines of the same name appear in lfn2. Routines included in lfn2 but not in lfn1, DO NOT GET COPIED onto lfn3. Neither lfn1 nor lfn2 is rewound before or after the processing.
For example -

```
FIND,OLDLIB,myoldlgo,ID=userid.
FTN,L.
REWIND,LGO.
COPYL,OLDLIB,LGO,NEWLIB.
NEWLIB.
```

Here we replace routines in 'myoldlgo' with new ones compiled by FTN, and execute the file NEWLIB so formed. Note that LGO must be rewound - COPYL will not do this. COPYL will list in the dayfile any substitutions it has made from the file LGO.

5.2 SKIP UTILITIES

SKIPF

Skipping records forward

SCOPE 2.1.2 does not have the facility of skipping records forward. The technique which can be used is to copy the records to be skipped onto a dummy file and to then return the dummy file.

Skipping sections forward

```
SKIPF,lfn,n.
```

This utility can be used to skip sections on those file types which support sections - RT=Z,BT=C and RT=W - and can also be used for skipping RT=S records. Parameter n is a decimal count of the sections (or S type records) to be skipped. The default is n = 1.

Skipping partitions forward

```
SKIPF,lfn,n,17.
```

This utility can be used to skip partitions on all types of files . The parameter 17 indicates that n partitions are to be skipped. The default is n = 1.

SKIPB

Skipping records backward

SCOPE 2.1.2 does not have the facility for skipping records backward. The user must rewind the file and copy records to a dummy file.

Skipping sections backward

```
SKIPB,lfn,n.
```

Sections can be skipped backward on those files which support section delimiters - RT=S or RT=W -. SKIPB cannot be used for skipping RT=Z records backward. The BKSP (backspace) utility can also be used to skip sections backwards.

SKIP UTILITIES

Skipping partitions backward

SKIPB,lfn,n,l7.

Skipping partitions backwards works as expected only for RT=W, unblocked files. For other record types the positioning of the file is unpredictable.

BKSP

BKSP,lfn,n.

BKSP is in fact the same utility as SKIPB. The default for lfn is FILE and the default for n is 1. Backspacing terminates when beginning-of-information is reached. The utility interprets S records as sections. Please use this utility with care as results can sometimes be unexpected. Never use it for RT=Z records. BKSP can be used to backspace over an end-of-partition on RT=W unblocked and RT=S. BKSP forces staging of a PRE-staged tape or of a 6000 permanent file.

REWIND

REWIND,lfn1,lfn2,...,lfnn.

This utility can be used to position one or more files to the beginning-of-information. REWIND does not force staging of a PREstaged tape or of a 6000 permanent file

DMPFILE FOR DUMPING CONTENTS OF TAPE OR DISK FILES

5.3 DMPFILE FOR DUMPING CONTENTS OF TAPE OR DISK FILES

DMPFILE gives a dump in octal and in characters of the contents of tape and disk files. The optional parameters restrict the quantity of output, as DMPFILE always reads to end-of-information and summarizes the number of words, records, sections and partitions for the complete file. The control card format is-

```
DMPFILE,lfn,XW=i,XR=j,XS=k,XP=l.
```

The logical file name, lfn, is compulsory, and the following parameters restrict the amount of output and are optional.

Warning- the default is to dump the whole file to OUTPUT.

```
XW=i  dump the first i words of every record
XR=j  dump the first j records of every section
XS=k  dump the first k sections of every partition
XP=l  dump the first l partitions.
```

Example

To scan the contents and format of a 9-track unlabelled tape VSN=12345A, of unknown record format by listing the first 2 records of every file-

```
TAPEIN,TAPE1,UL,RT=U,VSN=12345A.
DMPFILE,TAPE1,XR=2.
```

5.4 MERGE UTILITY FOR MERGING BINARY FILES

MERGE is a utility program which merges up to 5 binary input files into one output file with the option of copying end of partitions or not. Input files are copied from their current position to their end-of-information, and are not rewound after the merge. Data on the input files which contains parity errors is not copied onto the output file.

The program call has the form

```
MERGE,lfnout,lfninl,lfnin2,...        end-of-files not copied
MERGE,lfnout,lfninl,... ,ENDF=NO.     end-of-files not copied
MERGE,lfnout,lfninl,... ,ENDF.        end-of-files copied
MERGE,lfnout,lfninl,... ,ENDF=YES.    end-of-files copied
```

The sequence of input files (up to 5 allowed) is terminated by the ENDF parameter, or by a full stop (or closing bracket) if the default ENDF is selected.

The program is available on the permanent file

```
7600MERGE,ID=PROGLIB,ST=CCP
```

and uses routines from the 7600 Library.

Examples

A) Merge 2 unlabelled 7-track experimental data tapes, IN1 and IN2, onto a labelled 9-track default format tape, OUT ,without copying end-of-files.

```
abcde,TP1.
ACCOUNT,name,div,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
FIND,MERGE,7600MERGE,ID=PROGLIB.
TAPEIN,IN1,MT,UL,VSN=12345A,RT=U.
TAPEIN,IN2,MT,UL,VSN=12346B,RT=U.
TAPEOUT,OUT,VSN=67890X.
MERGE,OUT,IN1,IN2.
end-of-information
```

B) Merge files T1, T2, T3, and T4 to T20 copying end-of-files.

```
MERGE,T20,T1,T2,T3,T4,ENDF=YES.
```

C) Merge files TAPE1 and TAPE2 onto file TAPE10 without copying end-of-files.

```
MERGE,TAPE10,TAPE1,TAPE2,ENDF=NO.
```

5.5 FICHE PROGRAM FOR MICROFICHE OUTPUT

Usage of program

Microfiche output should be used instead of printout if the user's output is very long, if several copies are needed, or if users merely wish to retain reference copies of listings, etc. Normally, 207 pages can be written to each 15 cm by 10 cm microfiche. Users require a special viewer to read the microfiche, but these are available in the PEO's and at the Computer Centre user area. If you may need a private viewer for your group, they are available from the CERN Stores. The SCEM number is 54-97-00-100-0. The fiche program reads a 'print' file, and transforms it into a 'fiche' file suitable for transfer to the QUANTOR 105 microfiche machine. The actual transfer is done via unlabelled, 9-track, 800 BPI tape but this need not concern 7600 users. The 6500 or 6400 jobcard, however, requires the additional parameter NT1 . The tapes used by default for transfers to the microfiche machine all have VSN=FICHE and belong to a pool of 44 tapes used only for microfiche operations and maintained by the Computer Centre.

This program is designed to be used via a control-card macro call. For MFA or MFB jobs, it is assumed that the record type is RT=Z with C blocking.

Modifications to user's control cards

To use the program in his jobs, the user should add the following control cards to his job deck. They will give the default mode of operation- the user's OUTPUT file (as it exists before the microfiche jobstep) will be rewound and read by the fiche program. The user's OUTPUT file is also rewound after the execution of the microfiche jobstep, and will not be printed unless the user takes steps to COPY and DISPOSE it before calling QMAC60 or QMAC70). He will normally obtain on paper only his dayfile and a summary of the fiche program execution.

7600 Jobs

```
.
FIND,QMAC70,QMAC70,ID=PULIB.
QMAC70,DEFER.
```

6500 or 6400 Jobs- note need for NT1

```
abcde,CP60,NT1.
.
.
ATTACH,QMAC60.
QMAC60.
```

FICHE PROGRAM FOR MICROFICHE OUTPUT

Notes.

To change the file name for transfer to fiche, use 'FILE=lfn' as a parameter.

Your output is spooled for later transfer (saving you the cost of a tape stage) by specifying 'DEFER' in the call, but on the 7600 only.

Other options available

Many additional options are available to users, of which the most used are mentionned here. To request them, the user must specify OPTIONS (to read directives from INPUT) or OPTIONS=lfn. The directive cards all have format and then the bits marked out from FICHE write-up. This can be found on file FICHEUSERGUIDEPEDOC, ID=PULIB, on MFB, also available from the WRITEUP macro described in Chapter 1.

The time needed to generate a full fiche of 207 pages plus index page and title is approximately 3.5 seconds on the 7600, and 40 seconds on the 6500 or 6400. It can vary, increasing if pages are very full or heavily overprinted, etc. A page can contain up to 64 lines of user print-out, plus a microfiche tag line identifying the page. This tag is at the 'top left' corner. Users should also note that the QUANTOR 105 microfiche device restricts the length of a print-line to 132 printed characters, maximum. Additional characters are truncated by the program without warning to the user.

More information about the way of using this program and modifying its options can be found in the long write-up edited by Charles Curran and available from the Documentation Office (DD/US/6).

CONTROL CARD MACRO GENERATION - CATALOGED PROCEDURES

5.6 CONTROL CARD MACRO GENERATION - CATALOGED PROCEDURES

Users can manufacture their own 'control cards' with the control card macro generation program, DEFCCM. This program is available on all CDCSCOPE machines, but macros created on the 7600 will only execute on the 7600 and macros created on the MFA and MFB 6000 front-ends will only execute on the 6000s. One useful feature of 6000 macros is that they can be used under INTERCOM. More detailed information on the DEFCCM program than are given here may be found in Systems Group Note 94.

Why use macros at all?

Many users have regularly running jobs, containing sequences of 'fixed' control cards. Others run jobs where only the VSN or filename changes. To punch a new jobdeck each time or modify the original is really a waste of time, since the user could create a macro to do it all for him. Such macros are the rough equivalent on CDCSCOPE of the more familiar IBM catalogued procedures. At the simplest level, the user's macro will expand into a set of control cards. More useful are user-created macros which can be called with parameters. . These parameters can not only be substituted into the control cards- as filenames, perhaps- but can also directly control the expansion into control cards. The TAPEIN and TAPEOUT macros are good examples of what can be done by way of simplification in user's jobs. The casual user will probably not need to write such a complex macro himself but the following example might be typical.

The example shows the creation of a user 'control card' XFTN, which will compile and execute a user's program on the 7600, making the 7600LIBRARY and MANTRAP available. The input file for compilation may be on cards (the file INPUT), a local file (perhaps COMPILE, the output from UPDATE) or a permanent file on the MFA 6000, created with the INTERCOM EDITOR and so an RT=Z file.

Example of 7600 macro creation

```
abcde.
ACCOUNT,name,group,accno.
DEFCCM.
CATALOG,XFTN,XFTN,ID=userid,ST=CCP.
end-of-section
ENTRY XFTN I=COMPILE/INPUT
DEFAULT ID NONE
FIND,ZZZZZ0,7600LIBRARY,ID=PROGLIB.
FIND,ZZZZ01,MAN,ID=OPSYSTEM.
LIBRARY,ZZZZ01,ZZZZZ0.
IFEQ $ID NONE
JUMP LOCAL
COMMENT.** FILE TO COMPILE IS NOT LOCAL - ATTACH IT **
FILE,ZZZZ02,RT=Z,BT=C,FL=80.
ATTACH,ZZZZ02,$I,ID=$ID,ST=CCP.
```

CONTROL CARD MACRO GENERATION - CATALOGED PROCEDURES

```
ENTRY LOCAL
IFNE $I INPUT
REWIND,$I.
COPYS,$I,ZZZZ02.
REWIND,ZZZZ02.
COMMENT.** FORTRAN INPUT SECTION READY ON ZZZZ02 **
FTN,I=ZZZZ02,L.
LDSET,MAP=B/ZZZZMP.
LGO.
REVERT.
end-of-information
```

The user's macro, XFTN, is CATALOGed on the MFA 6000 for future use. Note that to create a similar macro on either 6000, the user would need the CP60 parameter on his jobcard and also a

```
REQUEST,XFTN,*PF
```

control card to permit CATALOG of the macro XFTN.

Using the 7600 macro created above.

All that needs to be done to use it in a subsequent 7600 job is the following-

```
abcde.
ACCOUNT,name,group,accno.
FIND,XFTN,XFTN,ID=userid.
XFTN,I=lfn,ID=userid.
end-of-information
```

The job dayfile will show the complete expansion of the macro XFTN into control cards. The parameter I defines the input file for FTN, and has the same interpretation as the standard I parameter of FTN. I alone implies I=COMPILE, no I implies I=INPUT and I=lfn implies lfn as the input to FTN. If the ID=userid parameter is given, the file is ATTACHed from MFA.

Format of DEFCCM input data

As is obvious from the above, the majority of commands to DEFCCM used to create the user macro are control cards. These, once they are put in the macro, need not be punched or entered again by the user. Less trivially, the user macro could take advantage of the following facilities-

A) User PARAMETERS are allowed, with default values.

B) Macros can contain CONDITIONAL JUMPS , which may alter the flow of control card execution according to user's parameters. Thus sections of control cards in the macro can be skipped, if

the user supplies a parameter and asks DEFCCM to do a jump, depending on the parameter value.

C) Elementary LOOPING facilities are provided. Sections of control cards in the user macro can be executed many times, perhaps printing a number of files, AUDITing a group of 'consecutive' IDs, etc.

In order to understand the 'Macro language', the write-up edited by Charles Curran and available from the Documentation Office (DD/US/7) or through the WRITEUP macro (see Chapter 1) should be consulted.

INTRODUCTION TO RECORD AND BLOCK TYPES

The Record Type of a file tells the system how logically to interpret the physical data. A logical record is that data which is normally transferred by a single READ or WRITE statement. Nine different record types are allowed by CDC SCOPE, six of which are likely to be of interest to the programmer at CERN.

The table below gives the Record Type, RT, associated with data coming from different sources. The column on the right lists other parameters relevant to each Record Type. It is not always necessary to specify these- some are often assumed by default and others have default values which apply in most cases. When necessary, these parameters would appear either on the TAPEIN/TAPEOUT or FILE cards. (see examples below and refer to Chapters 3 and 7, where FILE and TAPEIN, TAPEOUT are described). Later sections give details about each Record Type and the use of associated parameters.

Source of the file	RT	Other relevant parameters
7600 files		
Disk file written on 7600, default format	RT=W	(note it is unblocked)
Tape file written on 7600, default format	RT=W,	BT=I
Binary tape written on non-CDC computer (IBM, PDP, HP)	RT=U,	BT=K, MBL=mbl
CERNSCOPE binary tape	RT=X,	BT=C
CDC SCOPE standard binary	RT=S,	BT=C
Coded records in an external code- say card images in BCD or EBCDIC	RT=F,	BT=K, CM=YES, RB=rb, MBL=mbl, FL=80
Disk file of card images created on MFA/MFB by INTERCOM	RT=Z,	BT=C, FL=80

6000 files

FORTRAN unformatted READ/WRITE default on 6000	RT=W, BT=I
FORTRAN BUFFER default on 6000	RT=S, BT=C
Formatted FORTRAN READ/WRITE	RT=Z, BT=C, FL=fl

It should be noted that the default Record Types are different on the 7600 and the MFA /MFB machines. This affects users of INTERCOM who should read the section on Record Type Z. The Record Type of a file is determined by where the file is created, not by where it is stored. Thus, a file created in default format on the 7600 and stored on the 6000 disk will have RT=W, whereas a file created in default format in a 6000 job and stored on the 6000 disks will normally have RT=Z or RT=W, BT=I (FORTRAN jobs) or RT=S (utilities).

6.1 RECORD TYPE W

Record Type W - Control Word Record - is the default format used for all files on the 7600. Fortran READ/WRITE defaults to RT=W, BT=I under SCOPE 3.4.3 on the MFA/MFB 6000s. W records are normally handled automatically by the system. They can be of any length, and can contain either binary or character information. When using a STAGE card in preference to TAPEIN it is necessary to have a FILE card specifying EO=DD in order to recover from parity errors.

Each W-type record consists of an integral number of 60-bit words preceeded by a W-control word, which (together with other control information) contains the record length. The W-control word is added by the Record Manager on output and stripped off by the Record Manager on input. For details of the information carried by the W-control word, see the SCOPE 2.1.2 User's Guide.

A disk file is by default UNBLOCKED, that is the whole file is one continuous string of data in which the W-control words are used to determine where one record ends and the next one begins. On the other hand, tape files are ALWAYS BLOCKED with Block Type I- Internal blocking - BT=I. With block type I, the records are packed into groups of 512 words, the first of which is an I-control word, giving (among other information) the block number and the position of the first record in the block, since records may span blocks. These I-control words are handled automatically by the system.

Example of writing 9-track 1600 BPI labelled RT=W,BT=I tape

```
TAPEOUT,TAPE1,VSN=12345A.
```

6.2 RECORD TYPE U

Record type U (undefined) is used for reading binary tapes not written in one of the CDC SCOPE standard formats. Tapes written on experimental or non-CDC computers are normally read as RT=U tapes. This format can also be used to write tapes for such computers. One 'logical record' corresponds to one physical block on the tape. For both 7- and 9-track tapes, the tape can be either labelled or unlabelled. The preferred format for reasons of security and packing density is 9-track, 1600 BPI, labelled. Labelled tapes (both 7- and 9-track) should contain only one file for ANSI compatibility. All unlabelled tapes must be terminated by two consecutive tapemarks. Otherwise, the user must either use partial staging or copy his tape on the MFA 6000 or MFB 6000 machine to produce a correctly terminated tape. Record type U should always be used with the FILE card, containing at least -

```
FILE,lfn,RT=U,BT=K.
```

This is because unblocked files in RT=U cannot be read by FORTRAN.

The default maximum block size is 5120 characters or 512 60-bit words. The user need only specify MBL if he wishes to override the default. When interchanging tapes between a CDC 60-bit machine and any other computer having a different word size, it is recommended that a block size of 504 60-bit words be used, since this is an even multiple of most experimental machine word sizes.

For reading or writing tapes where the records are longer than 512 60-bit words, the MBL parameter must be specified (MBL=n, where n is the maximum number of 6-bit characters). The largest possible block length is 153,540 bits, MBL=25590. It is, however recommended to avoid the use of blocks longer than 512 60-bit words as it is less efficient and more error-prone. Reading blocks of more than 512 words requires the use of two Peripheral Processors (PPs) with subsequent timing problems. Since these PPs are also heavily used - 24 of them handling all 7600 INPUT/OUTPUT - this should be avoided. The block length should be a minimum of 144 bits to conform with industry standards for noise records. The user is cautioned that when reading a RT=U, BT=K tape having record lengths which do not fit exactly into 60-bit words, the last word on the record is padded with zeros. However, the Library routine LXBITS can be used to find the exact length of record read.

Example of reading 7-track 800 BPI unlabelled RT=U tape

```
TAPEIN,TAPE1,MT,UL,RT=U,VSN=12345A.
```

6.3 RECORD TYPE X- READING ONLY

Record type X - short block terminator - is the record type to be used for reading binary tapes written under the former CERNSCOPE system. Tapes CANNOT be written with RT=X.

CERNSCOPE tapes should be read as unlabelled on the 7600 . CERNSCOPE labels were written between the two load points at the beginning of the tape and the CDC SCOPE systems automatically start reading from the second load point.

If the tape was written before 14/2/1972 or if it is written up to the end of tape it will be terminated by a single file mark and should be read using partial staging. (See Chapter 7- Section 6). For example, a 7-track 800 BPI tape -

```
TAPEIN,TAPE1,UL,MT,RT=X,VSN=$12345A/F/0/1$.
```

If all records on a CERNSCOPE tape have less than 512 words, it is possible to read the tape as record type U (RT=U). If the tape has parity errors it is strongly recommended to do this as the parity error recovery code is much more reliable for RT=U than for RT=X. It is not possible to read RT=X 9-track CERNSCOPE tapes on the 7600 if the records contain an odd number of 60-bit words. They can be read as RT=U - contact the PEO, if you have difficulty.

6.4 RECORD TYPE S

Record type S - SCOPE logical record - is the default format of binary files for the utility programs, (such as COPY) on the MFA and MFB 6000 machines. However, 6000 FORTRAN defaults to RT=W, BT=I for unformatted READ/WRITE statements, to RT=S, BT=C for BUFFER statements and to RT=Z, BT=C for formatted READ/WRITE statements.

Also RT=S should be used for an UPDATE OLDPL which is to be processed on both the 7600 and the 6000 MFA/MFB.

On the 7600, to create the OLDPL file, use -

```
FILE,NEWPL,RT=S.
UPDATE,N.
CATALOG,NEWPL,myoldpl,ID=userid,ST=CCP.    (or ST=CCQ)

        to use it -

FILE,OLDPL,RT=S.
FIND,OLDPL,myoldpl,ID=userid,ST=CCP.       (or ST=CCQ)
UPDATE.
```

On the 6000, to create the OLDPL file, use -

```
REQUEST,NEWPL,*PF.
UPDATE,N,W.
CATALOG,NEWPL,myoldpl,ID=userid.

        to use it -

ATTACH,OLDPL,myoldpl,ID=userid.
UPDATE.
```

Each record consists of an integral number of 60-bit words, followed by an 8-character (48 bits) level number appendage. The appendage is supplied on output and removed on input by the Record Manager. The level number may indicate the logical end-of-record or end-of-file.

6.5 RECORD TYPE F

Record type F - Fixed length records - is the record type normally used for reading and writing coded files that have to be transferred between different computers. The records must be of fixed length, the number of characters per record - fl - being given by the parameter

FL=fl

Several records may be packed into one block, the number per block being given by the parameter

RB=rb

where the default is RB=1. Normally the data will be converted into an external code- EBCDIC, ASCII or BCD. This is done with the TAPEIN/TAPEOUT parameter

CM=YES

If the data is on a 9-track tape it will be assumed to be in EBCDIC. If it is in ASCII the US parameter must appear on the TAPEIN/TAPEOUT or STAGE card. A 7-track tape can only be in BCD code. If the block length exceeds 5120 characters, the maximum block length, mbl, must be given by the parameter

MBL=mbl

It is not recommended to exceed a block length of 5120 characters - the same comments apply as for RT=U.

