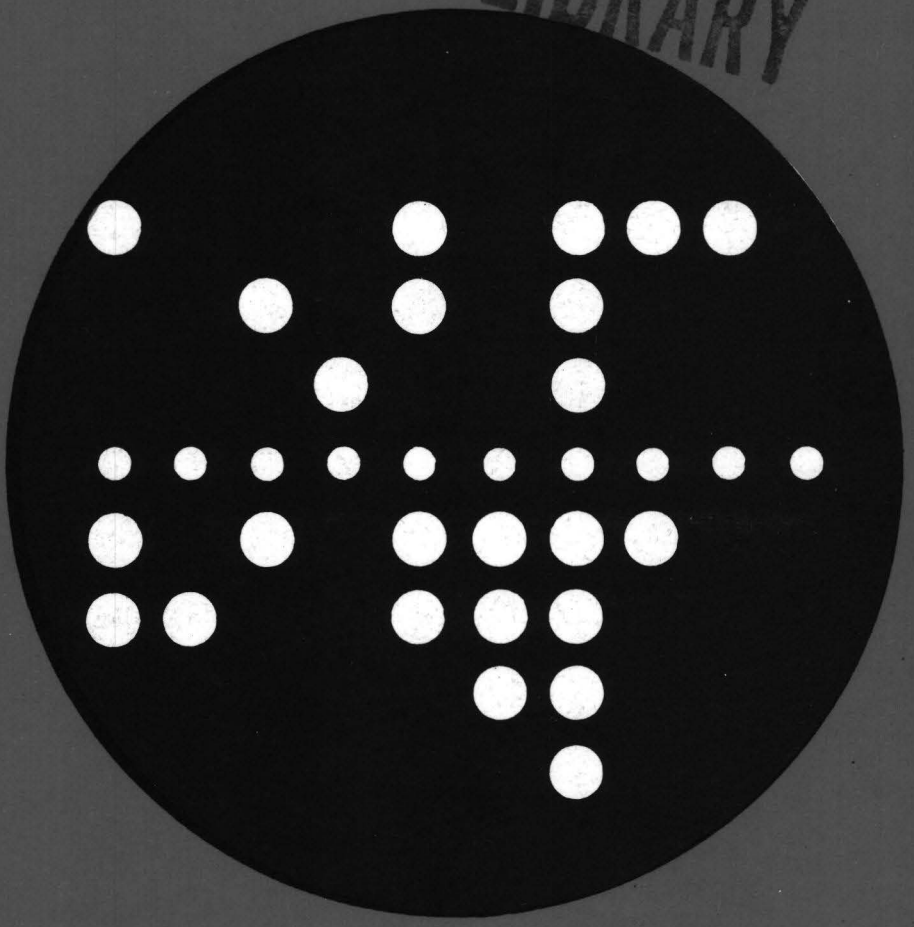
COMPUTING CENTRE NEWSLETTER

October 1979 - N°35



Commission of the European Communities
JOINT
RESEARCH
CENTRE
Ispra Establishment

CONTENTS

Editorial Note	2
Using Computercards is Wrong	3
Present Availability of PASCAL Compilers	13
Installation Notes	20
Statistics of Computing Installation, September	21
Utilisation by Objectives & Accounts, September	22
Statistics of Batch Processing, September	23
Histogram of Equivalent Time Usage	23
List of Personnel	24

EDITORIAL NOTE.

The Computing Centre Newsletter is published monthly except for August and December.

It describes developments, modifications and specific topics in relation to the use of the computing installations of the Joint Research Centre, Ispra Establishment.

The aim of the Newsletter is to provide information of importance to the users of the computing installations, in a form which is both interesting and readable.

The Newsletter also includes articles which are of intellectual and educational value in order to keep the users informed of new advances in computer science topics.

