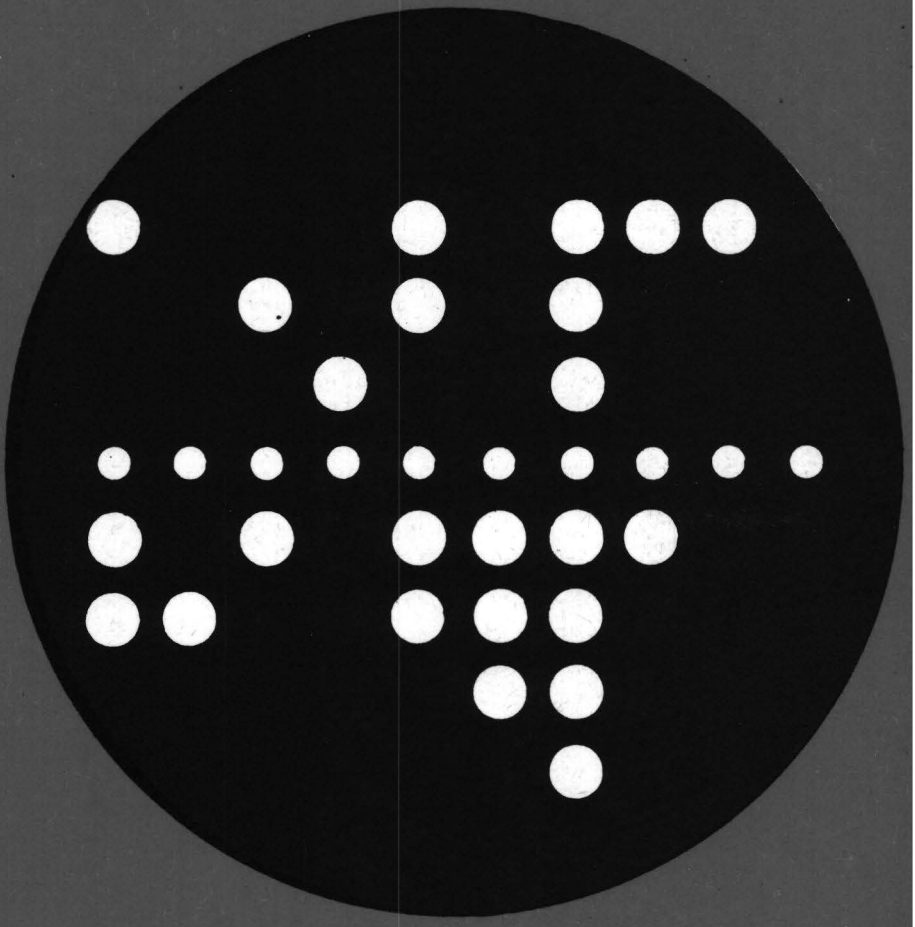
COMPUTING CENTRE NEWSLETTER

June 1979 - N° 32



Commission of the European Communities
JOINT RESEARCH CENTRE
Ispra Establishment

CONTENTS

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)

A GRADUAL MOVE TO FORTRAN 77

A.A. Pollicini

INTRODUCTION

More than one year has elapsed since the approval of the Standard FORTRAN ANSI X3.9-1978 [1] and people are now used to hear and read about FORTRAN 77, as the standardized language was named.
It is quite likely that FORTRAN 77 programs are currently running on those computers for which a compiler has already been made available. Unfortunately this is not the case for IBM!
Although a FORTRAN 77 compiler for IBM computers is being developed and might soon be announced, no official information has been circulated so far.

The more recent public document on the subject is a SEAS note[2] which states:

> 'Still by this meeting, no commitments were made by IBM when there would be a new compiler available'.

In this situation, rather than passively waiting for the availability of a compiler, it is sensible that we try to answer the following question:
What can we do in the meantime?
Of course we still have to use FORTRAN IV, (and therefore we cannot yet profit of a lot of new facilities). But we should devise a kind of languages compatibility which prevents the programs we are coding now from becoming obsolete in a short time.
Such an attempt is described in the following part of this article and consists of a simple programming discipline immediatly applicable by any programmer.
The proposal is based on a careful application of some features available both in the IBM G1 and H Extended FORTRAN compilers. These features have to be used in a way polarized by the awareness of the presence in FORTRAN 77 of unlike features.

It may be considered like a doping technique, finalized to an easier adaptation of the resulting programs to the expected compiler adhering to the approved 1978 standard.