Read 9-track EBCDIC labelled tape, 40 card images/block

```
TAPEIN,TAPE1,RT=F,FL=80,RB=40,CM=YES,VSN=12345A.
```

Write an ASCII COMPILE file to 9-track 1600 BPI unlabelled tape
1 card image/block

```
FIND,OLDPL,myoldpl,ID=userid.
UPDATE,Q,8.
TAPEOUT,TAPE1,RT=F,FL=80,UL,VSN=12345A,CM=YES,US.
COPY,COMPILE,TAPE1.
```

6.6 RECORD TYPE Z

Record type Z - Zero byte terminator - is the default record type used by the MFA and MFB 6000 machines for disk files containing coded information, such as card images and EDITOR files. It is the format required for INTERCOM EDITOR files. When using these files on the 7600 it is necessary to specify the number of characters per record, fl, by giving the parameter

```
FL=fl
```

If a 7600 job is creating a card image file which the user wishes to use under INTERCOM EDITOR, it is necessary to have a FILE card before the file is created-

```
FILE,lfn,RT=Z,BT=C,FL=80.
```

Note- it is NOT possible to write the COMPILE file from UPDATE directly in RT=Z. It must be first written in 7600 default format and then copied to a file for which RT=Z has been specified-

```
UPDATE,Q,8.
FILE,prog,RT=Z,BT=C,FL=80.
COPY,COMPILE,prog.
CATALOG,prog,prog,ID=userid,ST=CCP.
```

The file 'prog' can now be used with INTERCOM EDITOR.

With record type Z, trailing blanks are removed from the end of a record and the end of record delimited by 12 bits of zero (a zero byte) in the least significant end of a 60-bit word. On input the zero bits are stripped from the record and blank padding is inserted to fill out the record to the FL parameter as specified in the FILE card. On output, the Record Manager supplies the terminating 12-bit zero byte.

7.1 GENERAL INTRODUCTION TO TAPEIN/TAPEOUT

The 7600 is an unusual machine in that it has no direct access to magnetic tape units. Instead, tapes are copied by the system onto a disk, and the data is read from this file. This process gives problems as well as advantages, and tape users should read the later section on 'Tape Staging' to understand more about it.

The physical and logical properties of a tape are most conveniently specified by using the CERN control cards TAPEIN (reading) and TAPEOUT (writing). The TAPEIN/TAPEOUT card has the format-

```
TAPEIN,lfn,VSN=uservsn,options.
```

Here lfn is the logical file name, uservsn is the 'Volume Serial Number' and there may follow optional parameters depending on the tape being read. The user of TAPEIN/TAPEOUT will see from his dayfile that it creates for him LABEL, FILE, VSN, FIND and STAGE cards as necessary. These are described in the Chapter on control cards. The LABEL card gives details of the ANSI labels on the tape, if any. The FILE card gives details of the logical arrangement of the data on the tape- record type and block type, etc. The VSN card is used to specify alternate VSNs for tapes labelled outside CERN, which do not have the same VID (Visual IDentifier) as VSN (tape label Volume Serial Number). Users of such tapes must specify both VSN and VID on the TAPEIN/TAPEOUT card. The FIND card instructs the system to make a temporary copy of the tape on the 7600's disks- this saves the re-stage of the tape, if it is to be used several times in different jobs. Finally, the STAGE card gives physical details of the tape itself- the number of tracks, recording density, etc.

7.2 EXAMPLES OF TAPEIN/TAPEOUT

A) Using tapes can best be described by this example in which a FORTRAN program reads data from tape 12345A, written in variable length binary records on an external computer. The tape is assumed to be seven track and unlabelled (MT,UL on TAPEIN). After processing, the user's output data is written onto tape 67890X, with a label indicating what the data is.

```
abcde,TP1.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
TAPEIN,TAPE3,RT=U,VSN=12345A,MT,UL.
TAPEOUT,TAPE4,VSN=67890X,L=DSTFEB75.
FTN,L.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
      PROGRAM TEST(INPUT,OUTPUT,TAPE3,TAPE4)
      DIMENSION BUF(512),BUFO(200)
C
C     USE Z200 X-PACKAGE ( XREAD ) TO READ TAPE3
C
      LOGICAL XEOF,XRDIN
      CALL XSETIO
C
 10   CALL XREAD(3,1,BUF,1,512)
C
C     STOP IF END-OF-FILE, IGNORE PARITY ERRORS
C
      IF(XEOF(3)) STOP
      IF(XRDIN(3)) GO TO 10
C
C     FIND NUMBER OF BITS, WORDS TRANSFERRED
C
      NBITS=LXBITS(3)
      NWORDS=IXLONG(3)
      .
      . User processes data from TAPE3 and writes results onto
      . TAPE4- a 200 word array, say
      .
      WRITE(4) BUFO
      GO TO 10
      END
end-of-information
```

The JOB card and ACCOUNT card above are common to all 7600 jobs. The CERN Program Library and the MANTRAP post-mortem processor are made available to the job by the FIND and LIBRARY cards. The LDSET card writes the Loading Map to the file ZZZZMP for post-mortem

processing, if this is necessary. The functions IXLONG and LXBITS (from the 7600LIBRARY) are used to find the number of words (rounded up to a multiple of 60 bits) and the exact number of bits transferred by the call to XREAD.

A) To read a 9-track labelled tape in 7600 default format, making a temporary copy of the tape on the 7600 disks-

```
TAPEIN,TAPE1,VSN=12345A,L=DSTFEB75,ID=userid.
```

The same card should be used whenever the file is to be read. The system itself decides whether a copy of the tape exists on disk or whether to read the tape itself.

B) To read a 7-track unlabelled tape from an experimental computer-

```
TAPEIN,TAPE1,VSN=12346B,MT,UL,RT=U.
```

MT indicates a 7-track tape, UL that it is not labelled, RT=U that it is of record type U.

C) To read a CERNSCOPE 7-track tape- these have RT=X -

```
TAPEIN,TAPE1,VSN=12347C,MT,UL,RT=X.
```

D) To read a 9-track tape, unlabelled, containing card images-

```
TAPEIN,TAPE1,VSN=12348D,UL,RT=F,FL=80,RB=40,CM=YES.
```

The records are fixed length (RT=F), with 80 characters per record (FL=80), 40 records per block (RB=40), in EBCDIC code (CM=YES).

E) To write an unlabelled 7-track binary tape to take to another machine-

```
TAPEOUT,TAPE2,VSN=12349E,MT,UL,RT=U.
```

F) To read an externally labelled 9-track tape , VID=81234X, VSN=SAC123, written on a non CDC machine-

```
TAPEIN,TAPE1,VSN=SAC123,VID=81234X,RT=U.
```

Limitation of TAPEIN/TAPEOUT

TAPEIN/TAPEOUT cannot presently specify continuation reels with the VSN parameter, for example VSN=12345A/12346B. If continuation reels are required, the STAGE card must be used in conjunction with the FILE, LABEL and FIND cards as necessary.

7.3 TAPE STAGING

The 7600 configuration at CERN is such that the only I/O devices attached directly (on-line) to the 7600 are the three 7638 and three 844 disks. All other I/O (cards, output listings, tapes etc.) is controlled by the STATION - a software package which executes on the front-end machines, the MFA and MFB 6000s under SCOPE 3.4.3.

In order for a job executing on the 7600 to read a tape, the tape is read on a 6000 by the station program and the data is transferred or PRE-staged to the 7600 disks (staged information may go on 7638 as well as 844 disks). The 7600 job then reads the data from the 7000 disk file. A similar operation occurs when a 7600 job writes a tape. The data is actually written as a 7000 disk file. When the job terminates or the file is unloaded, the data is transferred or POST staged to the station which then writes the tape.

PRE-staging occurs when the file is first referenced. A user who has a TAPEIN or STAGE control card for a file first opened by a FORTRAN program might notice that the staging does not begin until the program has attempted to open the file. The use of TAPEIN with the ID parameter or the FIND utility changes this situation in that staging is forced immediately if the file is not found as a 7600 permanent file.

Tape staging makes the practice of extending a tape file by first skipping the information already recorded and then adding new information at the end both inconvenient and dangerous. If an error occurs during the process of writing the tape, information accumulated by a previous run is likely to be irretrievably lost. Instead the original information should be copied onto a second file, new information added onto this file, and then the complete file POST-staged to a second tape.

Scratch tapes

Staging also makes it impossible to know at execution time whether the end of a tape will be reached on output. If insufficient tapes are given in the VSN parameter remaining data will be written onto a scratch tape (VSN=SCRTCH). Since limited numbers of scratch tapes only are available, users who wish to transfer their results from a scratch tape to one of their own should contact the tape receptionist in building 513 (tel 4939) **within 48 hours** to find out which scratch tape was used. They should take the dayfile of their job with them.

7.4 LABELS

Tape labels are designed to help with the identification and protection of data. The labels used by CDC are ANSI (American National Standards Institute) standard labels. These labels can be processed by many other computer systems. Tapes at CERN are either classified as LABELLED or UNLABELLED. They cannot be moved from one group to the other without being processed by the tape receptionist. If it is desired to change a tape

A) From unlabelled to labelled, it must be PRE-LABELLED

B) From labelled to unlabelled, it must be DE-LABELLED

ALL THE DATA ON THE TAPE IS LOST WHEN THIS IS DONE.

All tapes issued by the tape receptionist since June 1st 1976, have been labelled unless specifically requested as unlabelled. The label written on the tape is a file consisting of two records in a special format that is automatically recognized by the system. The first record contains the tape number (Volume Serial Number - VSN parameter of the TAPEIN/TAPEOUT card) and the Owner identification (O parameter). The second record contains the user's Label identification (L parameter).

Owner Identificaton - O parameter

Owner identification is of interest only when a tape is written. It is a nine character field similar to that used for permanent files, indicating the current owner of the tape. It is specified by the O parameter on the TAPEOUT or LABEL card, e.g.

```
TAPEOUT,lfn,VSN=12345a,O=ddgggxxxx.
```

where ddggg are the owner's division and group code (e.g. EP512) and xxxx are up to 4 optional characters chosen by the owner. If the O parameter is not given it will be constructed from the division and group code of the ACCOUNT card followed by 4 blanks.

All new tapes issued by the Centre and existing tapes that are requested to be pre-labelled are pre-labelled with an Owner Identification consisting of the owner's division and group code followed by 4 optional characters given by the user - blanks if not given.

When the system (6000 or 7000 jobs) receives a request to write on a tape, this will only be allowed if -

1) - all 9 characters of the O parameter match the Owner Identification written on the tape

OR 2) - the first 5 characters (division and group code) of the O parameter match the first 5 characters of the Owner Identification on the tape and the other 4 characters on the tape are blanks

OR 3) - all the 9 characters of the Owner Identification are blank.

If none of these conditions are satisfied, the two fields will be printed in the dayfile and the tape given read only status.

The effect of these conditions will be -

1) - a tape with more than the default Owner Identification will be very highly protected

2) - when writing on a tape that contains the default Owner Identification it will be able to increase the level of security without having to have the tape prelabelled

3) - it will be possible to write on an existing labelled tape without it being prelabelled.

User's Label Identifier - L parameter

Whenever a tape is written, it is possible to specify a new file identifier to be put in the label with the parameter -

L=userlbl

where 'userlbl' is 1 - 17 character field. If it contains special characters such as + , / - or blanks, it must be enclosed in $ signs.

If the L parameter is not given the file identifier field is set to blanks. When reading a tape, if the L parameter is given (for example L=DSTJAN75) this value is checked against the value written on the tape and the job is aborted if the check is not correct. If the L parameter is not given, no check is made.

Note however that if it is wished to move a labelled tape from the CDC to an IBM the L field must be non-blank and for simplicity contain 8 or less alphanumeric characters. If the L field is blank Bypass Label Processing (BLP) has to be used to read the tape on the IBM.

LABELS

Expiration Date

It is possible to specify an expiration date for a tape using the T parameter -

T=ddd retention period in days

If T is not specified the expiration date is set as infinite.

On the CDC machines at CERN when a request is made to write on a tape the expiration date is ignored and so unexpired files can be overwritten. However on most other machines, including IBM, a request to write on a unexpired tape results in a request to the operator for permission to overwrite. Consequently if it is wished to move tapes freely between the CDC and IBM it might be desirable to specify T=0 when writing on the CDC so that the tape can be written on later on the IBM. (The default retention period on the IBM is zero days).

Security

Although tape labels give a good measure of security to a tape against operator error, it is quite simple for a user or someone else in his group to accidentally overwrite a valuable tape. For this reason, it is recommended that users who want to be sure that their tapes are not overwritten should attach the special yellow stickers-

DO NOT WRITE ON THIS TAPE

available from the computer receptionist. Note that at CERN the retention period in the label is ignored and so cannot be used as a protection method.

Multi-file labelled tapes.

SCOPE 2.1.2 on the 7600 does not support ANSI standard multi-file labelled tapes. It does however deal successfully with multi-partition labelled files in CDC internal formats (that is, RT=W and RT=S).

For labelled tapes with RT=U or RT=F, there should only be a single file. It is however possible to process a multi-partition tape if the partitions are separated by a single hardware tape mark and not by ANSI EOF and HDR1 labels- tapes in this format are not ANSI standard.

Note - It is possible to read ANSI standard multi-file labelled tapes on the 6000 machine and convert it to a format that can be used in the 7600. If you receive a tape in this format, contact the PEO.

7.5 MULTI-VOLUME FILES

Processing will be successful for labelled tapes of all record types, and for unlabelled tapes provided records do not span blocks. Thus, it will never work for unlabelled tapes with RT=W or for tapes with RT=S if the S records are longer than 511 words.

A) Multi-volume processing is automatic for labelled tapes. A specific call to CERN Library routine XTPCH2 is necessary for invoking reel-swapping of multi-volume unlabelled files.

B) All volumes of the file must have the same characteristics, such as tape density and record type.

C) All tape numbers (VSNs) are specified in the VSN parameter on the STAGE card (TAPEIN/TAPEOUT cannot be used) -

VSN=12345A/12346B/12347C

where 12345A, 123456B and 123457C are the tapes to be read.

D) For **writing labelled** tape files it is important to specify enough tape numbers in the STAGE card. On writing, trailer label processing and reel swapping is automatic when end-of-tape is encountered.

E) For **reading labelled** tape files it is sufficient to specify all tape numbers on the STAGE card. The complete information will be staged automatically. However, all tapes comprising one file should be specified in the correct order.

F) For **writing unlabelled** tapes it is necessary to specify all tape numbers in the STAGE card. On writing, reel swapping is automatic when end-of-tape is encountered.

G) For **reading unlabelled** tapes it is necessary to specify all tape numbers in the STAGE card and to implement

```
CALL XTPCH2(lun,nerror)
```

in the FORTRAN program whenever a new tape after the first is to be read. The parameter lun is either a constant or a variable identifying the logical unit number. XTPCH2 is described as part of the Library package Z200. Here, nerror returns the Record Manager error number, if any, and is zero otherwise. Even if the user reads a multi-volume unlabelled file with the STAGE card SF option, it is still necessary to call XTPCH2 to advance over the end of each reel.

H) FIND does not work for a labelled multi-volume file. If it is used for unlabelled files the SF parameter must be put on the STAGE card- for example

```
STAGE,TAPE1,VSN=12345A/12346B,SF,UL.
```

Note also that XTPCH2 will return zero and not Record Manager error 65 (no more tapes available) when it reaches the end of a FIND file.

7.6 PARTIAL STAGING

For certain tape formats, the user can specify that only a portion of the data is to be staged. This is done by using a special form of the VSN parameter on the TAPEIN or STAGE card.

VSN=$vsn/type/n/m$

type is either F (Files) or B (Blocks). The type specification applies to both the n and m parameters.

n is the count of the decimal number of files or blocks to be skipped before staging begins.

m is the decimal number of files or blocks to be staged.

Partial staging is recommended in the following cases - if the user is debugging a program and need not read the entire data tape, or if the input data tape is not properly terminated by two tapemarks.

Examples

A) To stage the first 50 records of a 9-track, unlabelled experimental data tape-

```
TAPEIN,TAPE1,RT=U,VSN=$12345A/B/0/50$,UL.
```

B) To stage the first 100 or so records of a labelled RT=W,BT=I tape where the average record length is 650 words-

number of blocks to transfer= 100*650/512, or 120 roughly

```
TAPEIN,TAPE1,VSN=$12346B/B/0/120$.
```

C) To stage the 4 files of data on an unlabelled 7-track data tape that has one tapemark after each file of data but does not have two consecutive tapemarks terminating the last file of data-

```
TAPEIN,TAPE1,MT,UL,RT=U,VSN=$12347C/F/0/4$.
```

D) To stage the third file of an unlabelled tape containing BCD card images-

```
TAPEIN,TAPE1,MT,UL,RT=F,FL=80,CM=YES,VSN=$12348D/F/2/1$.
```

PARTIAL STAGING

Notes

1) If on partial staging the error diagnostic SKIP COUNT ERROR occurs, the skip count is incorrect. That is, end-of-information was encountered before the skip count was satisfied.

2) If, when skipping either files or blocks a read parity error occurs, the skip function is terminated and staging begins with the next data block.

3) If, when skipping blocks an end-of-partition is encountered, the skip function is terminated and staging begins at the beginning of the next file.

Summary of PARTIAL STAGING possibilities.

VSN parameter-	RT=W	RT=U	RT=X	RT=S	RT=F
$vsn/B/O/n$ labelled	OK(1)	OK	no	OK(2)	OK
" unlabelled	OK(1)	OK	OK(2)	OK(2)	OK
$vsn/B/m/n$ labelled	no	OK	no	OK(2)	OK
" unlabelled	no	OK	OK(2)	OK(2)	OK
$vsn/F/O/n$ labelled	no	OK	no	no	OK
" unlabelled	no	OK	OK	no	OK
$vsn/F/m/n$ labelled	no	no	no	no	no
" unlabelled	no	OK	OK	no	OK

Note on blocks and records above.

1) For RT=W, records are packed into blocks of 512 words, the system adding one control word per record and one control word per block. The user must estimate the number of blocks required to give him the data he wants. For example, to transfer about 1000 records of average length 150 words -

NBLOCKS= (150 + 1)*1000 / (512 - 1) = 300, roughly.

2) For RT=X and RT=S, the number of blocks per record is-

NBLOCKS= 1 + (RECORD LENGTH/ 512)

7.7 RECORDING DENSITY/TAPE CAPACITY

The default recording density is 1600 BPI for 9 track tapes and 800 BPI for 7 track tapes.

For 9 track tapes the possible densities are 800, 1600 BPI.
For 7 track tapes the possible densities are 200, 556, 800 BPI.

For 9 track tapes inter-record gap = .6 inch.
For 7 track tapes inter-record gap =.75 inch.

Standard length of tape = 2400 ft.
Short tape = 1200 or 300 ft.

The maximum numbers of blocks that can be written to standard tapes, with 512 words per block, are approximately as follows-

Density-	1600	800	556	200 BPI
9-track	9200	5300	-	-
7-track	-	4000	2800	1090

Thus the maximum numbers of words that can be written to standard tapes, if the records are in default format and reasonably long (> 400 words) are approximately as follows-

Density-	1600	800	556	200 BPI
9-track	4,700,000	2,700,000	-	-
7-track	-	2,000,000	1,400,000	590,080

The user is cautioned that the new 667 tape units do not record at 200 BPI. Although tapes at 200 BPI can be read by the 667 units, it is extremely inefficient and the use of 200 BPI tapes is discouraged. Although it is possible to read blocks longer than 512 60-bit words, this is also not recommended since the software required is both less efficient and less reliable than that used for reading blocks of a shorter length.

7.8 PARITY ERROR RECOVERY- ERROR OPTIONS, OPERATOR ACTIONS

Error option

The error option (the EO parameter on the TAPEIN or FILE card) determines what an executing program should do when it finds a parity error. The possibilities are given in the FILE card section of the Chapter on control cards.

If TAPEIN is used to access the tape the error option is set by default to EO=DD. To change this option, the required EO parameter must be put on the TAPEIN card. If the tape is accessed by the user via the STAGE card explicitly, a FORTRAN program will by default take EO=AD, whereas system utilities such as COPY will by default take EO=T.

Console operator action

When a parity error occurs the console operator may or may not be informed depending on how the tape is accessed, and whether it is a 'closed shop' tape. If TAPEIN is used, or the NR parameter is put on the STAGE card, the operator is not informed of the parity error. The data is passed to the 7600 disk according to the EO option. If STAGE is used without the NR option, on finding a parity error a message is sent to the console operator and reading stops. Normally the operator gives a 'GO' to continue reading, and the data is passed to the 7600 disks but there is no indication that a parity error occured.

For closed shop tapes, the operator is always informed of parity errors regardless of how the tape is accessed. Normally he will force the tape to be re-read on a different unit after it has been cleaned.

In the case of a solid write parity error on any tape, it is cleaned and mounted on a different unit. If it should again have a solid write parity error, a pool tape is mounted replacing the original tape. A message is put in the user's dayfile and he has the possibility of recovering the data.

Even in cases where the operator is not asked to intervene on parity errors, it is possible that he may drop the job if he sees from the slow advance of the tape and the system dayfile that the tape is proving difficult to read.

7.9 TAPE AND DISK FILES

By default, tape files are always blocked, and disk files unblocked (except for RT=S). Thus, if a file is staged from tape to disk and the disk file cataloged, it is cataloged as a blocked file complete with a label if the tape was read as a labelled tape. On subsequent attaches of the file it is necessary to include a FILE card to specify the block type, and a LABEL card if the tape was labelled.