Administration and contact address:

Ms. A. Cambon (tel. 730)
Support to Computing
Building 36
J.R.C. Ispra Establishment
21020-ISPRA (Varese)

USING COMPUTERCARDS IS WRONG

H. I. de Wolde

As explained in a previous issue, the extensive use of computercards is wrong, dangerous, old fashioned, costly, and bears many perils.
The present computing facilities, although not representing the latest developments in the field, offer an extensive number of tools by which the use of punchcards may be avoided.
This article does not contain any new information; it just gives, on request of some users, one of the possible solutions towards program development with a very limited use of punchcards.

Suppose we have to develop a large FORTRAN program which requires many input data parts.
Furthermore, we have access to a TSO terminal, preferably a video.

Librarian

We choose the Librarian for the storage of the basic material, because this system is very well protected against loss or destruction of information, providing that the masterfile manager makes adequate security copies of the information.
Secondly the disk space utilization is very economical and the material is accessible under TSO and from batch jobs.
If you do not yet have access to a masterfile, you may create one by using the information presented in example 1.

The description of the available options and the composition of the reservation card may be found in the Green Book on Librarian.
It is strongly recommended not to create different masterfiles for each development but to share the use of these files. This enables the users to have sensible procedures for back-up and compress and to make the most economic use of the available space.

For the intended task, we need to create two modules in the masterfile:

- Source program
- Input data

Two ways are open for these tasks, either from punchcards in batch processing or by means of a terminal under TSO. The second method is the most economic in terms of manpower. Preparing a deck in punchcards requires roughly three times more efforts in comparison to transfering the same information by means of a video terminal. If, however, you have a big volume of new data to introduce in the computer, it may be convenient to rely on the punch service to have them punched on cards.

A card input deck may be loaded by the job in batch as shown in example 2. Immediately after a back-up of the file has been made by the master file manager, the deck should be destroyed. Otherwise you might rely too much on the presence of a card input deck and cause confusion with new versions of the module. The equivalent job is performed under TSO by the creation of a file, under EDIT control, and successively transfer to the masterfile as the example 3.

Execution

The composition of the deck for the execution of the job is given in example 4.

Such a deck may be punched and brought into the input stream by means of the card reader. However, a TSO SUBMIT can perform the same task and has some advantages. For example, different types of errors are detected and reported before the job submission is accepted by the system. This may save you considerable time.

Now we suppose that you have already created a partitioned data set, named for example TSOPROC.CNTL, in which only the qualifier CNTL is obligatory, the first part of the name may be chosen freely. If you don't have such a data set consult the HELP procedure CREARES.
Create a new member named for example EXEC, of the partitioned data set using of the Editor:

```
EDIT TSOPROC.CNTL(EXEC) NEW
copy the deck composition from example 4 but omit the // and the
jobname
SAVE
END
```

To place the job in the input stream, it is sufficient to give the command:

```
SUBMIT TSOPROC.CNTL(EXEC)
```

Updating

The updating and extension of a program under TSO is much easier and quicker to perform than the same operation in punchcard form. Again, many small errors may be detected before the job is placed in the input stream, which may save you several times the average turn around time.

After the LOGON procedure the commands are arranged as follows:

```
LIBGET PROGA DS('SYSU.name')
EDIT PROGA fortqi
   (editing commands)
SAVE
END
LIBSAVE *
```

In which PROGA is the assumed name of the Librarian module. We strongly advise users to issue frequently a SAVE command during editing and possibly a LIBSAVE, followed by a LIBGET instruction to store the corrections already performed. In case of an unexpected shut down of the computer this will save you a great deal of work. The LIBSAVE command cancels the working copy of the module, so it has to be followed by a LIBGET command to continue the updating.

Private Libraries

If you are developing a large program, it is very useful to store the completed and tested subroutines separately in the masterfile and as a load module in a private library. In this way, the active source part of the program remains smaller and might give a considerable saving in CPU time and channel traffic.

A completed subroutine may be extracted from the source deck, installed as a separate module and loaded to your private library, named SYS1.LIBxxxxx, in which xxxxx are characters which may be chosen by the user. This is performed using the information given in example 5.