Therefore, the spirit of this article is aimed at suggesting a way for the development of new FORTRAN programs, but is not concerned at all with existing programs which need to be converted.

FINDING OUT THE PROBLEMS

A general overview of the innovations introduced in the new language[3] is of great help in the classification of the kind of the arisen problems. First of all some introduced features have the welcome effect of normalizing some IBM extensions which violated the previous standard ANSI X3.9-1966 [4].
These features are pointed out in the following.

- Arrays. The maximum number of dimensions is fixed to seven.

- The value of an adjustable dimension for a dummy array may be passed either in the argument list or in COMMON.

- The evaluation of sequential exponentiations goes from right to left, thus A**B**C is evaluated as A**(B**C)

- Computed GOTO. Namely the execution of the next statement if the control variable is out of the range of the label list.

- I/O Statements in the short forms: READ f,list
  PRINT f,list

- ERR and END options in READ statements.

- Apostrophe notation both for character constants and for character editing in a FORMAT statement.

- Tabulation editing Tn in a FORMAT statement.

- A character constant may be used as actual argument in a function reference.

- A character constants may be used in a PAUSE statement.

- ENTRY statement.

- List-directed Input/Output.

The use of the above features as allowed by IBM compilers conforms to the FORTRAN 77 standard. Notice, however that the specifications of the standard, in some cases, may allow extended possibilities.
In addition the following IBM features are now integrated in the standard with only minor restrictions or deviations.

- IMPLICIT statement except for the length specifier (*n) in the data type declarator.

- Assignment of initial values in type statements, but avoiding the use of the length specifier in the data type declarator.

- Mixed mode arithmetic except for:(1) combination of the types DOUBLE PRECISION and COMPLEX,(2) the use of entities

of the types INTEGER*2 and COMPLEX*16 which do not exist.

- Alternate RETURN, with the only difference that any actual argument specifying a statement label must have the form:
  *label instead of &label.

- Generic Names (H Extended only) are allowed for intrinsic functions, but some exceptions apply.
  (1) The following IBM functions are not included: COTAN, ERF,ERFC,GAMMA,LGAMMA,QEXT;
  (2) and other are considered as specific names by the standard: AIMAG,CONJG,FLOAT,SINGL.
  Moreover, the GENERIC statement will have to be removed.

All other innovations may cause problems which need to be solved in order to achieve the compatibility we are aiming at.

There is a class of minor problem caused by the following features, for which a possible solution s immediately suggested.

- Suppression of Hollerith constants. It is enough to use always and only the apostrophe notation, already allowed by IBM compilers. Notice that the use of such constants is limited in FORTRAN IV to actual arguments in CALL statements or function references; to initial values in DATA or type statements and to messages in PAUSE statements.
  It should also be noticed that the H editing for character constants within the FORMAT statement was retained by the standard, but it seems an advisable practice to always use the same notation. The one allowed everywhere is the enclosing apostrophes.

- Array element names appearing in an EQUIVALENCE statements must have the same number of subscript expressions as the number of dimensions specified in the array declarator.
  Please conform to this requirement and avoid:

```
DIMENSION   A(10,10),B(10)
EQUIVALENCE (A(1),B(1))
```

- The value of any subscript expression must always be in the limit specified by the corresponding dimension.
  Please conform to this healthy requirement! For instance use A(2,2,2) instead of A(4,4,1) to reference the 10th element of the array declared as follows:
  ```
  DIMENSION A(2,3,5)
  ```
  Notice that both references are valid in FORTRAN IV but only the former one is valid in FORTRAN 77.

The major problems we will be faced with are caused by:

- The modification of the semantics of the DO loop,

- the improvement of the Input/Output facilities,

- the introduction of a definition status for entities,

- the introduction of the data type CHARACTER.

For such problems, it is not always possible to propose a solution in the form of a recommendation.
In particular for character handling some software aids are needed and the library routines described in the last paragraph are intended to be a contribution to move a first step forward.

Finally some marked incompatibilities remain, for which the only solution is to consider their use strictly forbidden. They are namely:

- LOGICAL * 1 data type,
- Hexadecimal constants,
- Dummy arguments enclosed in slashes,
- NAMELIST,
- PUNCH statement,

and, limitedly to the H Extended compiler:
- Extended precision on 128 bits,
- Asynchronus Input/Output.

DO LOOP

The DO loop as defined by FORTRAN 77 specifications, offers a set of extensions.
We have to wait for the availability of a compiler to be able to profit of them, but there are two changes implying careful attention in our intent of producing FORTRAN IV codes which are as much as possible compatible with FORTRAN 77.