Job mass storage limit

By default, each job is limited to a maximum of about 16,500,000 words of mass storage for all the files being used by the job. This corresponds to a little more than 3 full 9-track tapes. Consequently, if a job is processing a lot of data it may be necessary to release disk space when it has been finished with, by RETURNing files, or to remove the limit by using the control card-

```
LIMIT,0.
```

Returning files

When a job has finished processing a file of data the file should be released from the system by use of the control card -

```
RETURN,lfn1,lfn2,....
```

Here lfn1, lfn2 and so on are logical file names such as TAPE11. Alternatively, they may be returned from a FORTRAN program by calling the library routine RETRNF (K510) -

```
CALL RETRNF(n)
```

where n is the logical unit number, such as 11. Releasing files like this frees both LCM and disk space and thus helps to increase machine efficiency. An example of the use of such techniques might be-

```
COPYP,TAPE1,TAPE10.
RETURN,TAPE1.
COPYP,TAPE2,TAPE10.
RETURN,TAPE2.
END-OF-SECTION
      .
      .( FORTRAN )
      .
      CALL XREAD(NF,1,BUF,1,1000)
      IF(XEOF(NF)) GO TO 30
      .
30    CALL RETRNF(NF)
      .
```

TAPE AND DISK FILES

TAPE AUDIT PROGRAM

The Tape Audit Program allows any INTERCOM user to obtain either a complete list of all tapes that are presently allocated under his name, or the checkletter for one particular tape under the condition that it belongs to the user specified.

The program is loaded through the following two INTERCOM commands -

```
FETCH, TAPEAUD,PROGLIB.
TAPEAUD
```

INTERCOM replies -

```
ENTER USER NAME
```

The user may now enter a name, Format (A10), under which tape have been allocated. INTERCOM will respond with a list of all tapes with their checkletters that have been allocated to that name.

The user can also enter name and tape number, Format (A10,I5), in which case the program will list only the tape specified with its checkletter on condition that it is allocated to that user.

Very long tape lists or lists for several users are better produced by the batch version of Tape Audit. See note 4 below -

1. Exit from the program by entering the word END or by aborting as specified for the particular type of terminal used.

2. Input to Tape Audit is a 6000 PF which is updated approximately weekly.

3. Typical response time for each request is about 15 seconds + time to output the tape list.

4. Tape Audit can be run in batch mode as well - use control cards

```
abcde,CP60.
ACCOUNT,name,group,accno.
FETCH,TAPEAUD,PROGLIB.
TAPEAUD.
end-of-section
name1
name2
....
END
end-of-information
```

5. TAPEAUD is available on both front-ends and may be used from any terminal including job enquiry terminals.

7.10 TAPE HANDLING ON THE 6000'S

The front-end 6000 computers can provide useful facilities for tape copying and dumping and for dealing with problem tapes. Described below are the control cards that would be necessary for doing simple dumping and copying operations. Other possibilities are available for dealing with problem tapes and you are advised to contact the PEO for assistance with these.

JOB card

```
abcde,CP60,MTm,NTn
```

abcde	1-5 character job name
CP60	says job is to be run on the 6000
MTm	m is the maximum number of 7-track tape units needed
NTn	n is the maximum number of 9-track tape units needed

If m or n is zero, the corresponding parameter can be omitted.

ACCOUNT card

```
ACCOUNT,name,group,accno.
```

same as for the 7600.

TAPEIN

```
ATTACH,TAPEIN.
TAPEIN,lfn,VSN=12345X,options.
```

TAPEIN is a macro that will generate the necessary 6000 REQUEST and LABEL cards. Its parameters are the same as the 7600 TAPEIN except that the partial staging and ID parameter are not valid.

TAPEOUT

```
ATTACH,TAPEOUT.
TAPEOUT,lfn,VSN=123456,options.
```

This again is a macro that will generate the necessary 6000 REQUEST and LABEL cards. Its parameters are the same as the 7600 TAPEOUT.

PRINTBF

```
PRINTBF,lfn,nw,nr,nf.
```

This is a 6000 utility for dumping tapes and permanent files with RT=U or RT=S format. The data is dumped in octal and display code in a format very similar to that of DMPFILE on the 7600.

nw	number of words per block to be printed - default nw=12
nr	number of blocks per file to be printed - default nr=10
nf	number of files to be processed - default all files

Unlike DMPFILE, PRINTBF does not scan the file up to the end-of-information and then give a summary, but stops after nf files.

PRINTCF

PRINTCF,lfn,nw,nr,nf.

This should be used for dumping files in BCD (from 7-track tapes) or EBCDIC (from 9-track tape).

CCOPY

CCOPY,mode,lfn1,lfn2,options.

This makes a **bit for bit** copy of lfn1 to lfn2. It does not allow the user to provide FILE cards and change the record type of a file as is the case with the COPY utilities on the 7600.

Mode	Type of COPY required
F=f	copy f files
R=r	copy r records
EOI	copy to end-of-information
EOV	copy to end-of-volume
DUB	copy to a double end-of-file
lfn1	logical file name of the input file
lfn2	logical file name of the output file

Options	Description
I	Ignore multiple EOFs. Multiple EOFs are treated as a single EOF.
II	Do not copy (write) any EOFs to the output file (except a closing EOF if the next option 'EOF' is given also).
EOF	Make certain that the last 'write' operation to the output file is an 'EOF write'. By this option you can always trust that an eventual next copy to the same output file starts beyond an EOF.

N	Do not print informative-only messages in the dayfile. Only effective errors or possible errors (warnings) are printed.
EO=T	Terminate CCOPY, if a read error is encountered on input file - default.
EO=D	Drop, i.e. do not copy the present full block/S record if a read error is encountered.
EO=A	Accept, i.e. copy as much as possible the erroneous block/S record.
MBL=mbl	Maximum Block Length - necessary only if the maximum block length exceeds 2666B or 1334 decimal CDC words.

Note

1) 'BLANK TAPE READ' is treated as end-of-information.

2) Since the copy is done on a block-by-block basis the EO=D option must not be used on a tape where records may span blocks (e.g. RT=W,BT=I). The output tape will be readable only if it has RT=U, RT=F or RT=S.

3) Users are warned that when copying a tape froà 7-track to 9-track or vice versa, if the record length is not a multiple of both 6 and 8 bits the last frame on the output tape will be filled with zeros.

EXAMPLES

PRINTBF

A) To dump the first 20 words of the first 5 records of all the files in a 9-track unlabelled raw data tape coming from a small computer -

```
abcde,CP60,NT1.
ACCOUNT,name,group,accno.
ATTACH,TAPEIN.
TAPEIN,tape1,UL,RT=U,VSN=12345X.
PRINTBF,tape1,20,5,999.
end-of-information
```

B) To dump the whole of the first 5 blocks of the first file of a labelled 9-track tape written in 7600 default format (RT=W, BT=I). Note that PRINTBF dumps blocks and not records, so the dump will show the first 5 blocks complete with the Record Manager control words that have been inserted.

```
abcde,CP60,NT1.
ACCOUNT,name,group,accno.
ATTACH,TAPEIN.
TAPEIN,tape1,VSN=12345Y.
PRINTBF,tape1,5,1,1.
end-of-information
```

CCOPY

For tapes with RT=W,BT=I, CCOPY should be used only for making an exact copy of good tapes. It will not work for removing parity errors or multiple end-of-partitions.

A) To copy a perfectly good RT=W, BT=I, unlabelled tape onto a labelled tape -

```
abcde,CP60,NT2.
ACCOUNT,name,group,accno.
ATTACH,TAPEIN.
TAPEIN,T1,UL,VSN=12345X.
ATTACH,TAPEOUT.
TAPEOUT,T2,VSN=67890Y.
CCOPY,EOI,T1,T2.
end-of-information
```

B) To make a copy of a 7-track, unlabelled raw data tape removing a series of embedded end-of-files, throwing away blocks that have parity errors -

```
abcde,CP60,MT2.
ACCOUNT,name,group,accno.
ATTACH,TAPEIN.
TAPEIN,T1,UL,MT,RT=U,VSN=12345X.
ATTACH,TAPEOUT.
TAPEOUT,T2,UL,MT,RT=U,VSN=67890Y.
CCOPY,EOI,T1,T2,I,EO=D.
end-of-information
```

C) To make a copy of an unlabelled raw data tape, changing it from 7-track to 9-track, and throwing away any data containing parity errors -

```
abcde,CP60,MT1,NT1.
ACCOUNT,name,group,accno.
ATTACH,TAPEIN.
TAPEIN,T1,UL,MT,RT=U,VSN=12345X.
ATTACH,TAPEOUT.
TAPEOUT,T2,UL,RT=U,VSN=67890Y.
CCOPY,EOI,T1,T2,EO=D.
end-of-information
```

7.11 SUMMARY OF POSSIBLE TAPE PROBLEMS

	Single volume				Multi-volume			
	Single file		Multi-file		Single file		Multi-file	
	Lab.	Unlab.	Lab.	Unlab.	Lab.	Unlab.	Lab.	Unlab.
RT=W	1,2	2	1,2	2	1,2,7	2,3,7	1,2,7	2,3,7
RT=U	1	OK	1	OK	1,7	4,8	1,7	4,8
RT=X	OK	OK	no	OK	no	4,8,9	no	4,8,9
RT=S	1	OK	1	OK	1,7	5	1,7	5
RT=F	1	OK	1	OK	1,7	4,8	1,7	4,8
RT=Z	1,6	6	1,6	6	1,6,7	4,5,6,7	1,6,7	4,5,6,7

1. There is a deficiency in the present SCOPE 2.1.2 Record Manager which does not allow the proper creation of a labelled file in multiple job steps if any job step other than the last one is a FORTRAN program.

2. There is a deficiency in the present SCOPE 2.1.2 Record Manager which does not allow the proper creation of a record type W, block type I file in multiple job steps if any job step other than the last one is a FORTRAN program.

3. These files cannot be processed because W records can span blocks, and thus could also be split across two tape reels. The FORTRAN user who must call XTPCH2 to advance across a reel boundary would lose the record spanning the two reels. Multi-volume files for any record type that can span blocks should be labelled.

4. XTPCH2 must be called to invoke reel swapping (staging in the next volume).

5. For RT=S tapes created on the 6000, reel swapping is automatic when reading. Therefore XTPCH2 must not be called. For RT=S tapes created on the 7600 note 3 applies.

6. Since a file with RT=Z is treated like one with RT=S the same restrictions apply.

7. A temporary 7600 disk file (FIND) cannot be created with multi-volume staging, except as in note 8.

8. A temporary 7600 disk file (FIND) can be created with partial multi-volume staging, but only by using the FILE, FIND and STAGE cards and by specifying the SF parameter on the STAGE card.

9. Odd length records on a 9-track tape cannot be read. If maximum record length is less than or equal to 512 words, use RT=U instead.

7.12 RECOMMENDATIONS

There is unfortunately no single tape format that is suitable for all applications. In particular the interchange of tapes between the 7600 system and other computers requires careful consideration of the capabilities of both the 7600 and the other computer system. Users planning to interchange tapes between systems are strongly recommended to conduct a pilot test with their proposed format, before writing several tapes or hundreds of tapes which may then prove difficult or impossible to read. These guide lines attempt to cover the most common requirements. Specific cases should be discussed with the PEO.

Tapes should be dedicated to 7-track or 9-track use, and not switched from one to the other. 9-track tape reels should carry a 9 sticker. Operators will drop any job requesting a 9-track tape that does not have a 9 sticker or any job requesting a 7-track tape which does have a 9 sticker.

For both reliability and data capacity, 9-track tapes at 1600 BPI density are to be preferred to other track formats and densities. The preferred density for 7-track tapes is 800 BPI. Whenever possible tapes should be labelled with ANSI labels. Tape labels provide good protection against operator and user error. Users are nonetheless still advised to use the yellow stickers DO NOT WRITE ON THIS TAPE, obtainable from the computer receptionist to provide additional protection for valuable data. The default character code for 9-track tape labels at CERN is EBCDIC, for compatibility with IBM.

The choice of record format depends very much on the application. The Chapter on record types lists the formats likely to be of interest to CERN users. Points to watch include-

1) For record types other than W or S, write only one file per labelled tape reel. This applies in particular to record type U, which is recommended for tapes for interchange with outside computers. When reading binary tapes in a FORTRAN program, the use of the X- PACKAGE (CERN Program Library Z200) is very strongly recommended. Users who process multiple input tapes to produce one output tape are advised to consider using a 7600 permanent file to accumulate the output if the number of input tapes processed is greater than five. (See the section on *P files in the Chapter on permanent files).

2) Users who need to copy their tapes or perform other simple manipulation functions, are advised to do so on the front-end MFA 6000. - See the previous Section on 6000 tape-handling.

3) Use TAPEIN/TAPEOUT for access to tapes except for multi-reel files. If the tape is to be read more than once in a short period of time, the ID=userid parameter should be used in order to create or use a temporary disk file of the tape.

8.1 INTRODUCTION TO PERMANENT FILES

Permanent files can be created on the 7600, MFA 6000 (CCP Station) or MFB 6000 (CCQ Station) disks. In practice, the MFA 6000 provides the principal permanent file base, while the MFB 6000 is used mainly for INTERCOM files and the 7600 for temporary copies of heavily used files and tapes. The same rules apply to the filebases on MFA and MFB.

A 7600 job can create and access files on both the MFA and MFB 6000 disks. A permanent file is created using the CATALOG statement and re-accessed using the ATTACH statement. The ST or station parameter on the CATALOG/ATTACH card indicates which file base is to be used. ST=CCP refers to the MFA 6000, ST=CCQ to the MFB 6000. Note that ST=MFA and ST=MFB are also accepted as ST parameters. The 7600 disks cannot normally be used for storing permanent files. Instead, they are used for storing temporary copies of frequently used CCP or CCQ 6000 permanent files and tapes. This is done by using FIND for permanent files and TAPEIN with the ID parameter for tape files. (See Chapter 3)

CCP 6000/CCQ 6000 files

The file space on both the MFA and MFB 6000's is controlled on a divisional basis. The CUAC - Computer User's Advisory Comittee - assigns to each division budgets for permanent file space on the MFA and MFB filebases. These budgets are not to be exceeded. For the larger divisions, group budgets for permanent file space are also assigned by the divisional CUAC representatives. (See list below) the number of PRUs (Physical Record Unit = 64 words) allocated to each group depends on the number of active users in the group and the computer budget. Each group should keep within its budget. Users should contact their Computer Users' Advisory Comittee representative, to find out how much space they are allowed to use.

		Room	Extension	Bleep
DD	H. Grote	31/3-008	4961,4960	* 8-568
DG/DI	G. Lindecker	60/4-008	5886	
EF	H. Wenninger	36/3-016	4097,4270	* 8-361
EP	Mrs L. Griffiths	4/1-072	2660	* 8-968
FI	J. D. Mandica	5/R-	2823	
HS	M. Hoefert	24/1-023	4602,3893	
ISR	B. Zotter	30/6-008	3034	
PE	M. Baboulaz	5/1-018	4484,4126	
PS	P. Skarek	26/1-003	2213	
SB	A. Lecomte	54/1-023	3273	
SPS	C. Iselin	3/1-028	3657	
TH	F. Schrempp	4/2-064	2445	

8.2 MANIPULATING PERMANENT FILES

THE CATALOG STATEMENT

The CATALOG control statement makes a newly created local file permanent. Cataloging consists of making an entry in the permanent file directory for that file. The file is still available for use within the job after it has been cataloged. The form of the CATALOG statement is shown below. The first three parameters should always be given, and in the order shown.

CATALOG,lfn,pfn,ID=userid,options.

lfn	Logical file name. Up to 7 alphanumeric characters (say, TAPE1)
pfn	Permanent file name. Up to 40 alphanumeric characters (e.g. MYFILE)
ID=userid	Owner identifier . Up to 9 alphanumeric characters, obeying the following rules -

ID=DDGGGxxxx user ID

ID=DDGGG group ID

- where -

DD	is the divisional code
GGG	is the group code
xxxx	is an abbreviation of the owner's name.

The limit on the size of files with a group ID may be greater than that of files with a user ID. See the section on permanent file control.

Optional parameters.

ST=st	This parameter controls the residency of the permanent file. Whenever it is absent, the file is CATALOGed on the machine which is executing the job, providing that a 'REQUEST,lfn,*PF.' card has been executed. The REQUEST card is not needed if a 7600 job is CATALOGing a file on MFA or MFB, but station parameter must be set to ST=CCP for MFA, and to ST=CCQ for MFB.
RP=rp	Retention period for the file, rp days. Default is rp=14 days. Files whose retention period has been exceeded are automatically purged from the system. An infinite retention period (RP=999) does NOT

MANIPULATING PERMANENT FILES

GUARANTEE THAT THE FILE WILL ALWAYS BE THERE- see the section on archiving.

CY=cy — Cycle number. Default on first catalog is CY=1. Default on further cycles is 'highest cycle number + 1'. Only 5 cycles of the same permanent file name are allowed.

Passwords — Can be specified in the CATALOG statement. Normally, people using the 7600 only need never specify any. For special reasons of privacy or to obtain multi-read access on the MFA or MFB 6000 machines, it may be useful. See the Section on passwords for details.

THE ATTACH STATEMENT

The ATTACH statement causes a copy of a 6000permanent file to be attached to the job as a local file. If no ST parameter is specified (7600 files), the statement causes the 7600 file itself to be attached as a local file. The parameters of the ATTACH statement are just as for the CATALOG statement, except for the extra parameter LC=1. It implies that the lowest existing cycle of the file should be attached. The form of the ATTACH statement is -

```
ATTACH,lfn,pfn,ID=userid,options.
```

THE REQUEST STATEMENT

Before creating a file which is to be catalogued on the same machine (6000 or 7600) as the job is running,a

```
REQUEST,lfn,*PF.
```

card must be executed. This will grant permission to use a permanent file device. If the file is to be staged from the 7600 to a 6000 for CATALOGing, no REQUEST is needed for the moment.

7600 FILES AND *P FILES

Use of FIND, or TAPEIN with the ID parameter, creates a permanent file on the 7600 disks. These files may be purged whenever the disks are full. There is an automatic purging system whose criteria are

based on a combination of the number of times the file has been used and the length of time since the last access. A facility also exists for creating 7600 permanent files necessary for production jobs which will not be purged. These files are identified by the characters *P as the eighth and ninth character positions of the owner identifier. The file space for these files is budgetted. Budgets may be obtained from the Computer Co-ordinator and vary in size depending on the needs of the individual or group concerned. Until February 77, the Computer Coordinator is R. Bock (bleeper 8-786).

CYCLES

The user may catalog several files under one permanent file name, owner identifier and password set (if this exists). Each file is then known as a CYCLE. Up to 5 cycles may exist per permanent file name. Most commonly, a new cycle is a more recent version of a previous cycle, but as each cycle is a unique file, they may also contain completely separate and independent information. The user can specify any cycle number from 1 to 999 inclusive. If more than one cycle exists, cycle numbers do not have to be consecutive.

MFA and MFB 6000 permanent files created on the 7600 or 6000.

If a cycle number is not specified at catalog time, it is assumed to be one higher than the highest cycle existing (CY=1 if it is an initial catalog). If five cycles already exist, the catalog fails with a diagnostic message. If an attempt is made to catalog an already existing cycle using the CY=cy parameter, the file is cataloged with a cycle number one higher than the highest existing cycle.

If the cycle number is not specified at attach time, the file with the highest cycle number is attached. To attach a lower cycle, the CY=cy parameter must be used to specify the cycle number.

7600 permanent files

These behave in exactly the same way as 6000 permanent files.

PASSWORDS

A permanent file may be cataloged initially with up to 5 passwords. Each password implies one type of access permission, and restricts the use of the file by those who do not know the passwords. The possible passwords are TURNKEY, CONTROL, EXTEND, MODIFY, READ and MULTIREAD and are referred to in the initial CATALOG card as TK=tk, CN=cn,

MANIPULATING PERMANENT FILES

EX=ex, MD=md, RD=rd and XR=xr. Note that XR defines a single password for CONTROL, EXTEND and MODIFY unless they are explicitly defined. Each password may be up to 9 alphanumeric characters. In the ATTACH card they are referred to as a password list, PW=tk,cn,ex,...., in any order, listing as many as are required. If a CN or XR password has been defined on an initial catalog it must be given as PW=cn or PW=xr on the CATALOG statment in order to catalog a higher cycle of the same permanent file.

Users are not recommended to use any passwords at all for files used exclusively on the 7600, unless they require to restrict read access. Then they can use the READ and TURNKEY passwords at initial CATALOG time. If they wish to protect against accidentally purging their own 6000 files, they could use a MULTI-READ password on the initial CATALOG. INTERCOM users, on the other hand, may have problems unless they request MULTI-READ ACCESS on the initial catalog . This is done by specifying the XR parameter on the CATALOG card.

```
CATALOG,lfn,pfn,ID=userid,XR=xr.
```

The file pfn could now be simultaneously ATTACHed by any number of users of INTERCOM, provided they don't specify the password.

Users who wish to make a more sophisticated use of passwords are referred to the SCOPE 2.1.2 Reference Manual.

THE PURGE STATEMENT

A cycle of a permanent file may be deleted from the permanent file directory using the PURGE statement. If passwords have been established for the file, control permission must be granted for that cycle of the permanent file. Only one cycle of a permanent file may be purged at a time. If no cycle is specified, the highest existing cycle is purged. A 6000 permanent file can be purged either from a 6000 job run in either 6000 machine (CCP or CCQ), or from a 7600 job. A 7600 permanent file can be purged only from a 7600 job.

PURGE of a 6000 file from a 6000 job

```
PURGE,trash,myfile,ID=userid,CY=5,ST=CCP.     (or ST=CCQ)
RETURN,trash.
```

If the PURGE command is executed on the same machine as the file resides the ST parameter may be omitted.

The MFA or MFB 6000 INTERCOM purge procedure is the same, except that the control statements are replies to the prompt COMMAND-.

MANIPULATING PERMANENT FILES

PURGE of a 6000 file from a 7600 job

This is exactly the same as for the 6000 job except that the ST parameter must always be specified.

```
PURGE,trash,myfile,ID=userid,CY=4,ST=CCP.     (or ST=CCQ)
RETURN,trash.
```

PURGE of a 7600 file from a 7600 job

```
ATTACH,trash,myfile,ID=userid,CY=3.
PURGE,trash.
RETURN,trash.
```

Note that the option 'PU=ONLY' in the FIND statement, which purges the 7600 copy of a 6000 file, can also purge a 7600 file.