If SUBR1 existed already in the Private Library, the old version will be substituted by the new one. The most flexible and economic way of using this procedure is to load it in the same partitioned data set as mentioned earlier. For example with the name TSOPROC.CNTL(LOAD).
Before the submission of this job you have only to modify (under EDIT control), the name of the subroutine and the related record numbers.
Then the following command should be given:

```
SUBMIT TSOPROC.CNTL(LOAD)
```

After the successful execution of the job you must cancel the records n through m in module PROGA to reduce compilation time and because they are safely stored elsewhere under the name SUBR1.

Example 6 describes a situation in which the mounting of a tape is required. The whole procedure may be stored as a member of a partitioned data set and the following tasks are executed:

- The retrieval of the updated parts of the source program, called PROGA.
- The retrieval of the input data, called DATA.
- The compilation of the source.
- The link-editing to include the subroutine load modules.
- The request for tape mounting.
- The execution of the program.

Example 1

Creation of a masterfile

```
//JOBLIB   DD   DSN=LIBRA75,DISP=(SHR,KEEP),UNIT=DISK,
//         VOL=SER=COPICB
//STEP1    EXEC PGM=$$URIAN
//SYSPRINT DD   SYSOUT=A
//MASTER   DD   UNIT=DISK,VOL=SER=USERnn,DSN=SYSU.name,
//         DCB=(BLKSIZE=6444,DSORG=DA),DISP=(NEW,CATLG,DELETE),
//         SPACE=(CYL,(k))
//SYSIN    DD *
-OPT UNIT,DISK,options
/*
//STEP2    EXEC EURDR,U=DISK,V=USERnn
//GO.SYSIN DD*
     reservation card
/*
```

In which:

nn indicates a diskpack of the USER series
name is the second part of the masterfile name
k is the number of cylinders. One cylinder may contain about 6000 records of mixed nature.

Options

-OPT is a Librarian command card. The options at initializing a masterfile are the default definitions for the whole file. At the subsequent loading of modules into the file one may change these default values and define new ones for the single module.

The most common form is:

```
-OPT INIT,DISK,NORESEQ,SEQ=/73,8,10,10/,NOLIST,NOPUNCH,NOEXEC
```

The records are numbered starting in column 73 with a field width of 8 columns. The first sequence number is 10 increased each time by 10. The modules of this masterfile are not automatically renumbered after each run. If you expect to have input records with 80 columns of information you may write:
SEQ=/81,8,10,10/, but this may also be done at the module level.
The option NORESEQ defines no automatic updating of the recordsequence numbers. Using a card with 10 increase, the programmer can easily see which records have been added namely the cards numbered with no 10 multiple. Of course, if you insert more than 9 cards at a single place the system performs some resequencing.

The reservation card is composed as follows:

columns 1-7 the number of the "fiche d'activite"

9-12 the number of authorization

14-17 the number of the programmer (masterfile manager)

19-24 the expiration date

26-69 the masterfile name, left adjusted

8,13,18,25 must be left blank

Space on users disks may be reserved only half a year in advance. To renew the reservation it is sufficient to run the next job:

```
//STEP1     EXEC EURDR,U=DISK,V=USERnn
//GO.SYSIN  DD *
          new reservation card
/*
```

example 2:

Librarian: Load a module by cards

```
//      EXEC LIBRAP,A='SYSU.name',E='USERnn'
//SYSIN DD *
-OPT
-ADD mname,LIST
-DESC.....
-PGMR.....
.
.  cards
.
.
-EMOD
-END
/*
```

In which:
nn indicates a diskpack of the USER series
name is the name to be given to the new module.

Example 3:

Example of the transfer of a new module to the Librarian masterfile(using TSO).

```
LIBSAVE mname FORT DS('SYSU.name') PGMR(author) DESC('.....')
```

In which:

- mname is the module name without qualifiers
- name is the second part of the masterfile name
- author is the name of the programmer. (Don't forget to always use the same notation!)
- DESC('...') specifies up to 36 characters of description.

The masterfile name and the description must be enclosed within apostrophes.

Once the source program and the input data have been loaded, we may execute the program, assuming that we do not yet need tapes or additional files.