The execution of the DO statement results in the evaluation of the iteration count. When it is zero the loop remains inactive and none of the statements of its range are executed.
This means that if the terminal parameter is less than the initial one the DO loop will be skipped. This is in contrast with the current situation in which the loop is always entered and executed at least once.

Thus, if our applications have to perform in the same way now and in the future, it is necessary to introduce before each DO statement a conditional statement which takes into account the actual value of the iteration count.

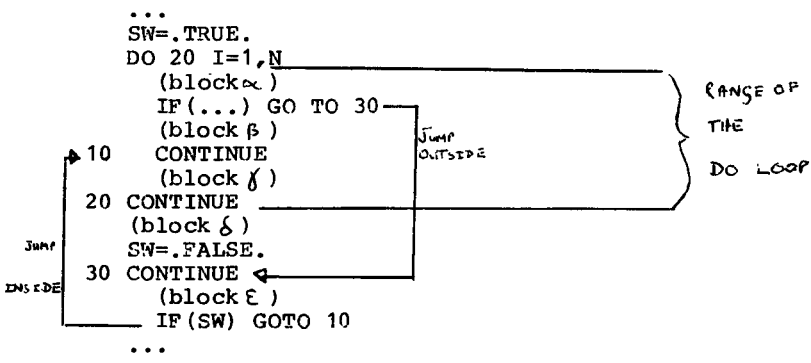
For instance:

```
      ...
      IF (INIT.GT.LAST) GO TO 10
      DO 10 I=INIT,LAST,INCR
      ...
   10 CONTINUE
```

The IF statement will become completely redundant in FORTRAN 77 and could be removed or transformed into a comment line when the program is compiled as a FORTRAN 77 one.
But this fact is of minor concern. Indeed the reliability of an application is far more important than its conciseness and we can enlarge reliability by redundancy.

Moreover, FORTRAN 77 strictly forbids jumps into the range of a DO loop from outside. Therefore, the range extension is no longer valid.

However, the execution of a CALL statement or an external function reference within the range of a DO loop, although physically implying a jump outside and coming back inside at the return from the subprogram formally does not violate the rule. In practice the following situation does not conform to the new standard:



If such a scheme really represents a suitable solution it could be made compatible with FORTRAN 77 by the duplication of the (block ε) as shown in the scheme below which looks clearer than the old one.

```
      ...
      DO 20 I=1,N
        (block α)
        IF(...) GO TO 8
          (block β)
          GO TO 10
    8   CONTINUE
          (block ε) ← DUPLICATED CODE
   10   CONTINUE
        (block γ)
   20 CONTINUE
      (block δ)
   30 CONTINUE
      (block ε)
      CONTINUE
      ...
```

Of course, if the (block ε) is quite large, the duplication of this portion of code is not recommended.
It would be preferable to define a clear interface between the (block ε) and its environment and then to transform the block itself into a subroutine call in place of both occurrences of the identical code.

INPUT - OUTPUT

The 1966 standard allowed effective access to peripherals with simple I/O statements, but dealt only with unformatted and formatted records on sequential files.
The 1978 standard has considerably enlarged the Input/Output facilities providing users with:

- sequential and direct access to external files of records which may be either:
  Unformatted or
  formatted by FORMAT control or
  formatted by list-directed formatting.
- sequential access to internal files of record which may only be formatted by FORMAT control.

As a consequence, a number of "specifiers" are needed to decribe what is the combination required for each file.
These specifiers are controlled by the OPEN,CLOSE,READ,WRITE statements and possibly inquired by the INQUIRE statement.
The more functional way to simulate the FORTRAN 77 Input/Output would certainly be to hide from the application program all the new concepts.
This could be done by designing a subroutine for each I/O statement and giving to each subroutine, as argument list, the list of the specifiers for the corresponding statement.
At the moment of the introduction of a FORTRAN 77 compiler, every CALL to each subroutine would be replaced by the corresponding I/O statement and the actual arguments in the CALL statement would become the value of the specifiers for that I/O statement. No other changes will be needed in the application program:

Let us consider some aspects of the implementation of these subroutines.
The means used for the implementation are not relevant with respect to the portability of the FORTRAN 77 program which will issue in the final phase. Therefore, the choice could equally well be either machine language or machine-dependent high level language.

The required effort to provide the complete range of facilities is quite important and not justified in a short term period such as the one preceding the release of an IBM processor for FORTRAN 77.