THE RENAME STATEMENT

```
RENAME,lfn,pl=a,..,pn=x.
```

The RENAME statement can only be used in a MFA or MFB 6000 job and can therefore only be used to rename a parameter of a MFA or MFB 6000 file. Changes can be made to the permanent file name, cycle number, passwords, retention period and owner identifier. Changes involving the cycle number or retention period of a file apply only to the cycle which is attached with logical file name lfn. Changes involving the permanent file name, passwords or owner identifier APPLY TO ALL CYCLES of the permanent file, but the RENAME fails if any of the cycles is an archived file. (This cycle should first be reinstalled by an ATTACH statement - see section below on ARCHIVED files). The parameter list in the RENAME statement is a list of parameters whose values are to be changed. The permanent file to be renamed must be first attached and if passwords exist for the file, they must appear in the password list of the ATTACH statement.

Example of MFA or MFB 6000 RENAME command

The cycle number is changed from CY=6 to CY=3, and the retention period altered to 120 days. It is assumed that the initial CATALOG specified XR=ALL to permit multi-read access to INTERCOM users.

```
ATTACH,lfn,myfile,ID=userid,PW=ALL,CY=6.
RENAME,lfn,CY=3,RP=120.
RETURN,lfn.
```

MANIPULATING PERMANENT FILES

AUDIT AND AUDLIST

This AUDIT utility allows users to obtain information concerning their permanent files. An AUDIT of MFA or MFB 6000 permanent files may only be run on the MFA or MFB 6000, and an AUDIT of 7600 files only on the 7600. The following example produces an AUDIT of 6000 (MFA or MFB depending upon where it is run)

```
abcde,CP60.
ACCOUNT,name,group,accno.
AUDIT,ID=userid,AI=P.
end-of-information
```

The above example lists all files with ID=userid giving the cycle number, size, date last altered and expiration date. The size is given in PRU -Physical Record Units- of 64 words. Space on the disks is normally allocated in blocks of 56 PRU, except for EDITOR files which are allocated in 14 PRU blocks. By changing the parameter AI=P (audit information=partial) to AI=F (full) all possible information on the files is printed.

AUDIT may also be used from INTERCOM simply by entering the command

```
AUDIT,ID=userid,AI=P
```

A SORTED AUDIT can also be displayed by using the AUDLIST utility on the 6000 machines. It gives the status of the permanent files of a division, a group or an individual user at the time of the last off-line permanent control, normally early that morning.

It has the form -

AUDLIST,dd	to list all files of division dd sorted by group and user
AUDLIST,ddggg	to list all files of group ggg of division dd sorted by owner
AUDLIST,ddgggxxxx	to list all files with identifier ddgggxxxx

The normal output from AUDLIST gives, in addition to that given by a partial AUDIT, the number of the corresponding dump tape.

AUDLIST may also be used from an INTERCOM terminal in which case it gives a short (80 chars/line) output. In order to get a full output from a terminal enter

```
AUDLIST,userid,T
```

MANIPULATING PERMANENT FILES

ARCHIVING FILES

Files not used for 60 days are ARCHIVED, and their length as shown on an AUDIT is zero. They may still be attached in a job for a further 300 days, as the operators are automatically requested to mount the dump tape containing the archived file and it is then reloaded automatically. There is then no indication that the file was ever archived. Since this operation always takes some time, users are advised not to request the reloading of archived files from Job Enquiry Terminals. After 300 days on the archive tape, if un-accessed, the file is permanently lost.

The full permanent file audit of all MFA and MFB 6000 files is available on microfiche at the RIOS stations and also at the Computer Centre. Users who require information on the history of their files since December 1974 should consult the microfiche copies held at the Computer Centre.

INTEGRITY

Newly created MFA/MFB 6000 permanent files are dumped to tape once a day for safety reasons. A full dump of the entire MFA/MFB 6000 file base is made each weekend. 7600 files are never dumped and can be lost due to disk hardware problems or 7600 deadstarts. Users of *P files should therefore not allow too much valuable output to be collected on the 7600 disks before copying it onto tape.

Note on CORRUPTED permanent files

If, exceptionally, an MFA/MFB file is corrupted or lost it is possible to reload it from tape by completing a special request form (available at RIOS) and sending it to the operations supervisor. Reloading is normally done overnight to avoid disturbance of the time sharing service.

8.3 PERMANENT FILE CONTROL

MFA and MFB file bases are subject to the same rules.

Permanent files are partially controlled on-line, so that

CATALOG of a file with an illegal identifier will be refused. (See CATALOG statement for rules for identifier).

CATALOG of any file which is greater than 4000 PRUs long will be refused unless it has a privileged group identifier which allows a greater maximum length. To get a privileged group identifier apply to your divisional representative - see Section 8.1.

A permanent file control program is run daily, to bring each division within its budget.

All expired files and files with illegal identifiers will be purged.

Any division which exceeds its permanent file budget will be brought back within budget by having files RECATALOGUED as -

PFN = <original identifier><original PFN>
ID = TEMP
CY = orininal cycle
RETENTION = 60 days

The files to be recatalogued are selected by an algorithà based on length and days since last access.
(In the case of divisions DD and EP only those groups exceeding their budget will have files RECATALOGUED).

Any files RECATALOGUED with ID=TEMP will be archive dumped after a a few days and PURGED 60 days after they were RECATALOGUED.

Any files not accessed for 60 days will be archive dumped, and then they will be PURGED if they are not accessed within a further 300 days.

8.4 EXAMPLES OF THE USE OF PERMANENT FILES

In the following jobs, FIND should replace ATTACH if the files are to be frequently accessed. The examples show how a user can catalog his card decks on a permanent file and subsequently compile and load them. He may also catalog the binary version produced by the compiler, and replace routines in this at load time.

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
COPYS,INPUT,cards.
CATALOG,cards,cards,ID=userid,ST=CCP,RP=120.
REWIND,cards.
FTN,I=cards,B=cardlgo.
CATALOG,cardlgo,cardlgo,ID=userid,ST=CCP,RP=120.
LDSET,MAP=B/ZZZZMP.
cardlgo.
end-of-section
.
. FORTRAN routines of user
.
end-of-information
```

Now the user has his original decks on the file 'cards,ID=userid' and a compiled version ready to load and execute on the file 'cardlgo, ID=userid'. Since only the first routine of a series found with the same name is loaded, he can now replace routines in his binary version prior to execution (note that no alteration is made to the permanent file cardlgo) -

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,MAN,MAN,ID=OPSYSTEM.
LIBRARY,lib,MAN.
ATTACH,bin,cardlgo,ID=userid,ST=CCP.
FTN,B=newones.
LDSET,MAP=B/ZZZZMP.
LOAD,newones.
LOAD,bin.
EXECUTE.
end-of-section
.
. FORTRAN cards of new routines
.
end-of-information
```

9.1 INTRODUCTION TO SCOPE ERROR DETECTION

The SCOPE system software and the 7000/6000 hardware in conjunction can detect a number of abnormal conditions during a run and cause job abortion with some kind of diagnostic messages. The system usually produces a dump of all the registers (the so called USER EXCHANGE PACKAGE DUMP) and print-outs of the first 64 memory locations + the area where the error was detected. Furthermore, there will usually be a diagnostic message included in the job dayfile, specifying not only the error but also the system routine which detected it. Sometimes there will be an error message printed at the end of the user's output file.

In this Chapter, we describe the errors more commonly encountered and different techniques for tracing them. We insist on using MANTRAP as a very convenient FORTRAN Dump Analyser.

9.2 ERROR TRACING USING MANTRAP.

MANTRAP is a package which analyses the dump of a FORTRAN program and prints what it finds in readable English. It prints out a trace back to the main program giving the values of all the variables in the routines which have been passed through.

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
FTN,L,OPT=1.
FIND,man,MAN,ID=OPSYSTEM.
LIBRARY,*,man.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
.
. FORTRAN cards
.
end-of-section
.
. DATA
.
end-of-information
```

For most applications, the output is self-explanatory but more details can be found in the Section SCOPE ERROR MESSAGES of this Chapter and more advanced MANTRAP techniques in the following section.

When using MANTRAP with a more complicated job the following points should be noted.

A) The card LDSET,MAP=B/ZZZZMP. must be the first card of the loading sequence - e.g.

```
LDSET,MAP=B/ZZZZMP.
LOAD,mybin.
LOAD,hisbin.
EXECUTE.
```

B) The values of local variables can only be given for those routines which have been compiled in this job. For other routines only trace-back information is available.

ERROR TRACING USING MANTRAP.

C) When using the FTN 4.6 compiler, the U parameter must appear on the FTN card and the TS option must not be requested (it is not compatible with MANTRAP). A special version of MANTRAP is also necessary. It is stored in the Library file NEWMAN,ID=OPSYSTEM - e.g.

```
FIND,new,NEWPRODUCTS,ID=OPSYSTEM.
LIBRARY,new.
FTN,L,U.
FIND,lib,7600LIBRARY,ID=PROGLIB.
FIND,man,NEWMAN,ID=OPSYSTEM.
LIBRARY,man,lib,new.
LDSET,MAP=B/ZZZZMP.
LGO.
end-of-section
.
. FORTRAN program
.
end-of-information
```

OUTPUT OF ARRAY ELEMENTS USING DARRAY

To get a detailed list of the contents of array elements when the program aborts, the routine DARRAY should be called. Normally, DARRAY will be called only once (before executable statements of the program), but it can be called any number of times at any stage of program execution, if required. Later calls will cancel earlier ones.

DARRAY is called as follows -

```
CALL DARRAY(NLINES,I,J,K)
```

NLINES = number of lines of output to be allocated for array elements, with an absolute maximum of 2000 lines. If more than 2000 lines are requested NLINES will be assumed to be ZERO.

I,J,K = maximum values of the i,j,k subscripts for printing purposes.

A call to DARRAY causes the requested elements of ALL arrays to be printed. It is not possible to restrict printing to some specific arrays only. Arrays are printed 5 elements per line with the first subscript varying most rapidly.

ERROR TRACING USING MANTRAP.

CALLING MANTRAP DELIBERATELY

It can sometimes be useful to force a program to abort and produce a dump, for example when a logical error is suspected or detected by the program. This is done simply by

```
CALL MANTRAP
```

FAILURE OF MANTRAP

If MANTRAP fails to give any output the beginning of the user program has probably been overwritten. In that case, the DEBUG facility might be useful.

9.3 THE SYMBOLIC REFERENCE MAP

The R parameter of the FTN card requests a Symbolic Reference Map which gets printed immediately after the FORTRAN listing (see example next page). All addresses given are relative to the starting point of the subprogram. When the program gets loaded all relative addresses will be changed to absolute addresses through addition of the so called relocation constant or the address in memory where the subprogram starts (see LOADING MAP).

The list of VARIABLES gives the relative addresses of simple variables or of the first element of arrays. A name in the column RELOCATION refers to a labelled COMMON, blank COMMON (/ /) or indicates that the variable is a formal parameter (F.P.) to the subprogram. The list of STATEMENT LABELS gives the relative addresses of (some of) the FORTRAN statements. Note that a zero in the left column only means that the assignment is not specified. Details of the Symbolic Reference Map are given in the FORTRAN Extended Reference Manual.

```
       SYMBOLIC REFERENCE MAP

ENTRY POINTS
    2  SUB1

VARIABLES  SN  TYPE     RELOCATION
152  I        INTEGER                162  II       INTEGER  ARRAY
416  IND      INTEGER  ARRAY          145  INIT     INTEGER
300  IR       INTEGER  ARRAY            0  IX       INTEGER          F.P.
153  J        INTEGER                332  JJ       INTEGER  ARRAY
161  K        INTEGER                  0  KK       INTEGER  ARRAY   / /
160  P        REAL                   154  XL       REAL
155  XR       REAL

EXTERNALS         TYPE     ARGS
     IRNDM      INTEGER       1                     RNDM     REAL     1
     SORTZV                   6

INLINE FUNCTIONS TYPE     ARGS
       MOD      INTEGER       2  INTRIN

STATEMENTS LABELS
    0  1        INACTIVE              16  2                     45  3

COMMON BLOCKS  LENGTH
      / /          52

STATISTICS
  PROGRAM LENGTH     433B      283
    BLANK COMMON      64B       52
```

9.4 THE LOADING MAP

By including the control card -

MAP,PART.

before the LGO card or loading sequence of a FORTRAN job, the loader produces a map of the core lay-out, i.e. how core is assigned to program units and COMMON blocks during the execution of a job (see example next page).

In column BLOCK, all SUBPROGRAMS and COMMON blocks are listed. Column ADDRESS gives the octal start address of each SUBPROGRAM and COMMON block. In a FORTRAN program the memory location at the start address contains the program name in DISPLAY code and the address of the ENTRY/EXIT point. Column LENGTH specifies in octal the space occupied in core by the SUBPROGRAM or COMMON block.

The system will transfer control to the main program at the address specified in the line PROGRAM WILL BE ENTERED AT ... This address is called the TRANSFER POINT of the job.

Small Core Memory (SCM) locations 0 through 77B are used by the system. Starting in location 100B the loader places program units from whatever file(s) have been given to it. Each program unit is preceded by any not previously loaded labelled COMMON blocks mentioned in the program unit. Next follow program units from any library(ies) specified and subprograms from the FORTRAN Library. Finally core is assigned to blank COMMON (/ /).

More information about the Loading Map can be found in the Loader Reference Manual.

Below is an example of a Loading Map. /OCT/ and /DATA/ are labelled COMMON blocks, mentioned for the first time in the main program GUIDE. SUB1 to PRSTAT are the user's own FORTRAN subroutines, while IUCOMP to UZERO come from the user's declared Libraries. Remaining routines come from the FORTRAN Library and the Scope 2.1.2 system.

THE LOADING MAP

LOADING MAP

PROGRAM WILL BE ENTERED AT GUIDE (340) SCM LENGTH 15134 LCM LENGTH 0

BLOCK	ADDRESS	LENGTH
/OCT/	100	1
/DATA/	101	163
GUIDE	264	303
SUB1	567	433
PL1	1222	1631
PL2	3053	1660
PRINT1	4733	564
PRSTAT	5517	224
IUCOMP	5743	11
SORTZV	5754	376
SHFTZF	6352	7
RNDM	6361	32
UZERO	6413	7
ENCODE=	6422	131
FORSYS=	6553	1143
GETFIT=	7716	33
INPC=	7751	243
KODER=	10214	1456
KRAKER=	11672	1551
OUTC=	13443	205
SYSTEM	13650	443
GOTOER=	14313	13
//	14326	606

9.5 THE USER EXCHANGE PACKAGE DUMP

For any fatal error the system provides a dump of the User Exchange Package (UEP) plus the first 100B memory locations and some lines of dump of central memory around the stop address (only if the stop address is within the field length of the job). The UEP itself contains the A, B and X registers (numbered from 0 to 7) + the contents of Small and Large Core Memory at addresses contained in the above mentioned registers (subject to the condition that those addresses are within the field length of the job). The UEP also contains a few other registers, of which P (Program address) and FLS/FLL (Small/Large Core Memory field length) are useful to the FORTRAN programmer.

The P register points to the location in SCM of the next instruction to be executed. This usually means that the error occurred at program address P-1. The FLS/FLL registers contain the SCM/LCM field lengths of the job, rounded upwards to the nearest multiple of 100B, i.e. the amount of core storage allocated to the job.

The A (address) registers hold SCM addresses pointing to the most recently referenced memory locations.

The X (operand) registers contain 60-bit words. Setting any of A1-A5 to a memory address in SCM will cause the corresponding X register to be loaded with the quantity in SCM at that address (however, the X registers may also be loaded independently of the A registers). Setting A6 or A7 to a Memory address causes the contents of the corresponding X register to be transferred to the memory location at that address.

THE USER EXCHANGE PACKAGE DUMP

```
USER EXCHANGE PACKAGE

  P    00 000622 A0 000452 BP 000000     SC(A0)= 0000 0000 0000 0001 5036   SC( P )= 4465 7516 0000 7474 6000
  RAS  00 150006 A1 000722 B1 001003     SC(A1)= 0000 0000 0000 0741        SC(B1)= 0000 0000 0000 0056
  FLS  00 015200 A2 000741 B2 000001     SC(A2)= 0000 0000 0000 0031        SC(B2)= 0000 0000 0000 0000
  PSD  00 060000 A3 000736 B3 000032     SC(A3)= 1720 4000 0000 0000        SC(B3)= 0000 0000 0004 5746
  RAL  00 000000 A4 000743 B4 777661     SC(A4)= 1723 6000 0000 0000        SC(B4)=
  FLL  00 000000 A5 000744 B5 777720     SC(A5)= 1724 4000 0000 0000        SC(B5)=
  NEA  40 146626 A6 001000 B6 000001     SC(A6)= 0000 0000 0000 0046        SC(B6)= 0000 0000 0000 0000
  EEA  13 010460 A7 000741 B7 000624     SC(A7)= 0000 0000 0000 0031        SC(B7)= 5150 0007 4743 0003 1765

  X0   1724 7000 0000 0000 0000          SC(X0)= 0000 0000 0000 0000        LC(X0)=
  X1   0000 0000 0000 0000 0741          SC(X1)= 0000 0000 0000 0031        LC(X1)=
  X2   0000 0000 0000 0000 0030          SC(X2)= 0000 0000 0000 0000        LC(X2)=
  X3   1720 4000 0000 0000 0000          SC(X3)= 0000 0000 0000 0000        LC(X3)=
  X4   1723 6000 0000 0000 0000          SC(X4)= 0000 0000 0000 0000        LC(X4)=
  X5   1724 4000 0000 0000 0000          SC(X5)= 0000 0000 0000 0000        LC(X5)=
  X6   0000 0000 0000 0000 0046          SC(X6)= 0155 5555 5555 5533        LC(X6)=
  X7   1724 7000 0000 0000 0000          SC(X7)= 0000 0000 0000 0000        LC(X7)=
```

9.6 TRACING ERRORS YOURSELF

Below follow descriptions of a number of procedures, useful in error tracing. The examples refer to the previously shown Symbolic Reference Map, Loading Map and User Exchange Package.

FINDING WHERE THE ERROR OCCURRED

The P register points to the address of the next instruction to be executed. The error was therefore detected at memory location P-1. In order to find what FORTRAN statement this corresponds to, do as follows.

Subtract from P-1 the largest address in the Loading Map which is smaller than P. Compare the result, R, with the list of statement labels in the Symbolic Reference Map for the subprogram which starts at the selected address. Find the two statement addresses (left column) that form the smallest interval around R. The error occurred in the section of code between the two corresponding statements (right column).

This, however, is sometimes too imprecise, usually due to a lack of statement numbers or because the list of statement label allocation is incomplete. To improve on the above, find the FORTRAN variables most recently referenced (i.e. indicated by the A-registers, c.f. below) and compare with the FORTRAN code. Should this still not be enough it will be necessary to recompile the program in order to get a full listing of the generated object code (use FTN,L,OL.) or use MANTRAP for post-mortem processing.

Example

P is 622B (User Exchange Package). This means the error occurred in subprogram SUB1 which starts at memory address 567B (Loading Map). Subtract 567B from 622B to obtain the relative address 33B where the error occurred in SUB1. This address is between 16B and 45B (Symbolic Reference Map, statement labels) which means the program stopped somewhere between statements 2 and 3 in subroutine SUB1.

FINDING THE FORTRAN VARIABLES IN USE

Each A-register is likely to point to a FORTRAN variable. To find it, subtract from the A-value the largest address in the Loading Map which is smaller than the A-value. Compare the result, R, with the list of variables in the Symbolic Reference Map of the subprogram

which starts at the address in question. If R is found in the list of relative addresses, then the corresponding (simple) FORTRAN variable is the one we are looking for. Otherwise, subtract from R+1 the largest relative address which is smaller than R (the address of the first element of an array) and convert to decimal - this gives the value of the subscript for the array element.

Examples

A) A2 is 741B (User Exchange Package). From the Loading Map we this is in routine SUB1 which starts at 567B. Subtract 567B from 741B to obtain 152B. From the Symbolic Reference Map, variables, we find that this is the address of the FORTRAN variable I. The current value of I is found from SC(A2) and is 31B or decimal 25 in this example.

B) A6 is 1000B. Subtract 567B from 1000B to obtain 211B. This is not in the list of variables, so we subtract the largest address which is smaller than 211B (162B) from 212B. The result, 30B or 24, shows that A6 points to the FORTRAN variable II(24), whose current value (in the SC(A6)) is 46B or 38.

FINDING WHERE A SUBROUTINE WAS CALLED FROM

Get a memory dump which covers the beginning of the subprogram, using the DMP utility after the LGO card. For example -

```
LGO.
EXIT.
DMP,567,700.
```

For subroutine SUB1 in the Loading Map given before. Find among the first few locations of the subprogram a word of the type 0400AAAAAA0000000000. This is the ENTRY/EXIT point of the subprogram and AAAAAA is the address of the memory location immediately after the one which contains the subprogram call. Subtract from AAAAAA the largest address in the Loading Map which is smaller than AAAAAA - the result is the relative address in the calling program of the most recent call to the subprogram.

9.7 CDC SCOPE ERROR MESSAGES

Below follows an alphabetic list of common CDC Scope error messages. Each message is accompanied by information about what it means and what may have caused the error.

FATAL ERROR 103

A dayfile message which appears when the Record Manager has detected an error (INPUT/OUTPUT). It is always associated with an explanatory message, printed in the OUTPUT listing, after the Loading Map and after any results already printed.

Example