Example 4:

Librarian: deckcomposition for compilation and execution

```
//....JOB....
$      TIME --
$      LINES --
$      CLASS 2
//STEP1    EXEC LIBRAP,A='SYSU.name',E='USERnn'
//SYSIN    DD *
-OPT
-SEL,PROGA,EXEC
-EMOD
-END
/*
//STEP2    EXEC LIBRAP,A='SYSU.name',E='USERnn'
//SYSIN    DD *
-OPT
-SEL DATA,EXEC
-EMOD
-END
/*
//STEP3    EXEC FTG1CLG
//CMP.SYSIN DD DSN=*.STEP1.INS.OSJOB,DISP=(OLD,DELETE)
//GO.SYSIN  DD DSN=*.STEP2.INS.OSJOB,DISP=(OLD,DELETE)
```

In which:

- PROGA is the name of the source module
- DATA is the name of the data module
- nn indicates a dispack of the USER series
- name is the second part of the masterfile name

Example 5:

Load a subroutine from Librarian into the Private Library

```
//....JOB CARD....
$      TIME --
$      LINES --
$      CLASS 2
//STEP1   EXEC LIBRAP,A='SYSU.name',E='USERnn'
//SYSIN   DD *
-OPT UTILITY
-OPT
-ADD SUBR1,EXEC
-DESC.....
-PGMR.....
-INC PROGA,n,m
-EMOD
-END
/*
//STEP2   EXEC LIBRAP,A='SYSU.name',E='USERnn'
//SYSIN   DD DSN=*.STEP1.INS.OSJOB,DISP=(OLD,DELETE)
//STEP3   EXEC FTG1C
//CMP.SYSIN DD DSN=*.STEP2.INS.OSJOB,DISP=(OLD,DELETE)
//STEP4   EXEC FTL,NC=NCAL
//LKED.SYSLMOD DD DSN=SYS1.LIBxxxxx,UNIT=DISK,
//              VOL=SER=USERkk,DISP=(OLD,KEEP)
//LKED.SYSLIN  DD DSN=&LOADSET,DISP=(OLD,DELETE)
//              DD *
  NAME SUBR1(R)
/*
```

In which:

SUBR1	is the name to be given to the subroutine which has to be added to the private library
PROGA	is the name of the source module containing the program
n,m	are respectively the first and the last record number of SUBR1 in the program module PROGA
USERnn	is the program module PROGA
USERkk	is the volume where the private library resides
name	is the second part of the masterfile name
LIBxxxx	is the name of the private library to be used

Example 6:

Source from Librarian, compile and link with Private Library.

```
//....JOB CARD....
$      TIME --
$      LINES --
$      CLASS --
$OC TP9=EUtttt,yyy,zzz
//STEP1   EXEC LIBRAP,A='SYSU.name',E='USERnn'
//SYSIN   DD *
-OPT
-SEL PROGA,EXEC
-EMOD
-END
/*
//STEP2   EXEC LIBRAP.A='SYSU.name',E='USERnn'
//SYSIN   DD *
-OPT
-SEL DATA,EXEC
-EMOD
-END
/*
//STEP3   EXEC FTG1CLG,PRN=xxxxx,VLB=USERkk,ULB=DISK
//CMP.SYSIN DD DSN=*.STEP1.INS.OSJOB,DISP=(OLD,DELETE)
//GO.FTaaFoo1 DD (---- tape description)
//GO.SYSIN    DD DSN=*.STEP2.INS.OSJOB,DISP=(OLD,DELETE)
```

In which:

EUtttt	is the tape volume serial number
yyy	is SL or NL
zzz	is Y or N (file protection ring)
SYSU.name	is the name of the masterfile
USERnn	is the volume where the masterfile resides
PROGA	is the source module name
DATA	is the data module name
xxxxx	is the last part of the name of the private library which has full name SYS1.LIBxxxxx
USERkk	is the volume where the private library resides
aa	is the FORTRAN unit name definition.