```
FILE TAPE1   RECORD          2 RECORD MANAGER ERROR - 0130

ERROR NUMBER   0103 DETECTED BY IOERR   AT ADDRESS 000023
```

Some of the most common Record Manager Errors are listed below. For further details, or other messages, refer to the Record Manager Reference Manual.

LCM DIRECT RANGE

LCM Direct Range, in the dayfile, indicates that the job has tried to reference an LCM address outside the LCM field length of the job. If the LCM field length (FLL register) is 0 or if the error occurred in one of the FORTRAN I/O-routines (e.g. KRAKER=) then the error is most likely due to an invalid index in a READ/WRITE statement.

OVERFLOW CONDITION

The job is aborted with **OVERFLOW CONDITION** in the dayfile whenever the result of an arithmetic operation is plus or minus infinity or numerically too large. Possible causes are

A) Division by zero

B) Use of an undefined quantity (4000000000000000AAAAAA) in an expression

CDC SCOPE ERROR MESSAGES

C) Use of a variable of wrong type (INTEGER instead of REAL, perhaps)

D) Genuine OVERFLOW (number generated greater than 1.26E+322)

PROGRAM RANGE

PROGRAM RANGE in the dayfile indicates that the program has branched to location 0 or has attempted to execute an illegal instruction. This happens as a consequence of the program overwriting itself, usually because defined DIMENSIONS have been exceeded. (see Debug facility C$ ARRAYS)

RECORD MANAGER ERRORS

RM 31 RT=F/Z and FL=0.

For RT=F and RT=Z the TAPEIN or FILE card must contain the parameter FL=n where n is the number of 6 bit characters per record.

RM 44 Null File Name.

The system finds a zero where it expects to find a logical file name - usually the beginning of the main program has been overwritten with zeros.

RM 71 Input Request, PD=0.

An attempt has been made to READ from a file for which the last operation was a WRITE. It is necessary to REWIND or BACKSPACE the file before it can be read.

RM 130 RT=W Control Word Parity Error.

RM 131 BT=I Control Word Parity Error.

If these errors occur at the beginning of a file - records 0,1,2 - then a FILE card or RT parameter on a TAPEIN card is probably missing or wrong. If they occur in the middle of a file then they could be due to a disk parity or bad read. Try restaging the data - see note below.

RM 132 BT=I Block Sequence Error.

This can be caused by the same things as RM130 and RM131. In addition it can be caused by trying to read as a

multi-volume file tapes which were written separately or by reading the second volume of a multi-volume file either on its own or before the first volume.

RM 142 **Excess Data.**

The record to be written/read was longer than the maximum record length expected. For Record Type U the default maximum record length is 5120 (decimal) 6 bit characters- for longer records the parameter MBL=n where n is the maximum number of 6 bit characters per record, must be put on the TAPEIN or FILE card.

RM 143 **Insufficient Data.**

An attempt has been made to read more words of data than exist on the record currently being read (record length shorter than the one expected. Incorrect FL/MBL on the FILE card).

RM 144 **Invalid Block.**

If the file has RT=U this is probably due to a disk error - try restaging the tape (see note below) . Otherwise the Record or Block Type is probably wrong.

RM 160 **Invalid FIT.**

The File Information Table (FIT), stored by FORTRAN at the beginning of the main program has been overwritten.

RM 173 **Invalid RL/PTL/MBL.**

First check that the Record Type (and Block Type) are correct. If this is the case the problem could be due to a disk error - try restaging the data - see note below.

RM 323 **Label Does Not Satisfy Label Card.**

The value of the L parameter on the TAPEIN or LABEL card does not agree with what is physically on the tape.

Note

If you need to restage a data file and have used the ID parameter with TAPEIN or the FIND card remember to add the PU=YES parameter to the TAPEIN/FIND card in order to purge the old 7600 copy of the data and read the tape.

CDC SCOPE ERROR MESSAGES

SCM DIRECT RANGE

If the P-counter or an A-register is set to a value which is larger than the SCM field length of the job, the job aborts with SCM DIRECT RANGE in the dayfile. Common causes are

A) An array subscript assuming an illegal value (numerically too large - maximum number of elements of an array is 2**17 - 1 = 131071 words - negative, 0 ,or wrong type)

B) Calling an UNSATISFIED EXTERNAL - the P counter is then usually 400000B

C) No DIMENSION specified for an array name - the array name appears as an UNSATISFIED EXTERNAL in the Loading Map

D) Using an incorrect number of arguments when calling a subprogram

TIME LIMIT

The job has exceeded the CPU Time Limit specified for the job. This limit is by default 10B seconds unless otherwise set by the T parameter on the JOB card. The value specified with the T parameter is interpreted as an octal number giving the number of seconds for the sum of all job steps. The CPU-time used for each job step can be estimated from the log in the second column of the dayfile. Possible causes for Time Limit include-

A) The FORTRAN code leads to an infinite loop

B) The upper bound of a loop is undefined

C) A loop increment is zero

D) The job simply needs a larger time limit

10.1 GENERAL

The FORTRAN DEBUG facility allows the user the possibility of checking what is happening as his program is executing. By introducing the D or D=lfn parameter on the FTN card, the computer produces a non optimized (OPT=0) code containing debugging instructions. Using DEBUG increases the compilation time and space requirements, and increases execution space by a minimum of 2500B words. The following message may appear in the dayfile -

DEBUG - MORE CORE NEEDED FOR DEBUG PROCESSING

This appears to give no error, but the DEBUG option is not processed. Inserting an 'RFL,147000.' card would fix the problem. Since a lot of checking is done the execution time of the program will naturally increase. Execution of the compiled program is permitted unless the following compilation errors occur.

A) Errors in specification statements

B) Missing DO-loop terminators

C) Missing FORMAT statement numbers

Assuming that these errors do not occur, execution can continue until end-of-job, or until a fatal error is encountered. Debug information produced during execution is written to a file DEBUG. The debug information should be interspersed with normal program output by equivalencing DEBUG to OUTPUT in the program card, i.e.

```
PROGRAM MESS(INPUT,OUTPUT,DEBUG=OUTPUT)
```

Alternatively, the debug outpuut may be written on a separate file and copied to the end of the normal job output.

10.2 ALLOWED DEBUG COMMANDS

The debug commands have the following format- C$ in cols.1,2 , a continuation character (non-blank, non-zero) if relevant in col.6, and an allowed COMMAND starting in col.7.

```
C$    COMMAND(parameter1,....,parametern)
```

The entries allowed for COMMAND are as follows-

DEBUG	heads a series of grouped debugging commands.
ARRAYS	checks that array elements referenced lie within the address range implied by the DIMENSION statement concerned.
STORES	follows the assignments of values to variables, whether variables satisfy relations with the operators .EQ., .GT. etc., or whether variables are OUT-OF-RANGE or INDEFINITE.
TRACE	follows the flow of control in computed GOTO, or assigned GOTO , arithmetic IF and the true side of logical IF statements.
CALLS	traces 'calls to' and 'returns from' SUBROUTINES .
FUNCS	traces references to FUNCTIONS. The returned function values themselves are also written to the DEBUG output file.
GOTOS	checks the assigned statement label in an assigned GOTO statement against the statement list.
OFF	deactivates debugging commands for following FORTRAN cards.
NOGO	prevents any attempts at execution if any fatal FTN error is present.
AREA	limits the action of groups of DEBUG commands to specified sections of FORTRAN coding (between certain FTN listing line numbers , for example).

The DEBUG statements are regarded as ordinary COMMENT cards when the DEBUG option is not selected. They need not therefore be removed after use.

10.3 WHERE TO PLACE DEBUG COMMANDS

DEBUG commands may be entered in any combination of the following three ways.

EXTERNAL DEBUG DECKS

These must be headed by a C$ DEBUG card. An EXTERNAL DEBUG DECK must precede the first source deck in the job INPUT file. If the DEBUG commands are to be read from any file other than the job INPUT file, they necessarily form an EXTERNAL DEBUG DECK. Such decks are useful for attempting to DEBUG an entire program, or a set of routines named on the C$ DEBUG card (see later).

INTERNAL DEBUG DECKS

Such decks must also be headed by a C$ DEBUG card and immediately follow the PROGRAM, SUBROUTINE or FUNCTION card of the routine to which they apply. Internal decks apply only to the routine in which they occur. They are thus suited for the debugging of specific program sections.

INTERSPERSED DEBUG COMMANDS

These are commands inserted as required into the sequence of FORTRAN cards. They are not headed by a C$ DEBUG card. They only apply to the program unit in which they appear, and they also change the FORTRAN line numbers generated in compilation. An interspersed DEBUG command will activate the DEBUG option it selects for all following lines in the program unit.

10.4 DETAILS OF INDIVIDUAL COMMANDS

```
C$    DEBUG
C$    DEBUG (namel,....,namen)
```

This command must head an external or an internal debug deck. The first form, if it occurs in an external deck, applies the DEBUG commands to the entire set of program units. Use of the second form with an external deck restricts the action of the DEBUG deck to the program units identified by namel,...,namen etc. These may be PROGRAM, SUBROUTINE or FUNCTION units. The first form should be used with an internal deck, which can only act upon the program unit in which it appears. Neither form is needed with an interspersed set of commands.

```
C$    ARRAYS
C$    ARRAYS(arrayl,....,arrayn)
```

This command checks that array elements referenced actually lie within the storage reserved for the array by its DIMENSION declaration. The first form of the command activates checking of all arrays, the second restricts debugging to the specified arrays arrayl, ...,arrayn.

For multi-dimensionnal arrays the individual indices are not checked. Since these arrays are stored with the first index variyng most rapidly, errors in the first indices may be not detected, for example with DIMENSION X(10,5)

```
      Y=X(20,3)      would not be flagged
but   Y=X(3,6)       would.
```

Checking is not performed for references to arrays in INPUT or OUTPUT or ENCODE or DECODE statements. Also, when an illegal reference is made and detected, the reference is still carried out.

```
C$    STORES(parameterl,....,parametern)
```

This command requires parameters. Three types are allowed, i.e.

1. variable name or array name
2. variable name.relational operator.constant
3. variable name.checking operator.

The first form will write a message to the DEBUG file whenever a value is assigned to the variable(s) involved, or to any element of the array(s) involved.

DETAILS OF INDIVIDUAL COMMANDS

The second form will write a message to the DEBUG file whenever the specified relations are satisfied. Relational operators allowed are any of .LT.,.LE.,.EQ.,.NE.,.GE.,.GT. for example, X.EQ.100. would give a debug message in the DEBUG output file whenever X had the value 100.

The last form is used to check whether the specified variables involved are OUT-OF-RANGE or INDEFINITE. Allowed checking operators are- .RANGE. , .INDEF. , .VALID. . These check if the variable is OUT-OF-RANGE, INDEFINITE and either OUT-OF-RANGE or INDEFINITE respectively. Note the full stop after the checking operators.

C$ TRACE(level)

This command traces the transfer of control within program units. Tracing occurs for GOTO , computed or assigned GOTO , arithmetic IF and the true side of logical IF statements. Tracing will occur inside DO loops up to the level specified. If the level requested is zero or C$ TRACE is used with no parameters, no DO loop tracing will be done.

CALL STRACE

The subroutine STRACE may be called anywhere within any program unit. The D parameter need not be used on the FTN card. Debug information tracing the transfer of control to STRACE as far back as the main program level, is written to the file DEBUG. This file must be equivalenced to OUTPUT or otherwise dealt with by the user.

C$ CALLS
C$ CALLS (subroutine1,....,subroutinen)

This command initiates the tracing of 'calls to' and 'returns from' subroutines. Messages are written to the DEBUG output file whenever calls or returns within the program unit being debugged are made. The routine name, level of call and line number in which the call or return was found are given.

The first form of the command traces all calls and 'returns from' all subroutines, while the second form traces 'calls to' and 'returns from' the named subroutines subroutine1,...,subroutinen.

DETAILS OF INDIVIDUAL COMMANDS

```
C$    FUNCS
C$    FUNCS (functionl,.....,functionn)
```

This command is similar to C$ CALLS except that it applies to FUNCTION subprograms only. The value assigned to the function is also written to the DEBUG file.

```
C$    GOTOS
```

This command initiates checking of all assigned GOTO statements to see if the statement number assigned is present in the GOTO statement list. If no match is found,a diagnostic message is written to the DEBUG file. However, if the statement does exist but is not in the GOTO list, control will still be transferred to that statement.

```
C$    OFF
C$    OFF (optionl,....,optionn)
```

This command in its first form de-activates all currently active DEBUG commands for all the following FORTRAN statements of the program unit in which the C$ OFF occurs. In its second form,only the specified debug instructions optionl,...,optionn are de-activated. C$ OFF commands are relevant only to interspersed debug commands.

```
C$    NOGO
```

This command, which has no parameters, is used to inhibit execution of the program if compiler errors are present. It is not affected by the C$ OFF or C$ AREA commands.

```
C$    AREA boundsl,.....,boundsn
C$    AREA/namel/boundsl,.,boundsn,/name2/boundsl,.,boundsn,.etc.
```

This command restricts the area of code within a program unit which will be debugged. With an internal debug deck, only the first form applies.

The second form applies only to an external debug deck and namel, ...,namen specify the PROGRAM, SUBROUTINE or FUNCTION units to which the deck commands apply. The parameters boundsl,....,boundsn denote the regions of the FORTRAN coding to which the debugging commands are going to be applied. Forms permitted for these bounds parameters are-

DETAILS OF INDIVIDUAL COMMANDS

(nl) debug the single line nl in the FTN compiler listing

(nl,n2) debug the lines nl to n2 inclusive. nl must refer to an earlier line of source code than n2.

More than one form is allowed for the numbers nl and n2. They may appear as simple numbers **nnnn** and are then interpreted as FORTRAN statement numbers. They may appear as **L**nnnn , and be interpreted as FORTRAN listing line numbers. According to the FORTRAN manual, the form **id.**nnnn is allowed, referring to UPDATE line identifiers.

id must start with an alphabetic character, and no special characters are allowed. However, at present, this form does not work in all cases. Finally, * is allowed instead of nl or n2 or both, where it signifies the first and last lines of the program unit respectively.

10.5 EXAMPLES OF THE USE OF THE COMMANDS

Example 1.

This example shows interspersed C$ ARRAYS and C$ STORES command. The control cards show how to use DEBUG when the program is being read from cards. Example 4 shows the control cards necessary if the program is on an UPDATE file.

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
FTN,D,L.
LGO.
end-of-section
      PROGRAM DERROR(INPUT,OUTPUT,DEBUG=OUTPUT)
      DIMENSION MAPL(4),MULT(10)
      DATA MAPL/1,5,9,13/
C$    ARRAYS
C$    STORES(I)
      I=MAPL(2)-2
      MULT(I)=1
      I=MAPL(4)-2
      MULT(I)=1
      STOP
      END
end-of-information
```

The debug information from running this program would appear as-

```
/DEBUG/  DERROR AT LINE 6- THE NEW VALUE OF THE VARIABLE I IS 3
/DEBUG/  DERROR AT LINE 8- THE NEW VALUE OF THE VARIABLE I IS 11
/DEBUG/  DERROR  AT  LINE  9-  THE SUBSCRIPT VALUE OF 11 IN ARRAY MULT
EXCEEDS DIMENSIONED BOUND OF 10
```

Example 2.

This example shows the use of an external debug deck using C$ CALLS.

```
C$    DEBUG
C$    CALLS
      PROGRAM MAIN(INPUT,OUTPUT,DEBUG=OUTPUT)
      X=1.
      CALL SUB1(X)
      STOP
      END
```

EXAMPLES OF THE USE OF THE COMMANDS

```
      SUBROUTINE SUB1(X)
      X2=2.*X
      CALL SUB2(X2)
      RETURN
      END

      SUBROUTINE SUB2(S)
      PRINT 1,S
 1    FORMAT(1HO,* REACHED SUB2 WITH S=*,F5.2)
      RETURN
      END
```

The debug information from this would, for example, appear as-

```
/DEBUG/ MAIN  AT LINE  3- ROUTINE  SUB1 CALLED AT LEVEL 0
/DEBUG/ SUB1  AT LINE  3- ROUTINE  SUB2 CALLED AT LEVEL 1

REACHED SUB2 WITH S= 2.00
/DEBUG/ SUB1  AT LINE  4- ROUTINE  SUB2 RETURNS TO LEVEL 1
/DEBUG/ MAIN  AT LINE  4- ROUTINE  SUB1 RETURNS TO LEVEL 0
```

Example 3.

If in example 2 the debug option D had been removed from the FTN card and a call to STRACE had been inserted in subroutine SUB2

```
      SUBROUTINE SUB2(S)
      PRINT 1,S
 1    FORMAT(1HO,* REACHED SUB2 WITH S=*,F5.2)
      CALL STRACE
      RETURN
      END
```

output would appear as -

```
REACHED SUB2 WITH S=  2.00
/DEBUG/ SUB2  AT LINE 4- TRACE ROUTINE CALLED
   SUB2 CALLED BY SUB1 AT LINE 3 FROM 1 LEVELS BACK
   SUB1 CALLED BY MAIN AT LINE 5 FROM 2 LEVELS BACK
```

Note that, since the D parameter does not appear on the FTN card, the cards

```
C$    DEBUG
C$    CALLS
```

are treated as COMMENT cards.

EXAMPLES OF THE USE OF THE COMMANDS

Example 4.

If a source program is stored on a UPDATE file there are two possible ways of introducing DEBUG commands. Either they can be inserted into the FORTRAN code using UPDATE or introduced as an external DEBUG deck on the INPUT file.

A) DEBUG commands inserted via UPDATE

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
FIND,OLDPL,pfname,ID=userid.
UPDATE.                 Include file DEBUG on the PROGRAM card
FTN,L,I,D=COMPILE.      Compilation with source code and DEBUG commands
                        on file COMPILE (**)
MAP,PART.
LGO.
end-of-section
*DELETE MAIN.2
      PROGRAM MAIN(INPUT,OUTPUT,TAPE10,DEBUG=OUTPUT)
*INSERT MAIN.7
C$    ARRAYS
*INSERT SUB3.5
C$    ARRAYS
C$    STORES(J,NM,P)
end-of-section
.
. Data for FORTRAN program
.
end-of-information
```

(**) Note that 'FTN,L,I.' defaults to 'FTN,L,I=COMPILE.'

EXAMPLES OF THE USE OF THE COMMANDS

B) DEBUG commands on the INPUT file

```
abcde.
ACCOUNT,name,group,accno.
FIND,lib,7600LIBRARY,ID=PROGLIB.
LIBRARY,lib.
FIND,OLDPL,pfname,ID=userid.
UPDATE.                 Include file DEBUG on the PROGRAM card
FTN,L,I,D.              Compilation with source code on file COMPILE
                        and DEBUG commands on INPUT file
MAP,PART.
LGO.
end-of-section
*DELETE MAIN.2
      PROGRAM MAIN(INPUT,OUTPUT,TAPE10,DEBUG=OUTPUT)
end-of-section
C$    DEBUG
C$    AREA /MAIN/ (*,*)      Following DEBUG commands apply to the
                             whole of the routine MAIN
C$    ARRAYS(X,Y,IX)
C$    AREA /SUB3/ (25,40)    Following DEBUG commands apply to
                             routine SUB3 from label 25 to label 40
C$    ARRAYS
C$    STORES(J,NM,P)
end-of-section
.
. Data for FORTRAN progra m
.
end-of-information
```

11.1 INTRODUCTION

It is possible in a FORTRAN program to regain control after occurence of an error at execution time by making use of the routines SYSTEMC and REPRIV.

SYSTEMC is used to recover from errors detected by FORTRAN e.g. FTN ERR39, square root of negative number.

REPRIV is used for recovery from errors detected by the system e.g. overflow, time limit....

Some CERN applications packages (e.g. SUMX, MARC, NICOLE) have error recovery included. Users who are modifying these programs should take care if they want to add their own error recovery.

Warning

Use of REPRIV is incompatible with the use of the post-abort processor MANTRAP. If the user calls REPRIV, then MANTRAP will fail to work.

11.2 ERRORS DETECTED BY FORTRAN

It is possible to recover from any of the FORTRAN errors by making calls to a routine called SYSTEMC. Supposing we wish to recover from error IERROR - e.g. error 39 square root of a negative number - we need an array IARRAY(6) and a recovery routine FTNERR which is declared as EXTERNAL . The sequence initialising recovery in the main program becomes -

```
PROGRAM ZAP(INPUT,OUTPUT,....)
.
.
DIMENSION IARRAY(6)
EXTERNAL FTNERR
DATA IARRAY/0,0,1,100,0,10/
DATA IERROR/39/
IARRAY(5)=LOCF(FTNERR)
CALL SYSTEMC(IERROR,IARRAY)
.
.
END
```

The names FTNERR and IERROR can, of course, be chosen by the user. The meaning of the elements of IARRAY is as follows-

Element	Meaning	Typical value
1	Set to 1, error is fatal. Zero, non-fatal.	0
2	Print frequency of diagnostic message.	0
3	Print frequency increment. On each occurence of error, the print frequency parameter is augmented by this. Only significant if IARRAY(2)=0. Special value- 0- never list error Special value- 1- always list error Special value- n- list first n occurences only	5
4	Print limit	20
5	Recovery routine address, found with LOCF	LOCF(FTNERR)
6	Traceback limit. Tracing back occurs from the routine detecting the error up to this limit.	10

ERRORS DETECTED BY FORTRAN

If any word is negative in the array in a series of calls to SYSTEMC , the previous value for that word is used. Every recovered error must have a separate SYSTEMC call . This means that if one wants to know what error has occured, the simplest way is to have just one recovery routine, but supply an ENTRY POINT in that routine for each recovered error . The recovery routine itself must call a terminating routine for the job, or, for example, read in another set of data and process a new event.

11.3 ERRORS DETECTED BY THE SYSTEM

It is possible to recover from System-detected errors, such as INDEFINITE CONDITION, TIME LIMIT and those listed below by calling the CERN Program Library routine **REPRIV**. The user must declare an array **JARRAY(n+1)** in his program, where n is the number of different errors from which he wishes to recover. The last element of JARRAY should always be zero. Also he must declare his recovery routine SYSERR as **EXTERNAL**. The names JARRAY and SYSERR can be chosen to suit the user. The initialising sequence becomes-

```
PROGRAM ZAP1(INPUT,OUTPUT,....)
.
.
DIMENSION JARRAY(2)
EXTERNAL SYSERR
DATA JARRAY/900,0/
CALL REPRIV(JARRAY,SYSERR)
.
.
END
```

The error numbers allowed as elements of JARRAY and their associated errors are as follows. The maximum number of permitted recoveries is also indicated. Initialising recovery of any error of an indicated group also initialises recovery of the other members of the group.

Error	Meaning of the error	Maximum recoveries	Group
900 or 0	all errors are recovered	-	-
901	underflow	100	1
902	overflow	100	1
903	indefinite condition	100	1
904	step	1	2
905	breakpoint	1	2
906	program range	1	2
907	SCM direct range	20	3
908	LCM direct range	20	3
909	SCM block range	20	3
910	LCM block range	20	3
911	SCM parity	1	2
912	LCM parity	1	2
913	operator drop	1	4
914	rerun	1	4
915	time limit	1	4
916	mass storage limit	1	4
917	LCM limit exceeded	1	4

Note- error 916 cannot be recovered by a normal FORTRAN program.

ERRORS DETECTED BY THE SYSTEM

The subroutine SYSERR provided by the user should call a routine to restart the program or provide a safe termination of the job for the user. If it does nothing but executes a RETURN statement, the job will be aborted. The error which has been detected by the system is communicated to SYSERR as the first argument of a call, so the user should provide

SUBROUTINE SYSERR(NERROR)

which will give as NERROR the error number. He can then decide what steps to take in his recovery procedure. Note that recovery of an error is deactivated by a call to SYSERR , and to reactivate his recovery the user should call a second routine RPRIV2 from SYSERR to reactivate recovery. Should the user need more information, he can make calls to the following routines from SYSERR -

REPXJP(KARRAY) transmits the EXCHANGE JUMP PACKAGE to the array KARRAY(16), which must be dimensioned as indicated.

REPDMP gives the EXCHANGE JUMP PACKAGE and a dump of RA to RA+100 , as well as P-100 to P+100 .

PDUMP(A,B,F)
words to be
dumped.

For F=4-7, A and B are the beginning and end locations in octal, of the area to be dumped.

F=0,3,4,7	octal dump
F=1,5	real dump
F=2,6	integer dump

ERRORS DETECTED BY THE SYSTEM

AN EXAMPLE OF FORTRAN AND SYSTEM ERROR RECOVERY.

The following example illustrates recovery from FORTRAN and System errors. It is assumed that the program is looping, reading an event from a data tape in routine GETDAT, and analysing it in ANALYSIS. The summary of the job is produced by ENDIT after time limit, too many errors or 10000 events. Note that the recovery procedure for all except time limit is to read a new event, as it is likely to be an unforeseen data condition which causes the errors. Note also that the main program is a dummy- this is to allow ENTRY points in the 'main subroutine', which is a very easy way to restart program runs in a way transparent to the user. Both the recovery routines FTNERR and SYSERR will return control to the system if the error condition is not understood. This avoids the possibility of disaster if you make a programming error in your recovery procedures.....

```
      PROGRAM ZAPNOT(INPUT,OUTPUT,....)
      DIMENSION IARRAY(6),JARRAY(3)
      EXTERNAL SYSERR,ERR39,ERR41
      DATA IARRAY/0,0,0,100,0,10/,JARRAY/902,915,0/
C
C     RECOVER FROM OVERFLOW AND TIME LIMIT
C
      CALL REPRIV(JARRAY,SYSERR)
C
C     RECOVER FROM SQRT(NEGATIVE) AND TAN(ACCURACY LOST)
C
      IARRAY(5)=LOCF(ERR39)
      CALL SYSTEMC(39,IARRAY)
      IARRAY(5)=LOCF(ERR41)
      CALL SYSTEMC(41,IARRAY)

C     CALL MAIN PROGRAM, DISGUISED AS SUBROUTINE
C
      CALL MAINP
      END

      SUBROUTINE MAINP
      .
      .
   10 IEVENT=IEVENT+1
      ENTRY GETDAT1
      CALL GETDAT
      .
      .
      CALL ANALYS
      .
      .
      IF(IEVENT.LT.10000) GO TO 10
      ENTRY ENDIT1
```

```
      CALL ENDIT
      END

      SUBROUTINE SYSERR(IERROR)
      DATA ITOTAL/0/
      ITOTAL=ITOTAL+1
      CALL RPRIV2
C
C     IF MORE THAN 20 ERRORS OR TIME LIMIT, END RUN
C
      IF(IERROR.EQ.915.OR.ITOTAL.GT.20) CALL ENDIT1
C
C     ALL OTHER ERRORS, READ A NEW EVENT IN
C
      PRINT 10,IERROR,ITOTAL
   10 FORMAT(*  FATAL SYSTEM ERROR *,I3,* TOTAL IS *,I3)
      CALL GETDAT1
      END

      SUBROUTINE FTNERR
      DATA IERR39,IERR41/0,0/
C
C     SQRT OF NEGATIVE NUMBER
C
      ENTRY ERR39
      IERR39=IERR39+1
      PRINT 10,IERR39
   10 FORMAT(*  ERROR 39- TOTAL *,I3)
      IF(IERR39.GT.20) CALL ENDIT1
      CALL GETDAT1
C
C     TAN, ARGUMENT TOO LARGE
C
      ENTRY ERR41
      IERR41=IERR41+1
      PRINT 20,IERR41
   20 FORMAT(*  ERROR 41- TOTAL *,I3)
      IF(IERR41.GT.20) CALL ENDIT1
      CALL GETDAT1
      END

      SUBROUTINE ENDIT
C
C     RUN SUMMARY
C
      STOP
      END
```

12.1 FORTRAN FORMAT STATEMENTS.

In this Chapter we describe both features of the more common FORMAT specifications that can cause confusion and surprise, and also additional formatting features now available with the FTN compiler. It should be noted however that these later features are not standard ANSI FORTRAN and so may not be available on other computers.

I FORMAT- Iw, Iw.z

This FORMAT is used for DECIMAL INTEGER constants. w is a decimal integer constant designating the total number of characters in the field, including sign and blanks. z is a decimal integer constant designating the minimum number of digits output. It is ignored on input. Leading zeroes are generated when the output requires less than z digits. If z=0 , a zero value will produce all blanks. If z=w , no blanks will occur when the field is positive. Not specifying z is taken as z=1 . On input, blanks are interpreted as zeroes. On output, if the field is too short, all asterisks fill the field.

Note - numbers in the range -2**59+1 to +2**59-1 can be output correctly, but 2**48-1 is the largest used by the computer in multiplication and division.

E FORMAT- Ew.d

This FORMAT is used for REAL numbers which also have a decimal exponent. w once again is the field width including signs, decimal point, E, the exponent and any blanks. d is the number of digits to the right of the decimal point. Normally, w is greater or equal to (d+6). If the field for output is too short, all asterisks fill the field. If it is too long, the number is right justified with blanks on the left. Note that if the quantity to be output is indefinite then an I will be output. If the quantity is out of range, then an R will be output. If integers are output under an E format, the results are quite unpredictable since the internal FORMAT of integers and floating point numbers differ. Most often, a very small value or 0.0 will be output. For input there are several extra points to note-

A) If a decimal point is supplied, it overrides d

B) Leading blanks are ignored

C) An entirely blank field is taken as -0

D) The exponent must be right justified in the field specified by w

The range of permissible numbers for E FORMAT is 3.13E-294 to 1.26E+322 approximately, in absolute value.

FORTRAN FORMAT STATEMENTS.

F FORMAT- Fw.d

The F specification concerns REAL numbers without a decimal exponent. d is the number of digits to the right of the decimal point. Normally, w is greater than or equal to (d+2). On output, when d is zero, only digits to the left of the decimal point are printed. If the field is longer than required, the number is right justified with blanks on the left. If the value is indefinite, I is printed. If it is out of range , an R is printed. On input, Fw.d is treated just as Ew.d.

O FORMAT- Ow, Ow.d

The O specification indicates OCTAL input or output. On input, only the Ow form is allowed, with w less than or equal to 20. A sign may precede the first octal digit, blanks are allowed and interpreted as zeroes. The item being read should be an integer in the user program. On output, if w<20, only the rightmost digits are output. If w>20, the number is right justified with blanks on the left. Leading zeroes are not suppressed.

Note - if d is specified, at least d digits will be output. Leading zeroes are supressed and a minus sign appears for negative numbers. If the number cannot be output in w digits, all asterisks fill the field.

Z FORMAT- Zw

HEXADECIMAL values are converted under the Z specification. w is the field width, and leading blanks or sign may occur. A maximum of 15 hexadecimal digits is allowed. On output, if w>15, leading blanks will appear. Remember that a valid hexadecimal quantity contains only 0,1,..,9, and A,B,..,F.

A and R formats- Aw,Rw

These formats are used for input/output of CHARACTER information. w indicates the total number of characters in the field. Up to 10 characters of 6 bits can be stored per 60 bit word. On input, if w<10, A FORMAT stores the characters left justified in the word and fills the remainder with blanks. R FORMAT stores the characters right justified, and fills the remainder with binary zeroes. On input, if w>10, the rightmost 10 characters are stored by A FORMAT and the remainder ignored. On output, A FORMAT outputs the leftmost w characters in the word. If w>10, the characters are right justified with blanks on the left. R outputs the rightmost characters.

L FORMAT- Lw

This FORMAT is used for input/output of LOGICAL variables. On output, .TRUE. and .FALSE. become a right justified T and F in the w character field, respectively. On input, if the first non-blank character in the w character field is T or F , the variable is set to

.TRUE. or .FALSE. respectively. If neither T nor F is the first non-blank character encountered, a diagnostic is produced. An all blank field produces a value of .FALSE.

X FORMAT- nX

This FORMAT is used only to SKIP characters in an input or output line. On output, any character positions not filled previously in the record generation will be set to blank. n is the number of characters to be skipped. 0X is ignored, and X is assumed to be 1X . A minus sign may precede nX and will back up n characters. However, it will not back up past column 1.

H FORMAT- nH

This specification can be used to read n HOLLERITH characters into an existing H field within a FORMAT statement. For example , with an input data card such as-

```
THIS IS A VARIABLE HEADING
```

one can read the data into a FORMAT with the H specification-

```
    READ 10
 10 FORMAT(27H
```

Any subsequent PRINT 10 or WRITE(LUN,10) will output the data card heading. On output, nH in a FORMAT will output the next n characters in the format.

T FORMAT- Tn

This is used for COLUMN selection on input or output. Control skips forwards or backwards until column n is reached, when the next FORMAT specification is processed. If n>80, the pointer is moved to that column but the next specification will only read blanks. The order of reading or writing of elements need not be the same as the card or page data. For example one can read from column 50, by using T50 , and then from column 6 on the same data card by using T6.

FORTRAN FORMAT STATEMENTS.

FREE FORMAT INPUT/OUTPUT- *

List directed output

```
WRITE (LUN,*) iolist      to write to file LUN
PRINT *, iolist           to write to file OUTPUT
```

In addition to its conventional use for free-form output, this is a convenient way of putting debug printing in a program since it requires no format statement. The output values are real or integer, depending on the variable type.

```
PROGRAM TEST(OUTPUT)
X=3.6
PRINT *,5HSQRT(,X,6H) IS =, SQRT(X)
END
```

List directed read