When the user stores this procedure as a member of a partitioned data set for example: TSOPROC.CNTL(EXEC), one single command will put the job in the input stream:

```
SUBMIT TSOPROC.CNTL(EXEC)
```

References: 1] T.S.O. HELP procedures

2] Installation Notes (JER)

3] Green Book: Librarian

4] ~~Newsletter~~ No 2 ~~Private Program Libraries~~

No 14 IBM Time Sharing option, concepts features & facilities

No 21 The Librarian TSO interface now in use.

PRESENT AVAILABILITY OF PASCAL COMPILERS

A. A. Pollicini

There has been a PASCAL compiler installed on the IBM 370/165 of the JRC-Ispra Computing Centre since September 1977.
This compiler originated from the Informatics Department of IREP at Grenoble University.
The release at present installed is W2.04.00, update (78..33). [1]
The compiler can be accessed in batch by invoking one of the following catalogued procedures and requires 300 kbytes of core.

A) Compilation and module editing

```
//LOAD   EXEC PASCL
//CMP.INPUT   DD  *
     source program
/*
//LKED.SYSLMOD DD UNIT=DISK,VOL=SER=USERxx,DISP=MOD,
//            DSN=MYLIB(MYPROG)
```

Where the generic names "MYLIB" and "MYPROG" stand for a user library and a user program respectively.
USERxx should be replaced by the name of the appropriate user volume on which the library (MYLIB) is stored.

The user program is now stored in load form on the library and may be executed as follows:

```
//XPR   EXEC  PGM=MYPROG
//STEPLIB DD UNIT=DISK,VOL=SER=USERxx,
//           DSN=MYLIB,DISP=SHR
//OUTPUT  DD SYSOUT=A,DCB=(RECFM=FA,BLKSIZE=133)
//INPUT   DD *
     input data
/*
```

B) Compilation, module editing and execution

```
//RUN    EXEC PASCLG
//CMP.INPUT DD *
     source program
/*
//GO.INPUT DD *
     input data
/*
```

A new PASCAL compiler for IBM computers was obtained this year from the Australian Atomic Energy Commission [2] and has recently been installed.
Two catalogued procedures have been designed to access the compiler in batch. The core requirement is 200 kbytes.

AA) Compilation only

```
//COMP EXEC APASC
//CMP.SYSIN DD *
     source program
/*
```

BB) Compilation, module editing and execution

```
//RUN   EXEC APASCLG
//CMP.SYSIN DD *
     source program
//GO.SYSIN DD *
     input data
/*
```

Notice that the compiler returns a completion code greater than zero only if it cannot compile the source. Therefore, in the case of syntax errors the returned code is zero (as well as in the case of successful compilation). For this reason during program development it is advisable to use the procedure APASC.

Copies of the reference manual of the AAEC PASCAL compiler may be purchased at the Computing Support Library (Mrs. Cambon - bld. 36).

GENERAL CONSIDERATIONS

Unfortunately, in the PASCAL world, portability problems are more important than users might expect for such a widely available programming language.
Although the original definition of prof. Wirth [3] was always a fixed reference for any implementor, the standardization of the language by official bodies has only been recently started. The first draft was published this year [4]. This may explain why the two implementations present a number of incompatibilities reviewed in the following:

The first point is related to a different use of some special characters.
In fact the original character set of the language includes some symbols outside the ones allowed by the EBCDIC set. It is usual to replace them by combination of symbols, but in some case there is a lack of uniformity as shown in table I.

PASCAL REPORT	IREP Compiler	AAEC Compiler	USE
{ }	(* *)	(* *)	Comments
[ ]	( ) (a) or |< >| (b)	(. .)	Arrays (a) and sets (b)
↑	"	@	pointers

TABLE I

A second point requiring care is the use of I/O statements and procedures, for which the situation is summarized in Table II.

PASCAL REPORT	IREP COMPILER	AAEC COMPILER
INPUT and OUTPUT files declared in the PROGRAM statement	INPUT and OUTPUT to be declared as FILE OF ... in the context of the source program	as in the REPORT
RESET cannot apply on INPUT file; REWRITE cannot apply on OUTPUT file	RESET applies on INPUT and REWRITE on OUTPUT	as in the REPORT
EOLN (file) PAGE (file) -	- PAGE (file) predefined constants EOL, EOP	EOLN (file) - -

TABLE II

As an example, write commands on a line printer may occur in one among the following alternatives.

```
IREP compiler                AAEC compiler

WRITELN(...);                WRITELN(...);
WRITE(...,EOL);              WRITELN(...);
WRITE(...);WRITE(EOL);       WRITE(...);WRITELN;
```

Moreover, the control of pagination is performed in the IREP implementation in the following ways:

```
WRITE(...,...,EOP);    or    WRITE(...,...);PAGE(OUTPUT);
```

On the contrary the AAEC compiler makes use of the ASA control characters as first character of each line as in FORTRAN FORMAT.

Additionally PACKED ARRAY is allowed in the AAEC compiler and the standard keyword SET is used instead of POWERSET.

These comparisons do not cover all the differences between the two implementations, but are given to warn users about the need for careful reading of the specific reference manuals, before using either of the compilers.

As a conclusion of the present announcement a short example coded accordingly to the AAEC implementation is shown in Appendix 1.

REFERENCES

[1] FAUCHE,JP.; HENNERON,G.; TASSART,G.
Complements au "PASCAL User Manual and Report" concernant l'implementation du Compilateur PASCAL, realisee par l'IREP.
IREP - Universite des Sciences Sociales de GRENOBLE (1977)

[2] COX,G.W.; TOBIAS,J.M.
PASCAL 8000 IBM 360/370 version for OS and VS environments.
Version 1.2 Reference Manual
Australian Atomic Energy Commission (1978)

[3] JENSEN,K.; WIRTH,N.
PASCAL User Manual and Report
Springer-Verlag (2nd edition) (1978)

[4] ADDYMAN,A.M. et al.
A DRAFT DESCRIPTION OF PASCAL
Software practice and Experience - Vol. 9, n. 5 (May 1979) pp. 381-424

APPENDIX 1.

Example of a PASCAL program according to the AAEC implementation

```
PROGRAM  WHATDAY (INPUT,OUTPUT );

(* THE INTENT OF THIS  PASCAL  PROGRAM IS
   TO   S A Y   W H A T   D A Y   OF THE WEEK IS
   ASSOCIATED TO A DATE SPECIFIED BY THREE INTEGERS
   IN THE ORDER 'DAY' 'MONTH' 'YEAR'.
   EACH DATE IS EXPECTED ON A SEPARATE INPUT RECORD. *)

   CONST NULL = 0;
   TYPE  DAYOFWEEK = PACKED ARRAY(.1..9.) OF CHAR;
   VAR   DAY: DAYOFWEEK;
         DATE,MONTH,YEAR,MM,YY,CC: INTEGER;

   PROCEDURE FINDDAY;
(* PLEASE DON'T WORRY ABOUT THIS OBSCURE ALGORITHM! *)
      VAR   PRM1,PRM2,PRM3,PRM4,ORDER: INTEGER;

      BEGIN
         PRM1:= (13*MM-1) DIV 5;
         PRM2:= YY DIV 4;
         PRM3:= CC DIV 4;
         PRM4:= PRM1+PRM2+PRM3+DATE+YY-2*CC;
         ORDER:= PRM4 MOD 7;
         IF ORDER<NULL THEN ORDER:= ORDER+7;
         CASE  ORDER  OF
            0 : DAY:= 'SUNDAY    ';
            1 : DAY:= 'MONDAY    ';
            2 : DAY:= 'TUESDAY   ';
            3 : DAY:= 'WEDNESDAY';
            4 : DAY:= 'THURSDAY ';
            5 : DAY:= 'FRIDAY    ';
            6 : DAY:= 'SATURDAY '
         END
      END; (* FINDDAY *)

   BEGIN
      WHILE NOT EOF(INPUT)  DO
      BEGIN
         READLN( DATE,MONTH,YEAR );
         CC:= YEAR DIV 100; YY:= YEAR MOD 100;
         IF MONTH>2 THEN MM:= MONTH-2
         ELSE BEGIN MM:= MONTH+10;
                 IF YY>NULL THEN YY:= YY-1
                 ELSE BEGIN CC:= CC-1; YY:= 99 END
              END;
         FINDDAY;
         WRITELN('0 THE DAY ASSOCIATED TO ',
                  DATE:2,'.',MONTH:2,'.',YEAR:4,
                 ' IS ',DAY:10)
      END;
      WRITELN('0 END OF DATA');
   END. (* WHATDAY *)
```