```
READ (LUN,*) iolist      to read from file LUN
READ *, iolist           to read from file INPUT
```

The input data consists of a string of values separated by a blank, a comma, a slash or a line boundary such as end-of-record or end-of-card. The separators may be preceded or followed by any number of blanks but there must be no imbedded blanks in the data items.

The separator 'slash' terminates the current READ and sets the remaining list elements to undefined.

For more details, see the FORTRAN Reference Manual (60497800x).

13.1 OCTAL REPRESENTATION

SOME NON-STANDARD FLOATING POINT OCTAL WORD PATTERNS

1777000000000000000000	INDEFINITE, result of operations such as 0/0, 0*infinity or calling of certain mathematical subprograms with abnormal arguments.
3777000000000000000000	plus infinity.
4000000000000AAAAAA	minus infinity, but if this word is at memory address AAAAAA then it was set by the loader and has remained UNDEFINED throughout the job.

OCTAL/DECIMAL CONVERSION

Octal	decimal	octal	decimal	octal	decimal	octal	decimal
100	64	1000	512	10000	4096	100000	32768
200	128	2000	1024	20000	8192	200000	65536
300	192	3000	1536	30000	12288	300000	98304
400	256	4000	2048	40000	16384	400000	131072
500	320	5000	2560	50000	20480	500000	163840
600	384	6000	3072	60000	24576	600000	196608
700	448	7000	3584	70000	28672	700000	229376

The absolute value of a negative octal number is found by complementing it, i.e. replacing each octal digit by its difference from 7. An octal number is negative if bit 59 - most significant, leftmost bit - is set when the number is loaded into a memory word. MANTRAP interprets these octal representations into normal numbers or INDEFINITE, etc...

OCTAL REPRESENTATION OF FLOATING POINT NUMBERS

13.2 OCTAL REPRESENTATION OF FLOATING POINT NUMBERS

+0	0000000000000000000	-0	7777777777777777777
.1E-04	1677517426542161503	-.1E-04	6100260351235616274
.2E-04	1700517426542161503	-.2E-04	6077260351235616274
.3E-04	1700767242023252434	-.3E-04	6077010535754525343
.4E-04	1701517426542161503	-.4E-04	6076260351235616274
.5E-04	1701643334272616102	-.5E-04	6076134443505161675
.6E-04	1701767242023252434	-.6E-04	6076010535754525343
.7E-04	1702445463665743373	-.7E-04	6075332314112034404
.8E-04	1702517426542161503	-.8E-04	6075260351235616274
.9E-04	1702571371416377725	-.9E-04	6075206406361400052
.1E-03	1702643334272616102	-.1E-03	6075134443505161675
.2E-03	1703643334272616102	-.2E-03	6074134443505161675
.3E-03	1704472445214052461	-.3E-03	6073305332563725316
.4E-03	1704643334272616102	-.4E-03	6073134443505161675
.5E-03	1705406111564570651	-.5E-03	6072371666213207126
.6E-03	1705472445214052461	-.6E-03	6072305332563725316
.7E-03	1705557000643334272	-.7E-03	6072220777134443505
.8E-03	1705643334272616102	-.8E-03	6072134443505161675
.9E-03	1705727667722077712	-.9E-03	6072050110055700065
.1E-02	1706406111564570651	-.1E-02	6071371666213207126
.2E-02	1707406111564570651	-.2E-02	6070371666213207126
.3E-02	1707611156457065176	-.3E-02	6070166621320712601
.4E-02	1710406111564570651	-.4E-02	6067371666213207126
.5E-02	1710507534121727024	-.5E-02	6067270243656050753
.6E-02	1710611156457065176	-.6E-02	6067166621320712601
.7E-02	1710712601014223351	-.7E-02	6067065176763554426
.8E-02	1711406111564570651	-.8E-02	6066371666213207126
.9E-02	1711446722743247736	-.9E-02	6066331055034530041
.1E-01	1711507534121727024	-.1E-01	6066270243656050753
.2E-01	1712507534121727024	-.2E-01	6065270243656050753
.3E-01	1712753412172702436	-.3E-01	6065024365605075341
.4E-01	1713507534121727024	-.4E-01	6064270243656050753
.5E-01	1713631463146314631	-.5E-01	6064146314631463146
.6E-01	1713753412172702436	-.6E-01	6064024365605075341
.7E-01	1714436560507534121	-.7E-01	6063341217270243656
.8E-01	1714507534121727024	-.8E-01	6063270243656050753
.9E-01	1714560507534121726	-.9E-01	6063217270243656051

OCTAL REPRESENTATION OF FLOATING POINT NUMBERS

.1E+00	1714631463146314631 4	-.1E+00	6063146314631463
.2E+00	1715631463146314631 4	-.2E+00	6062146314631463
.3E+00	1716463146314631463 1	-.3E+00	6061314631463146
.4E+00	1716631463146314631 4	-.4E+00	6061146314631463
.5E+00	1716777777777777777 7	-.5E+00	6061000000000000000
.6E+00	1717463146314631463 1	-.6E+00	6060314631463146
.7E+00	1717546314631463146 2	-.7E+00	6060231463146315
.8E+00	1717631463146314631 4	-.8E+00	6060146314631463
.9E+00	1717714631463146314 5	-.9E+00	6060063146314632
.1E+01	1720400000000000000 0	-.1E+01	6057377777777777777
.2E+01	1721400000000000000 0	-.2E+01	6056377777777777777
.3E+01	1721600000000000000 0	-.3E+01	6056177777777777777
.4E+01	1722400000000000000 0	-.4E+01	6055377777777777777
.5E+01	1722500000000000000 0	-.5E+01	6055277777777777777
.6E+01	1722600000000000000 0	-.6E+01	6055177777777777777
.7E+01	1722700000000000000 0	-.7E+01	6055077777777777777
.8E+01	1723400000000000000 0	-.8E+01	6054377777777777777
.9E+01	1723440000000000000 0	-.9E+01	6054337777777777777
.1E+02	1723500000000000000 0	-.1E+02	6054277777777777777
.2E+02	1724500000000000000 0	-.2E+02	6053277777777777777
.3E+02	1724740000000000000 0	-.3E+02	6053037777777777777
.4E+02	1725500000000000000 0	-.4E+02	6052277777777777777
.5E+02	1725620000000000000 0	-.5E+02	6052157777777777777
.6E+02	1725740000000000000 0	-.6E+02	6052037777777777777
.7E+02	1726430000000000000 0	-.7E+02	6051347777777777777
.8E+02	1726500000000000000 0	-.8E+02	6051277777777777777
.9E+02	1726550000000000000 0	-.9E+02	6051227777777777777
.1E+03	1726620000000000000 0	-.1E+03	6051157777777777777
.2E+03	1727620000000000000 0	-.2E+03	6050157777777777777
.3E+03	1730454000000000000 0	-.3E+03	6047323777777777777
.4E+03	1730620000000000000 0	-.4E+03	6047157777777777777
.5E+03	1730764000000000000 0	-.5E+03	6047013777777777777
.6E+03	1731454000000000000 0	-.6E+03	6046323777777777777
.7E+03	1731536000000000000 0	-.7E+03	6046241777777777777
.8E+03	1731620000000000000 0	-.8E+03	6046157777777777777
.9E+03	1731702000000000000 0	-.9E+03	6046075777777777777

OCTAL REPRESENTATION OF FLOATING POINT NUMBERS

.1E+04	1731764000000000000	-.1E+04	6046013777777777777
.2E+04	1732764000000000000	-.2E+04	6045013777777777777
.3E+04	1733567000000000000	-.3E+04	6044210777777777777
.4E+04	1733764000000000000	-.4E+04	6044013777777777777
.5E+04	1734470400000000000	-.5E+04	6043307377777777777
.6E+04	1734567000000000000	-.6E+04	6043210777777777777
.7E+04	1734665400000000000	-.7E+04	6043112377777777777
.8E+04	1734764000000000000	-.8E+04	6043013777777777777
.9E+04	1735431200000000000	-.9E+04	6042346577777777777
.1E+05	1735470400000000000	-.1E+05	6042307377777777777
.2E+05	1736470400000000000	-.2E+05	6041307377777777777
.3E+05	1736724600000000000	-.3E+05	6041053177777777777
.4E+05	1737470400000000000	-.4E+05	6040307377777777777
.5E+05	1737606500000000000	-.5E+05	6040171277777777777
.6E+05	1737724600000000000	-.6E+05	6040053177777777777
.7E+05	1740421340000000000	-.7E+05	6037356437777777777
.8E+05	1740470400000000000	-.8E+05	6037307377777777777
.9E+05	1740537440000000000	-.9E+05	6037240337777777777
.1E+06	1740606500000000000	-.1E+06	6037171277777777777
.2E+06	1741606500000000000	-.2E+06	6036171277777777777
.3E+06	1742444760000000000	-.3E+06	6035333017777777777
.4E+06	1742606500000000000	-.4E+06	6035171277777777777
.5E+06	1742750220000000000	-.5E+06	6035027557777777777
.6E+06	1743444760000000000	-.6E+06	6034333017777777777
.7E+06	1743525630000000000	-.7E+06	6034252147777777777
.8E+06	1743606500000000000	-.8E+06	6034171277777777777
.9E+06	1743667350000000000	-.9E+06	6034110427777777777

13.3 TABLE OF POWERS OF 2

Power of 2	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1,024
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152
22	4,194,304
23	8,388,608
24	16,777,216
25	33,554,432
26	67,108,864
27	134,217,728
28	268,435,456
29	536,870,912
30	1,073,741,824
31	2,147,483,648
32	4,294,967,296
33	8,589,934,592
34	17,179,869,184
35	34,359,738,368
36	68,719,476,736
37	137,438,953,472
38	274,877,906,944
39	549,755,813,888

TABLE OF POWERS OF 2

40	1,099,511,627,776
41	2,199,023,255,552
42	4,398,046,511,104
43	8,796,093,022,208
44	17,592,186,044,416
45	35,184,372,088,832
46	70,368,744,177,664
47	140,737,488,355,328
48	281,474,976,710,656
49	562,949,953,421,312
50	1,125,899,906,842,624

13.4 THE CDC STANDARD CHARACTER SET

CDC Graphic	Display Code	029 Punch	ASCII Graphic
:	00	8-2	:
A	01	12-1	A
B	02	12-2	B
C	03	12-3	C
D	04	12-4	D
E	05	12-5	E
F	06	12-6	F
G	07	12-7	G
H	10	12-8	H
I	11	12-9	I
J	12	11-1	J
K	13	11-2	K
L	14	11-3	L
M	15	11-4	M
N	16	11-5	N
O	17	11-6	O
P	20	11-7	P
Q	21	11-8	Q
R	22	11-9	R
S	23	0-2	S
T	24	0-3	T
U	25	0-4	U
V	26	0-5	V
W	27	0-6	W
X	30	0-7	X
Y	31	0-8	Y
Z	32	0-9	Z
0	33	0	0
1	34	1	1
2	35	2	2
3	36	3	3
4	37	4	4
5	40	5	5
6	41	6	6
7	42	7	7
8	43	8	8
9	44	9	9

THE CDC STANDARD CHARACTER SET

CDC Graphic	Display Code	029 Punch	ASCII Graphic	Comments
+	45	12-8-6	+	
-	46	11	-	
*	47	11-8-4	*	
/	50	0-1	/	
(	51	12-8-5	(	
$	53	11-8-3	$	
=	54	8-6	=	equal sign
blank	55	none	blank	blank
,	56	0-8-3	,	comma
.	57	12-8-3	.	period
≠	60	8-3	≠	numeric sign
]	61	12-8-2	]	left brace
[	62	11-8-2	[	right brace
%	63	0-8-4	%	percent
≠	64	8-7	"	not equal - quote
→	65	0-8-5	_	right arrow - underline
∨	66	12-8-7	!	union sign - exclamation mark
∧	67	12	&	intersection sign - ampersand
↑	70	8-5	'	up arrow - apostrophe
↓	71	0-8-7	?	down arrow - question mark
<	72	12-8-4	<	less than
>	73	0-8-6	>	greater than
≤	74	8-4	≤	smaller than or equal to
≥	75	0-8-2	≥	greater than or equal to
¬	76	11-8-7	~	circumflex - tilde
;	77	11-8-6	;	semicolon

13.5 SOME CENTRAL PROCESSOR INSTRUCTIONS AND THEIR FORTRAN EQUIVALENTS

The following central processor instructions are useful to know while debugging a FORTRAN program since they have an easily discernible relation to the FORTRAN code. A complete list of CPU instructions appears in the next section.

Code	Instruction description	Length	FORTRAN equivaLent
0100K	return jump to K	30	CALL subprogram with ENTRY/EXIT address K.
0210K	GO TO B1+K	30	**GO TO** (L1,...,LN),L where B1 holds the value of L and K points to a list of unconditional jump instructions.
033jK	GO TO K if Xj negative	30	a logical IF statement always terminates with this kind of jump instruction.
0400K	unconditional jump to K	30	GO TO memory address K.
30ijk	floating sum Xi=Xj+Xk	15	addition of real constants or variables.
31ijk	floating difference Xi=Xj-Xk	15	subtraction of real constants or variables.
36ijk	integer sum Xi=Xj+Xk	15	addition of integer constants or variables.
37ijk	integer difference Xi=Xj-Xk	15	subtraction of integer constants or variables.
40ijk	floating product Xi=Xj*Xk	15	multiplication of real constants or variables.
44ijk	floating divide Xi=Xj/Xk	15	division of real constants or variables.
5110K	set A1 to K	30	immediately before a call to a subprogram with formal parameters, address register 1 is set to point to the first location of a list of addresses of the actual arguments.

SOME CENTRAL PROCESSOR INSTRUCTIONS AND THEIR FORTRAN EQUIVALENTS

CENTRAL PROCESSOR INSTRUCTIONS

A detailed description of all CPU instructions appears in the 7600 Hardware Reference Manual (publication 60367200X).

00000	error exit to EEA	15 bits
0100K	return jump to K	30 bits
011jK	block copy K+(Bj) words from LCM to SCM	30 bits
012jK	block copy K+(Bj) words from SCM to LCM	30 bits
01300	exchange exit to NEA if exit mode flag not set	15 bits
013jK	exchange exit to K+(Bj) if exit mode flag set	30 bits
014jk	read LCM at (Xk) to Xj	15 bits
015jk	write Xj into LCM at (Xk)	15 bits
0160k	reset input channel (Bk) buffer	15 bits
016jk	read input channel (Bk) status to bj	15 bits
016j0	read real-time clock to Bj	15 bits
0170k	reset output channel (Bk) buffer	15 bits
017jk	read output channel (Bk) status to Bj	15 bits
02i0K	jump to K+(Bi)	30 bits
030jK	branch to K if (Xj) equal 0	30 bits
031jK	branch to K if (Xj) not equal 0	30 bits
032jK	branch to K if (Xj) positive	30 bits
033jK	branch to K if (Xj) negative	30 bits
034jK	branch to K if (Xj) in range	30 bits
035jK	branch to K if (Xj) out of range	30 bits
036jK	branch to K if (Xj) definite	30 bits
037jK	branch to K if (Xj) indefinite	30 bits
04ijK	branch to K if (Bi) equal (Bj)	30 bits
05ijK	branch to K if (Bi) not equal (Bj)	30 bits
06ijK	branch to K if (Bj) < (Bi)	30 bits
07ijK	branch to K if (Bi) < (Bj)	30 bits
10ij0	transmit (Xj) to Xi	15 bits
11ijk	logical product of (Xj) and (Xk) to Xi	15 bits
12ijk	logical sum of (Xj) and (Xk) to Xi	15 bits
13ijk	logical difference of (Xj) and (Xk) to Xi	15 bits
14i0k	transmit complement of (Xk) to Xi	15 bits
15ijk	logical product of (Xj) and complement of (Xk) to Xi	15 bits
16ijk	logical sum of (Xj) and complement of (Xk) to Xi	15 bits
17ijk	logical difference of (Xj) and complement of (Xk) to Xi	15 bits
20ijk	left shift (Xi) by jk places	15 bits
21ijk	right shift (Xi) by jk places	15 bits
22ijk	left shift (Xk) by (Bj) places to Xi	15 bits
23ijk	right shift (Xk) by (Bj) places to Xi	15 bits
24ijk	normalize (Xk) to Xi and Bj	15 bits
25ijk	round normalize (Xk) to Xi and Bj	15 bits

SOME CENTRAL PROCESSOR INSTRUCTIONS AND THEIR FORTRAN EQUIVALENTS

26ijk	unpack (Xk) to Xi and Bj	15 bits
27ijk	pack (Xk) and (Bj) to Xi	15 bits
30ijk	floating sum of (Xj) and (Xk) to Xi	15 bits
31ijk	floating difference of (Xj) and (Xk) to Xi	15 bits
32ijk	floating dp sum of (Xj) and (Xk) to Xi	15 bits
33ijk	floating dp difference of (Xj) and (Xk) to Xi	15 bits
34ijk	round floating sum of (Xj) and (Xk) to Xi	15 bits
35ijk	round floating difference of (Xj) and (Xk) to Xi	15 bits
36ijk	integer sum of (Xj) and (Xk) to Xi	15 bits
37ijk	integer difference of (Xj) and (Xk) to Xi	15 bits
40ijk	floating product of (Xj) and (Xk) to Xi	15 bits
41ijk	round floating product of (Xj) and (Xk) to Xi	15 bits
42ijk	floating dp product of (Xj) and (Xk) to Xi	15 bits
43ijk	form mask of jk bits to Xi	15 bits
44ijk	floating divide (Xj) by (Xk) to Xi	15 bits
45ijk	round floating divide (Xj) by (Xk) to Xi	15 bits
46000	pass (no operation)	15 bits
47i0k	population count of (Xk) to Xi	15 bits
50ijK	set Ai to (Aj) plus K	30 bits
51ijK	set Ai to (Bj) plus K	30 bits
52ijK	set Ai to (Xj) plus K	30 bits
53ijk	set Ai to (Xj) plus (Bk)	15 bits
54ijk	set Ai to (Aj) plus (Bk)	15 bits
55ijk	set Ai to (Aj) minus (Bk)	15 bits
56ijk	set Ai to (Bj) plus (Bk)	15 bits
57ijk	set Ai to (Bj) minus (Bk)	15 bits
60ijK	set Bi to (Aj) plus K	30 bits
61ijK	set Bi to (Bj) plus K	30 bits
62ijK	set Bi to (Xj) plus K	30 bits
63ijk	set Bi to (Xj) plus (Bk)	15 bits
64ijk	set Bi to (Aj) plus (Bk)	15 bits
65ijk	set Bi to (Aj) minus (Bk)	15 bits
66ijk	set Bi to (Bj) plus (Bk)	15 bits
67ijk	set Bi to (Bj) minus (Bk)	15 bits
70ijK	set Xi to (Aj) plus K	30 bits
71ijK	set Xi to (Bj) plus K	30 bits
72ijK	set Xi to (Xj) plus K	30 bits
73ijk	set Xi to (Xj) plus (Bk)	15 bits
74ijk	set Xi to (Aj) plus (Bk)	15 bits
75ijk	set Xi to (Aj) minus (Bk)	15 bits
76ijk	set Xi to (Bj) plus (Bk)	15 bits
77ijk	set Xi to (Bj) minus (Bk)	15 bits

13.6 FORTRAN EXTENDED SUPPLIED PROCEDURES

In this section, the following conventions apply-

X,Y,V	Real numbers
I,J,K,L	Integers
D,E,F,G	Double precision variables declared as - DOUBLE D,E,F,G
Z,W	Complex variables declared as - COMPLEX Z,W

The complete list of FORTRAN Extended Intrinsic Functions, FORTRAN Extended Basic External Functions and CDC supplied procedures is given in the CDC FORTRAN Manual (60497800).

FORTRAN EXTENDED INTRINSIC FUNCTIONS

Intrinsic Function	Definition	Number of Arguments	Symbolic Name	Type of Argument	Type of Function	Example
Absolute Value	|A|	1	ABS	Real	Real	Y=ABS(X)
			IABS	Integer	Integer	J=IABS(I)
			DABS	Double	Double	D=DABS(E)
Truncation	Sign of A times largest integer < |A|	1	AINT	Real	Real	Y=AINT(X)
			INT	Real	Integer	I=INT(X)
			IDINT	Double	Integer	J=IDINT(D)
Remain-dering	A1(modA2)	2	AMOD	Real	Real	V=AMOD(X,Y)
			MOD	Integer	Integer	J=MOD(K,L)
Choosing largest value	Max(A1,.. A2,...)	2-63	AMAX0	Integer	Real	X=AMAX0(I,J,K)
			AMAX1	Real	Real	A=AMAX1(X,Y,V)
			MAX0	Integer	Integer	L=MAX0(I,J,K)
			MAX1	Real	Integer	I=MAX1(X,Y)
			DMAX1	Double	Double	D=DMAX1(E,F,G)
Choosing smallest value	Min(A1,.. A2,...)	2-63	AMIN0	Integer	Real	Y=AMIN0(I,J)
			AMIN1	Real	Real	V=AMIN1(X,Y)
			MIN0	Integer	Integer	L=MIN0(I,J)
			MIN1	Real	Integer	J=MIN1(X,Y)
			DMIN1	Double	Double	D=DMIN1(E,F,G)

FORTRAN EXTENDED SUPPLIED PROCEDURES

FORTRAN EXTENDED INTRINSIC FUNCTIONS

Intrinsic Function	Definition	Number of Arguments	Symbolic Name	Type of Argument	Type of Function	Example
Float	Conversion from integer to real	1	FLOAT	Integer	Real	X=FLOAT(I)
Fix	from real to integer	1	IFIX	Real	Integer	I=IFIX(Y)
Transfer of Sign	Sign of A2 with |A1|	2	SIGN ISIGN DSIGN	Real Integer Double	Real Integer Double	V=SIGN(X,Y) I=ISIGN(J,K) D=DSIGN(E,F)
Real part of complex argument		1	REAL	Complex	Real	X=REAL(Z)
Imaginary part of complex argument		1	AIMAG	Complex	Real	Y=AIMAG(Z)

CDC SUPPLIED INTRINSIC FUNCTIONS

Intrinsic Function	Definition	Number of Arguments	Symbolic Name	Type of Argument	Type of Function	Example
Logical Product	Bit-by-bit logical AND	2-63	AND	Any type	no mode	A=AND(X,Y,Z)
Logical Sum	Bit-by-bit logical OR	2-63	OR	Any type	no mode	A=OR(X,Y,Z)
Complement	Bit-by-bit Boolean comp.	1	COMPL	Any type	no mode	B=COMPL(A)
Shift - Shift A by I bit positions (left circular if I is positive, right with sign extension and end off if I is negative) 0< |I| <60		2	SHIFT	A1-Any type A2-Integer	no mode	B=SHIFT(A,I)
Mask - Form mask of I bits set to 1 starting at the left of the word		1	MASK	Integer	no mode	A=MASK(I)
Address of a variable or array element or EXTERNAL subprogram		1	LOCF	Any type	Integer	J=LOCF(Q)

FORTRAN EXTENDED BASIC EXTERNAL FUNCTIONS

Intrinsic Function	Definition	Number of Argument	Symbolic Name	Type of Argument	Type of Function	Example
Exponential	exp(A)	1	EXP	Real	Real	X=EXP(Y)
		1	DEXP	Double	Double	D=DEXP(E)
	exp(X+iY)	1	CEXP	Complex	Complex	Z=CEXP(W)
Natural Logarithm	Log(A)	1	ALOG	Real	Real	X=ALOG(Y)
		1	DLOG	Double	Double	D=DLOG(E)
	Log(X+iY)	1	CLOG	Complex	Complex	Z=CLOG(W)
Common Logarithm	log(A)	1	ALOG10	Real	Real	Y=ALOG10(X)
		1	DLOG10	Double	Double	D=DLOG10(E)
Trigonometric Cosine	cos(A)	1	COS	Real	Real	X=COS(Y)
		1	DCOS	Double	Double	D=DCOS(E)
	cos(X+iY)	1	CCOS	Complex	Complex	Z=CCOS(W)
Trigonometric Sine	sin(A)	1	SIN	Real	Real	X=SIN(Y)
		1	DSIN	Double	Double	D=DSIN(E)
	sin(X+iY)	1	CSIN	Complex	Complex	Z=CSIN(W)

Intrinsic Function	Definition	Number of Argument	Symbolic Name	Type of Argument	Type of Function	Example
Hyperbolic Tangent	tanh(A)	1	TANH	Real	Real	Y=TANH(X)
Square Root	(A)1/2	1	SQRT	Real	Real	Y=SQRT(X)
		1	DSQRT	Double	Double	D=DSQRT(E)
		1	CSQRT	Complex	Complex	Z=CSQRT(W)
Arctangent	arctan(A)	1	ATAN	Real	Real	Y=ATAN(X)
		1	DATAN	Double	Double	D=DATAN(E)
	arctan(A1/A2)	2	ATAN2	Real	Real	V=ATAN2(X,Y)
		2	DATAN2	Double	Double	D=DATAN2(E,F)

CDC SUPPLIED EXTERNAL FUNCTIONS

Hyperbolic Sine	sinh(A)	1	SINH	Real	Real	Y=SINH(X)
Hyperbolic Cosine	cosh(A)	1	COSH	Real	Real	Y=COSH(X)
Arcosine	arccos(A)	1	ACOS	Real	Real	X=ACOS(Y)
Arcsine	arcsine(A)	1	ASIN	Real	Real	X=ASIN(Y)
Trigonometric Tangent	tan(A)	1	TAN	Real	Real	X=TAN(Y)

MATHEMATICAL ROUTINE ERRORS IN FORTRAN.

13.7 MATHEMATICAL ROUTINE ERRORS IN FORTRAN.

Argument checking is provided unconditionally for the following functions-

EXP,ALOG,SIN,COS,SQRT,ATAN,ATAN2,ASIN,ACOS,CABS,SINH,COSH.

All mathematical functions available in FORTRAN will give error messages when called with invalid arguments provided that the T or traceback option is in use. It is not always obvious what the valid range of arguments to a given function may be, so they are tabulated (for the most part) in the following list.

In the table below, the following conventions apply-

I,J	integers
X	a real number
D	a double precision number
Z	a complex number

Note that 2**47 = 1.40737488355328 E+14

Routine	Entry	Error message produced	Error condition
GOTOERR=	GOTOERR	error in computed GOTO statement- index invalid	GOTO index out of range
CABS	CABS	floating overflow	REAL(Z)**2+AIMAG(Z)**2 is infinite
DTOX*	DTOX	floating overflow	X*DLOG(D)>741.67
DTOD*	DTOD$	floating overflow	D2*DLOG(D1)>741.67
DTOZ*	DTOZ$	floating overflow in D**REAL(Z)	REAL(Z)*DLOG(D)>741.67
		IMAG(Z)*LOG(D) too large	AIMAG(Z)*DLOG(D)>741.67
ITOD*	ITOD$	floating overflow	D*DLOG(DBLE(I))>741.67
ITOJ*	ITOJ$	integer overflow	ABS(I**J)>(2**47-1)
ITOX*	ITOX$	floating overflow	X*ALOG(I)>741.67
ITOZ*	ITOZ$	floating overflow in I**REAL(Z)	REAL(Z)*ALOG(I)>741.67
		IMAG(Z)*LOG(I) too large	AIMAG(Z)*ALOG(I)>741.67
XTOD*	XTOD$	floating overflow	D*DLOG(DBLE(X))>741.67
XTOY*	XTOY$	floating overflow	Y*ALOG(X)>741.67
XTOZ*	XTOZ$	floating overflow in X**REAL(Z)	REAL(Z)*ALOG(X)>741.67
		IMAG(Z)*LOG(X) too large	AIMAG(Z)*ALOG(X)>741.67

MATHEMATICAL ROUTINE ERRORS IN FORTRAN.

CCOS	CCOS	ABS(real part) too large	ABS(REAL(Z))>2**47
		ABS(imag part) too large	AIMAG(Z)>741.67
CSIN	CSIN	ABS(real part) too large	ABS(REAL(Z))>2**47
		ABS(imag part) too large	AIMAG(Z)>741.67
CEXP	CEXP	ABS(real part) too large	REAL(Z)>741.67
		ABS(imag part) too large	AIMAG(Z)>2**47
COS	COS	arg too large, accuracy lost	ABS(X)>2**47
DCOS	DCOS	arg too large, accuracy lost	ABS(D)>2**47
SIN	SIN	arg too large, accuracy lost	ABS(X)>2**47
DSIN	DSIN	arg too large, accuracy lost	ABS(D)>2**47
TAN	TAN	arg too large, accuracy lost	ABS(X)>2**47
EXP	EXP	argument too large, floating overflow	X>741.667483199142
		argument too small	X<-675.8185010459477
DEXP	DEXP	argument too large, floating overflow	D>741.67
MASK	MASK	mask out of range	N<0 or N>60

7600 PROGRAM LIBRARY CATALOG