EXAMPLE DATA.

```
12  10  1492
 9   5  1956
 7   4  1959
29   2  1980
```

RESULTS.

```
THE DAY ASSOCIATED TO 12.10.1492 IS WEDNESDAY
THE DAY ASSOCIATED TO  9. 5.1956 IS WEDNESDAY
THE DAY ASSOCIATED TO  7. 4.1959 IS TUESDAY
THE DAY ASSOCIATED TO 29. 2.1980 IS FRIDAY
END OF DATA
```

INSTALLATION NOTES (JOB REQUESTS STATEMENTS).

In order to reflect the introduction of the new $OC control cards in the HASP system(see Newsletter 34 -September 1979 for details), one new section for the installation notes(JER) has been added and two of the existing sections(INFO and UTIL) have been updated. To obtain copies of these installation notes users should consult the information in Newsletter 28 - February 1979 pages 6 and 7.

To list the JER(Job Execution Requirements) section it is necessary to request only 1000 lines of printed output(i.e lll should be 001).

An example of the job necessary to list the JER notes is given below:

```
//.........JOB(your job card)...
$      LINES 001
//     EXEC  LIHNO,MEMB=JER
```

Statistics of computing installation utilization.
Report of computing installation exploitation
for the month of September 1979.

	YEAR 1978	YEAR 1979
General		
Number of working days	21 d	20 d
Work hours from 8.00 to 24.00 for	16.00h	16.00h
Duration of scheduled maintenance	18.00h	18.18h
Duration of unexpected maintenance	3.91h	18.35h
Total maintenance time	21.91h	36.53h
Total exploitation time	314.09h	283.47h
CPU time in problem mode	138.85h	130.27h
Batch Processing		
Number of jobs	7914	6369
Number of cards input	1919000	1154800
Number of lines printed	25387000	18836000
Number of cards punched	121000	70800
CPU time	132.11h	114.14h
Number of I/O (Disk)	15012000	19876000
Number of I/O (Magnetic tape)	3841000	3236000
T.S.O		
Number of LOGON's	1313	2487
Number of messages sent by terminals	55611	156243
Number of messages received by terminals	285194	861517
CPU time	4.82h	14.11h
Number of I/O (Disk)	687000	2364000
Connect time	631.34h	1709.99h
IMS		
Total time service is available	381.40h	264.29h
CPU time	1.92h	2.02h
Number of I/O (Disk)	336000	655900

Utilisation of computer centre by objectives and appropriation accounts for the month of September 1979

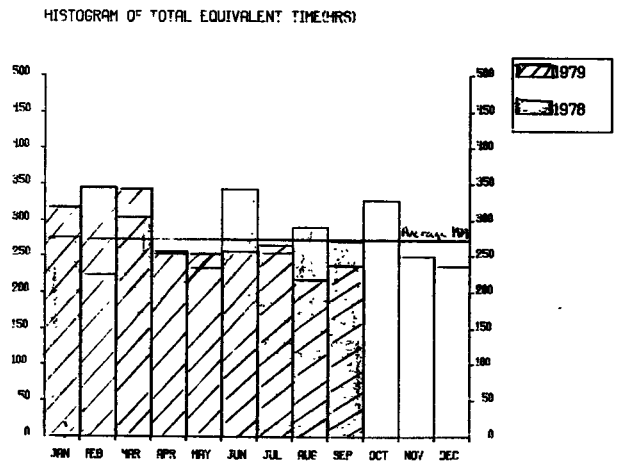
		IBM 370/165 equivalent time in hours
1.20.2	General Services - Administration - Ispra	31.62
1.20.3	General Services - Technical - Ispra	0.11
1.30.3	Central Workshop	2.17
1.30.4	L.M.A.	-
1.90.0	ESSOR	41.39
1.92.0	Support to the Commission	2.43
2.10.1	Reactor Safety	98.82
2.10.2	Plutonium Fuel and Actinide Research	-
2.10.3	Nuclear Materials	13.17
2.20.1	Solar Energy	0.07
2.20.2	Hydrogen	0.25
2.20.4	Design Studies on Thermonuclear Fusion	18.52
2.30.0	Environment and Resources	22.11
2.40.0	METRE	2.60
2.50.1	Informatics	35.86
2.50.2	Training	-
2.50.3	Safeguards	8.82
	TOTAL	277.94
1.94.0	Services to External Users	3.54
	TOTAL	281.48

BATCH PROCESSING DISTRIBUTED BY REQUESTED CORE MEMORY SIZE

	100	200	300	400	600	800	1000	1200	1400	>1400
No. of jobs	1752	2052	1014	820	319	25	11	18	22	-
Elapsed time	76	127	127	153	95	13	10	13	9	-
CPU time	3.3	17.5	22.6	24.1	29.7	4.5	6.3	4.2	2.7	-
"Equiv" time	23	39	46	63	38	6	5	7	4	-
"Turn" time	0.4	1.0	1.5	2.4	3.3	3.2	6.2	4.3	4.0	-
I/O (disk)	2014	2855	3266	5271	1114	207	24	335	184	-
I/O tape)	1987	381	202	600	3	-	-	18	2	-

NOTE.
All times are in hours.
"Equiv" means equivalent.
"Turn" means turn around.
All I/O transfers are measured in 1000's.

PERCENTAGE OF JOBS FINISHED IN LESS THAN										
TIME	15mn	30mn	1hr	2hrs	4hrs	8hrs	1day	2day	3day	6day
%year 1978	34	52	69	84	96	98	99	100	100	100
%year 1979	35	52	66	80	93	99	100	100	100	100



Projected total for 1979 = 3245 hours(using average).
Total for 1978 was = 3253 hours.

REFERENCES TO THE PERSONNEL/FUNCTIONS OF THE COMPUTING CENTRE.

			Room	Tele.
Manager of The Computing Centre		J.Pire		
Responsible for User Registration	Ms.	G.Rambs		
Operations Sector				
Responsible for the Computer Room		A.Binda-Rossetti		
Substituted in case of abscence by:				
Responsible for Peripherals		G.Nocera		
Systems Group				
Responsible for the group		D.König		
Substituted in case of abscence by:		P.A.Moinil		
Responsible for TSO Registration		C.Daolio		
Informatics Support Sector				
Responsible for the Sector	(f.f.)	H.de Wolde	1883	1259
Secretary	Mrs.	G.Hudry	1873	787
Responsible for User Support		H.de Wolde	1883	1259
General Inf./Support Library	Mrs.	A.Cambon	1871	730
Advisory Service/List of Consultants (See Note 1)			1870	730

A.Inzaghi A.A.Pollicini
H.I. de Wolde
R.Meelhuysen M.Dowell

NOTE 1. The advisory service is available in the same room as the Computing Support Library (room 1870). Exact details of the advisory service times for a specific week can be found at the head of any output listing (for that week).
Any informatics problem may be raised. However, the service is not designed to help users with problems which are their sole responsibility. For example, debugging of the logic of programs and requests for information which can easily be retrieved from available documentation.
If necessary, other competent personnel from the informatics division may be contacted by the consultant but not directly by the users.
The users should only contact the person who is the consultant for that specific day and only during the specified hours.
Outside the specified hours general information may be requested from Mrs. A. Cambon in the Computing Support Library.

HOW TO BECOME A REGULAR READER OF THE NEWSLETTER.

Persons interested in receiving regularly the "Computing Centre Newsletter" are requested to fill in the following form and send it to :-

Ms. A. Cambon
Support To Computing
Building 36
Tel. 730.

Please add me to the Newsletter mailing list.

NAME .....

ADDRESS .....

.....

.....

TELEPHONE .....