This section contains the 7600 Program Library Catalog, valid on 1/12/76. Documentation about all the programs and subroutines listed below is available from the Self-service Documentation Room (Building 513 / 1-023) or the Program Library Office (tel 4951). Since the Program Library is frequently updated, users are invited to consult the Program Library News in the Computer Newsletter or the Program Library Notes available from the Computer Notice Boards.

A. ARITHMETIC ROUTINES

A104	INTP	Integer multiplication
A200	DPCMPLX	Double-precision complex arithmetic
A400	BOOL	36-Bit boolean arithmetic
A401	IDENZB	Logical identity
A402	SAME	Compare leftmost 36-bits of two words
A403	AN66ZF	60-Bit Boolean arithmetic
A500	RATIO	Rational algebra package

B. ELEMENTARY FUNCTIONS

B101	ATG	Arc-tangent function
B102	PROXIM	Bring an angle into range
B200	SINH	Hyperbolic sine and cosine
B400	POWEZE	Power series generator

C. POLYNOMIALS AND SPECIAL FUNCTIONS

C100	POLY	Polynomial function
C201	DDVETA	Double-precision roots of cubic
C202	CPOLY	Complex polynomial roots
C203	NZEROS	Number of zeros of a complex function
C204	MULLERC	Polynomial root finder
C205	RZERO	Zeros of a function of one variable
C206	POLY2	Zeros of complex polynomials
C300	ERF	Error function and normal frequency function
C301	GAUSIN	Inverse Gaussian probability function
C302	BESSEL	Bessel functions of orders zero and one
C303	COMBES	Bessel function of complex argument and order
C304	DILOG	Dilogarithm function
C305	GAMMA	Gamma function
C306	CGAMMA	Complex Gamma function
C307	CDIGAM	Complex Digamma or Psi function
C308	ELLICK	Complete elliptic integrals K and E
C309	FRECS	Fresnel integrals
C310	CRAGAM	Ratio of complex Gamma functions
C311	LEGFN	Legendre functions
C312	BESJY	Bessel functions J<0>,J<1>,Y<0>,Y<1>
C313	BESIK	Modified Bessel functions I<0>,I<1>,K<0>,K<1>
C314	THETA1	Theta and Jacobian elliptic functions
C315	ALEGF	Associated Legendre functions
C316	COUL1	Regular Coulomb wave functions
C317	ADIGAM	Digamma or Psi function

C322	DGAMMA	Double-precision Gamma function
C323	GPLOG	Generalized polylogarithms
C324	CGPLOG	Complex-valued generalized polylogarithms
C325	WHIT	Whittaker function M<K,M>
C326	COUL2	Coulomb wave functions
C327	BESIN	Modified Bessel functions I<A+N>(X)
C328	BESJN	Bessel functions J<A+N>(X)
C329	DBESIN	Double-precision modified Bessel functions I<A+N>(X)
C330	DBESJN	Double-precision Bessel functions J<A+N>(X)
C331	BESCJ	Complex Bessel functions J<A+N>(V)
C332	DBESCJ	Double-precision complex Bessel functions J<A+N>(Z)
C333	CLOGAM	Logarithm of the complex Gamma function
C335	CWERF	Complex error function
C336	SININT	Sine and cosine integrals
C337	EXPINT	Exponential integral
C338	FERDIR	Fermi-Dirac function
C339	DAWSON	Dawson's integral
C340	BSIKR3	Modified Bessel functions of order 1/3,2/3
C341	ALOGAM	Logarithm of the Gamma function
C342	STRHO	Struve functions H<0>,H<1>
C400	NEWTON	Solution of simultaneous non-linear equations

D. OPERATIONS ON FUNCTIONS, INTEGRATION, MINIMIZATION

D100	ARSIMP	Simpsons-Rule integration
D101	WTS	Ten and twelve-point Gaussian integration
D102	GAUSS1	Gaussian integration 1
D103	GAUSS	Gaussian integration
D104	CAUCHY	Cauchy principal-value integration
D105	TRIINT	Integration over a triangle
D106	GQUAD	N-point Gaussian quadrature
D108	TRAPER	Trapezoidal-rule integration with an estimated error
D109	DGAUSS	Double-precision Gaussian integration
D110	GPINSP	General-purpose integration in single-precision
D111	GPINDP	General-purpose integration in double-precision
D112	RGAUSS	Recursive Gaussian quadrature of multidimens. Integrals
D113	CGAUSS	Complex integration along a line segment
D114	RIWIAD	Adaptive multidimensional Monte-Carlo integration
D115	CHEBQU	Double-precision Clenshaw-Curtis integration
D201	DIFEQ1	Step-by-step integration of second-order differential equation
D202	DIFEQ2	Step-by-step integration of second-order differential equation
D203	INTSTP	Runge-Kutta integration
D204	DFEQS1	First-order diff'l-equations by Predictor-Corrector
D205	DFEQS2	First-order diff'l-equations by Predictor-Corrector
D206	NORSIK	First-order diff'l-equations by stepsize-method
D207	BULSTO	First-order diff'l-equations by extrapolation
D208	MERSON	Merson integration
D209	DIFSUB	First-order diff'l-equations, stiff or non-stiff
D300	EPDE1	Solution of elliptic partial differential equation

D301	EPDE2	Solution of elliptic partial differential equation
D400	NUMDI1	Numerical differentiation
D506	MINUITS	Function minimization and error analysis
D507	MINSQ	Minimization of sum of squares of functions
D508	LINSQ	Linear least-squares fit
D509	MINVAR	Minimum of a function of one variable
D510	FUMILI	Fitting Chisquare and likelihood functions
D516	MINUITL	Function minimization and error analysis, (long version)
D600	FRED1	Solution of a linear integral equation
D700	RFT	Real fast Fourier transform
D701	FFTRC	Real and complex fast Fourier transforà
D702	CFT	Complex fast Fourier transforà

E. INTERPOLATION AND APPROXIMATIONS

E100	POLINT	Polynomial interpolation routine
E102	MAXIZE	Maximum and minimum elements of arrays
E103	AMAXMU	Largest absolute number in scattered vector
E104	FINT	Linear interpolation function
E105	DIVDIF	Interpolation using divided differences
E106	LOCATF	Search for given element in ordered array
E201	CUR2FT	Fit curves with two or three parameters
E202	LSQFIT	Least-squares polynomial fit with statistical analysis
E204	PSI1	Bivariate least-squares polynomials
E205	PSI2	Multidimensional least-squares polynomial approximation
E206	TRICOF	Coefficients of trigonometric series
E207	TRISUM	Summation of trigonometric series
E208	LSQ	Least-squares fitting subroutine
E209	SPLIN3	Third-order spline approximation
E220	LSQQR	Least-squares solution of overdetermined linear system
E400	ECTRAN	Telescoping of power series
E401	ECTRAD	Telescoping of power series
E402	CHSUM1	Summation of Chebyshev series
E403	CHSUM2	Summation of Chebyshev series
E404	CHECOF1	The coefficients in a Chebyshev expansion
E405	CHECOF	The coefficients in a Chebyshev expansion
E410	CPSC	Complex power series coefficients

F. MATRICES, VECTORS AND LINEAR EQUATIONS

F100	MATIN1	Matrix inversion with solution of linear equations
F101	MATIN2	Double-precision matrix inversion and linear equations
F103	MATRIX	Matrix manipulation with inversion and linear equations.
f104	SYMINV	Inversion of a symmetric matrix
F105	POLROT	Rotate a three-dimensional polar-coordinate systeà
F106	SPXINV	Inversion of a symmetric matrix
F107	SMXINV	Inversion of a symmetric matrix
F108	MUXMAC	Multiplication of a number of complex square matrices

F109	MXEQU	Matrix equations
F110	MXPACK	TC matrix manipulation package
F116	DOT	Scalar product function
F117	CROSS	Vector product
F118	ROT	Rotating a vector
F119	DIST	Distance between two points
F120	DIRCOS	Direction cosines
F121	VECMAN	Vector algebra routines
F122	MATRED	Suppression of given rows and columns in a matrix
F123	CMXPAK	Multiplying real and complex matrices
F124	CXJOIN	Join or split complex matrix into real and imaginary parts
F126	SPDINV	Inversion of a symmetric positive-definite matrix
F133	CDOT	Complex inner product
F136	CDOTC	Conjugate inner product
F140	ERCROS	Crossproduct and error matrix
F141	TRIVEC	Triple vector product and error matrix
F142	ERDOT	Scalar product and its error
F143	ERSCAL	Scaling a vector and its error matrix
F144	TRISCA	Triple scalar product and its error
F145	ERMOD	Modulus of a vector and its error
F146	ERNORM	Normalizing a vector and its error
F200	QREIG	Eigenvalues of a real matrix
F201	EIG5	Eigenvalues of a real matrix
F202	LRCH	Eigenvalues of a real symmetric band matrix
F204	EIGEN	Eigenvalues and eigenvectors of real symmetric matrix
F205	DEIGEN	Double-precision eigensystems of real symmetric matrix
F210	ALLMAT	Eigenvalues and eigenvectors of complex matrices
F220	EISPACK	Matrix eigenvector and eigenvalue package
F221	EISCG1	Eigenvalues/vectors of complex general matrix
F222	EISCH1	Eigenvalues/vectors of complex Hermitian matrix
F223	EISRG1	Eigenvalues/vectors of real general matrix
F224	EISRS1	Eigenvalues/vectors of real symmetric matrix
F225	EISST1	Eigenvalues/vectors of real symmetric tridiagonal matrix
F230	DEFLS	Deflate matrix with known eigenvalue and eigenvector
F301	DET	Determinant of a matrix
F302	SAECUL	Characteristic polynomial
F401	LINEQ1	Solution of a system of simultaneous linear equations
F402	LINEQ2	Solution of a system of simultaneous linear equations
F403	BLEQ	Solution of banded linear equations
F404	TRIDIA	Tridiagonal systems of linear equations
F405	LINSYS	Solution of system of linear equations
F413	CMLIN	Complex determinant, linear equations, matrix inversion
F600	SVD	Singular-value matrix decomposition
F900	EPSIL	Antisymmetric Levi-Civita tensor

G. STATISTICAL ANALYSIS AND PROBABILITY

G100	PROB	Conversion of Chisquare to probability
G101	CHISIN	Inverse of Chisquare distribution
G102	PROBKL	The Kolmogorov distribution
G104	STUDIS	Student's T-distribution and its inverse
G110	DISLAN	The Landau distribution
G111	DISVAV	The Vavilov distribution
G200	IDIM1	Estimation of the dimensionality of a point-set

H. OPERATIONS RESEARCH TECHNIQUES AND MANAGEMENT SCIENCE

H100	SIMPLEX	Linear optimization using the Simplex-algorithm
H300	ASSIGN	Assignment subprogram
H603	PERT	Critical path analysis and resource allocation

I. INPUT

I301	RDNUM	Read a format-free number
I403	SXCARD	Reading cards generated by SUMX
I410	INBCD	Fast unformatted BCD input
I901	UNPAK	A general labelled-data input subroutine

J. OUTPUT

J300	MXPRNT	Matrix print
J402	NAMEZB	Print visible characters
J408	DISPZA	Dispose routine
J410	OUTBCD	Fast unformatted BCD output
J500	WHISTO	Histogram plotting
J501	PLOTXYK	Jolo plotting systeà
J502	SCOOP	Plotting for the incremental plotter
J503	STAP	Plotting routine giving the standard deviation
J504	HIST	Histograms and two-dimensional scatter plots
J507	HISTO	One and two-dimensional histogram plotting
J509	CONPRT	Print function contours in two variables
J510	GD3	Graphic display system
J511	MAP	Table and plot of real function
J512	CPPLOT	Plot a display file generated by GD3
J513	PRPLOT	Print a display file generated by GD3
J520	SURF3D	Plot function of two variables as three-dimensional surface
J530	BINSIZ	Reasonable intervals for histogram binning

K. INTERNAL INFORMATION TRANSFER

K200	DYNSAP	Dynamic storage allocation package
K201	TABMAN	Dynamic table manager
K402	WEOR	Write end-of-record
K403	WIND	Search forward file-mark on tape
K404	BACKZF	Search backward file-mark on tape
K405	FRWDZF	Skip logical records on magnetic tape
K425	UBKVBS	Unblocking IBM/360 spanned variable-length records
K510	RETRNF	Return files

L. EXECUTIVE ROUTINES

L100	HEWPAK	Hewlett-Packard absolute assembler
L200	SNOBOL	Snobol4 compiler
L210	PASCAL	Programming language compiler
L220	BCPL	Non-numeric programming language compiler
L230	POP-2	Programming language compiler
L240	PL-11	PDP-11 high-level language compiler
L300	SPY	Timing and monitoring program performance
L310	TIME76	Instruction timing of CDC/7600 code
L400	PATCHY	Maintain and update card files
L401	UPPA	Update PAM files
L410	UPDTSC	Create Update source file from program deck
L710	DOUBLE	Fortran single to double-precision converter
L720	QUOTES	Hollerith string converter in formats
L800	SUPRLAY	Fast generalized overlay routine

M. DATA HANDLING

M101	SORTZV	Sorting routine for one-dimensional array
M103	FLPSOR	Sort a one-dimensional array into itself
M106	SORTX	Sorting rows of a matrix
M107	KEYSORT	Sort records with multiple keywords
M200	IATOIN	Conversion from A1 Format to Integer
M201	CNVTZF	Convert numbers from IBM/7090 Format
M202	BTODZF	Code conversion routines
M207	INVTZS	Convert numbers to IBM/7090 Format
M211	CMMMZC	Convert numbers from CDC/3800 to CDC/6000 Format
M215	PSCALE	Find power-of-ten scale for printing
M217	PAK3	Pack and unpack floating-point numbers
M219	CVT360	Conversion from IBM/360 to CDC/6000 Format
M221	CNVTHP	Conversion from Hewlett-Packard to CDC/6000 Format
M222	CBYTE	Conversion from 8-bit bytes to display-code
M223	PAK2	Packing and unpacking floating-point numbers
M224	SETFMT	Edit format for printing a floating-point vector
M225	CVTCII	Conversion from CII SIGMA7 to CDC/6000 Format
M226	CCBYTE	Conversion from display-code to 8-bit bytes
M227	PAPTAP	Conversion from 8-bit bytes to display-code
M228	DUMP8B	Dumping 8-bit byte information
M230	DPCCON	Character display-code conversion
M401	BITSZA	Select bit string out of a word
M402	HOLN	Hollerith literals
M403	SHIFT	Circular shift
M404	IMBDZA	Bit manipulation
M405	INVERT	Invert a selected bit string
M406	STOCAR	Character storing routine
M407	STORE	Packing and unpacking of small integers
M408	IPACK	Packing several integers into a word
M409	UBUNCH	Concentration and dispersion of character-strings
M410	A1MANIP	Manipulating BCD-strings in A1 representation
M411	IFORMT	Extract of integer from a BCD-field
M412	AFROMI	Character to integer conversion
M413	BCDWD	Pack several BCD integers into word
M414	IDIGIT	Fetch a character from continuous BCD-string
M415	UHOLLR	Message transfer

M416	UBLOW1	Concentration and dispersion of bit-strings
M417	DECODM	Routine to unpack bits
M418	PACKCH	6-Bit character manipulation
M419	LEFT	Shift function
M420	UBYTE	Unpack 60-bit words into bytes of 8, 16 or 32 bits
M421	BITBYT	Package for handling bits and bytes
M422	PACBYT	Handling packed vectors of bytes
M423	INCBYT	Handling packed vectors of bytes
M424	BLOW8	Packing and unpacking 8-bit bytes
M425	LXBITS	Function returning the record length in bits
M426	BLOW	Unpack continuous 60-bit words into bytes of any length
M427	PKCHAR	Pack 6-bit characters
M428	LOCBYT	Searching for byte-content
M429	SUMBIT	The number of one-bits in a word
M430	FTO360	Convert 60-bit words to IBM/360 32-bit words
M432	FPAC66	Convert Ferranti-Argus 24-bit numbers to CDC/6000
M435	CHMOVE	6-Bit character manipulation
M436	BUNCH	Pack bytes of any length into continuous 60-bit words
M501	IUSAME	Locating a string of same words
M502	UOPT	Decoding BCD-options

N. DEBUGGING

N100	XLOC	Address of a variable
N101	ILGLZV	Detect infinite and indefinite arguments
N103	IUWEED	Detect indefinite and infinite in an array
N104	TRACEBQ	Print trace-back
N105	TRACEQ	Print trace-back
N106	TRACEQR	Fortran traceback after reprieving
N202	DUMRZL	Register store, and dump
N203	TCUMP	Memory dump
N205	GDUMP	Memory dump

Q. SERVICE OR HOUSEKEEPING-PROGRAMMING AIDS

Q500	BARB	Text formatting
Q501	BARBA	Text formatting with lower case
Q600	KWIC	Keyword in context, index list generator
Q800	FLODIA	Flow diagram program for Fortran programs
Q900	TIDY	Cleaning up Fortran programs

R. LOGICAL AND SYMBOLIC

R200	LISP	Texas list-processing interpreter
R201	SCHIP	Symbolic algebra

T. MAGNETS AND ELECTRONICS

T301	PRTAGS	Perturbed alternating-gradient synchrotrons
T308	BEATCH	Particle motion in beam transfer channels
T600	MAGNET	Two-dimensional magnetic fields including saturation
T601	MAREC	Magnet relaxation prograà

T602	MAREA	Magnet relaxation prograà
T603	FITAXI	Fit and calculate axially symmetric field
T620	ECAP	Electronic circuit analysis

U. QUANTUM MECHANICS

U100	CLEBS	Clebsch-Gordan coefficients in algebraic forà
U101	LORENC	Lorentz transformation routines
U110	CLEBSG	Clebsch-Gordan, Wigner 3-J, 6-J, 9-J, Racah, Jahn U-Function

V. RANDOM NUMBERS AND GENERAL PURPOSE UTILITIES

V100	RANNOR	Generator for random-numbers in normal distribution
V101	NORRAN	Fast generator of random-numbers in normal distribution
V102	NORMCO	Generate pair of random-numbers in normal distribution
V103	IRND01	Random bit generator
V104	RNDM	Random-number generator
V105	NRAN	Generate arrays of uniform random-numbers
V110	POISSN	Poisson random-number generator
V111	BINOMI	Binomial random-number generator
V112	MINOMI	Multinomial random-number generator
V130	RAN3D	Generate random three-dimensional vector
V150	HISRAN	Generate random-numbers according to any histograà
V151	FUNRAN	Generate random-numbers according to any function
V200	KBINOM	Binomial coefficients
V201	COMBI	Combinations of R integers out of N
V202	PERMU	Permutations and combinations
V300	UBLANK	Preset an array
V301	UCOPY	Copying an array
V302	UCOCOP	Copying a scattered vector
V303	USWOP	Exchanges two arrays
V304	IUCOMP	Searching a vector for a given element
V305	IULAST	Searching a vector for a non-equal element
V306	MOVE	Storage move
V401	GRAPH	Find compatible node-nets in an incompatibility graph

W. HIGH ENERGY PHYSICS. BEAMS, KINEMATICS, PHASE SPACE

W100	TRAMP1	Tracking and matching prograà
W101	IPSOFA	Beam acceptance
W104	TRAJ	Particle trajectories in the C.P.S.
w110	TR1PAR	Transmission of charged particles
W111	TR2PAR	Transmission of charged particles
W112	TR3PAR	Transmission of charged particles
W126	ELENS	Analysis of electrostatic lenses
W127	RAYTRA	Ray-tracing through potential field
W132	ISRPRO	Secondary particle yields for ISR
W150	TRSPRT	Transport, second-order beam optics
W151	TURTLE	Beam transport simulation, including decay
W300	RELKIN	Relativistic kinematics program
W303	MISMAS	Kinematics for a missing mass experiment

W304	RELKMC	Relativistic kinematics
W305	DALPLT	Boundary Dalitz-plot
W307	STU	Two-body kinematics
W500	SLY	Saclay phase-space
W501	ATHOS	Three-body phase-space and kinematics
W502	PHSP4B	Four-body phase-space
W503	PHSP5B	Five-body phase-space
W505	FOWL	General Monte-Carlo phase-space
W515	GENBOD	N-Body Monte-Carlo event generator
W600	MIXTURE	Range momentum relations for any mixture
W601	MLR	Monte-Carlo generation of multiple scattering
W700	NISCO	Normalized isospin coefficients

X. HIGH ENERGY PHYSICS. PARTICLE DETECTION AND MEASUREMENT

X202	VOMAS	Particle mass table
X203	PAROFI	Particle name table
X401	STRAG	Error due to Straggling
X510	MOMENTM	Momentum of charged particle in magnetic field
X602	PRIPAR	Momentum of primary particle

Y. STATISTICAL DATA ANALYSIS AND PRESENTATION

Y200	SUMX	Statistical analysis and histogramming
Y201	IUCHAN	Find histogram-channel
Y202	IUBIN	Find histogram-channel
Y203	IUHIST	Find histogram-channel
Y250	HBOOK	Statistical analysis and histogramming
Y251	HPLOT	HBOOK graphics interface for histogram plotting
Y300	GAME	Monte-Carlo generation of histograms

Z. MISCELLANEOUS SYSTEM-DEPENDENT FACILITIES

Z002	TIMEZB	Running time
Z005	DATEZB	Date and time
Z006	TIMING	Job time limit and elapsed time
Z007	DATIME	Job time and date
Z022	PF7000	Interface with CDC/7600 permanent file manager
Z023	STAGEF	Stage magnetic tape files to or from CDC/7600
Z024	REQPF	Assign file to permanent-file device
Z025	USTAGE	Suppress post-staging of tape
Z026	INCLCM	Increase or decrease large-core memory allocation
Z027	MEMORY	Increase or decrease small-core memory allocation
Z028	LBCMZB	Length of blank common
Z029	NOARG	Number of parameters in a call statement
Z034	WHICH	CDC/6000 and CDC/7000 computer identification
Z035	ABORT	Abnormal termination of Fortran programs
Z037	REPRIV	Error recovery for CDCscope
Z038	REPINIT	Recovery from Hardware or system conditions
Z039	REPFL	Recovery from Fortran Library errors
Z040	RCV103	Recovery from FTN error 103 (Record-Manager)
Z044	INTRAC	Identify interactive jobs
Z200	XBAS	The basic X-package for handling tapes
Z202	IXFPZL	Detect end-of-record,section,partition,information
Z260	EQUBUF	External file assignment at execution

Z261 KFILE Internal and external file names at execution

GLOSSARY OF MOST COMMONLY USED ABBREVIATIONS AND KEYWORDS

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
B	Binary file - FTN
BCD	Binary Coded Decimal
BLP	Bypass Label Processing
BPI	Bits per Inch
BT	Block Type
C	Block Type C
C	Compile file - UPDATE
CCP	Station for MFA 6000 machine
CCQ	Station for MFB 6000 machine
CDC	Control Data Corporation
CLS	Closed Shop Tape Station
CM	Conversion Mode
CN	Control Password
CNL	Computer Newsletter
CUAC	Computer Users Advisory Comittee
CY	Cycle
D	Debug - FTN
D	Data width - UPDATE
DMP	Dump
DST	Data Summary Tape
E	Existing label - STAGE
EB	EBCDIC code
ECS	Extended Core Storage
EO	Error Option
EX	Extension Password
F	Record Type F
F	Full - UPDATE
FET	File Environment Table
FIT	File Information Table
FL	Field Length
FTN	FORTRAN compiler
I	Block Type I
I	Input file - FTN, UPDATE
ID	Identifier
INDEX	IN dependant Digital EXchange
I/O	Input / Output
JM	Job Manager message prefix
K	Block Type K
L	Label
L	List parameter - FTN, UPDATE
LCM	Large Core Memory
LD	Loader message prefix

LFN	Local File Name
LGO	Load and Go file
LV	Level
MBL	Maximum Block Length
MD	Modification Password
MFA	Main Frame A
MFB	Main Frame B
MFZ	Main Frame Z
MRL	Maximum Record Length
MT	Seven Track Tape
N	No Label - STAGE
N	New program library - UPDATE
NT	Nine Track Tape
O	Output file - UPDATE
O	Owner ID
OPT	Optimisation - FTN
P	Old program library - UPDATE
P	P-Register
PEO	Program Enquiry Office
PF	Permanent File message prefix
PFA	Permanent File station MFA
PFB	Permanent File station MFB
PFN	Permanent File Name
PL	Print Limit
PL	Program Library
PP	Peripheral Processor
PRU	Physical Record Unit
PW	Password
Q	Quick - UPDATE
R	Read Label - LABEL
R	UPDATE files not rewound - UPDATE
RB	Record per Block
RCW	Recovery Control Word
RFL	Request Field Length
RIOS	Remote Input Output Station
RM	Record Manager
RP	Retention Period
S	Record Type S
S	Source output - UPDATE
SCM	Small Core Memory
SL	Standard Label
SPOT	SPun Off Tasks
ST	Station
SYS	System
T	Time parameter - JOB
TP	Tape parameter - 7000 JOB
TPS	Tape Staging Station

TS	Time Sharing
U	Record Type U
UEP	User Exchange Package
UG	Users' Guide
UL	Unlabelled
US	US ASCII code
VSN	Volume Serial Number
VID	Visual IDentifier
W	Record Type W
W	Write Label - LABEL
W	Sequential new program library - UPDATE
X	Record Type X
XR	Multi-read Password
Z	Record Type Z