The real aim should be to maintain the established facilities, in a way quickly adaptable to FORTRAN 77, keeping as far as possible the structure and the coding of the application program unchanged.

Sequential I/O in the way we are used to use it is intrinsically compatible with FORTRAN 77 which assumes by default the suitable values for all specifiers not pertaining to FORTRAN IV. Therefore, the requirement concerns the two extensions included in the IBM compilers, namely Direct Access and list-directed formatting. Since the latter extension is compatible in its current form (see "FINDING OUT THE PROBLEMS"), we need to simulate only the Direct Access I/O statements.
At this point the scope of the necessary tool is restricted to subroutines receiving only the specifiers for Direct Access.
A reasonable solution would be obtained by implementing in FORTRAN IV, an OPEN subroutine which issues a DEFINE FILE statement, and separate subroutines for read and write operations containing respectively the statements

```
          READ  (u'nr[,fmt]) list
  and     WRITE (u'nr[,fmt]) list
```

But the DEFINE FILE statement as allowed by IBM compilers accepts only integer constant as logical unit and number of blocks.
This lack of flexibility does not allow for the parameterization of the statement with dummy arguments and the use of FORTRAN IV is, therefore, not possible. On the other hand the choice of Assembly language implies a bigger effort which once again is not justified in a transitional period.

In conclusion what is suggested for an immediate, but adaptable use of Direct Access I/O is summarized in the following recommendations:

- Put the DEFINE FILE statement(s) at the beginning of the executable segment of the Main program (or of a subroutine if the structure of the program requires few interfaces between different units).

- Make its(their) location higly visible by adequate comments such as, for instance:

```
C  REMEMBER TO REPLACE THE FOLLOWING STATEMENT WITH
C  AN OPEN STATEMENT WHICH DECLARES:
C  ACCESS=DIRECT
C  RECL=....  (BLOCKLENGTH)
```

- Do not use the FIND statement for simplicity.

- Put each Direct Access READ and WRITE in isolated segments of the program units which contain them.

- Make their location also clearly visible by comments such as these:

```
C  REMEMBER TO REPLACE THE FOLLOWING STATEMENT WITH
C  THE CORRESPONDING FORM VALID IN FORTRAN 77 SPECIFYING
C  REC=....      (RECORD TO BE ACCESSED)
```

DEFINITION STATUS

The 1978 Standard states that the status of any entity of an executable program may be either defined or undefined. The definition status of an entity may change during the execution as a consequence of many events.
In particular, the entities which are local to a subprogram become undefined at the execution of a RETURN or END statement. This allow the processor to reuse free storage in allocating new entities.
But this also implies that the compiler must be able to recognize which entities have to remain defined on exit from the subprogram, that is they have to retain their value at the completion of the execution of the subprogram for a further use. For this purpose the 1978 Standard provides the statement SAVE which is a new verb in the FORTRAN dictionary.
From now on the more plain and reliable way to obtain this in FORTRAN IV will be to make such entities global using a Common block. For clarity it is suggested to put all similar entities belonging to a subprogram in a named Common block which appears both in the Main program and in the appropriate subprogram.

Example.

```
C  DEMONSTRATIVE PROGRAM FOR SAVE STATEMENT SIMULATION
   COMMON/SAVE01/STOR01(3)
   COMMON/SAVE02/STOR02(10)
   ...
   END
   SUBROUTINE SUB01
   COMMON/SAVE01/A,B,K
   ...
   END
   FUNCTION FUNC02(ARR,J)
   COMMON/SAVE02/V
   DIMENSION V(10)
   ...
   END
```

This program configuration can be either kept unchanged or adapted to FORTRAN 77.
In the latter case the adapted program will take advantage of the SAVE facility and the only modifications will be:
(1) to remove all these Common blocks from the Main program and
(2) to replace in each subprograms the specific Common block with an appropriate SAVE statement.

```
   SUBROUTINE SUB01
   SAVE A,B,K
   ...
   END
   FUNCTION FUNC02(ARR,J)
   SAVE V
   DIMENSION V(10)
   ...
   END
```

CHARACTER DATA TYPE

The text of the standard makes a clear distinction between character storage unit and numeric storage unit.
In this way manufacturers may choose the storage allocation of character string which is the most effective on their computer.
Therefore, an equally distinct separation is required in structuring the entities of a program.
From now on we should keep any declaration of variables and arrays designed for storing characters, separate from all other declarative statements.
In general all the following points have to be observed for our simulation purpose.

- Different Common blocks must be used for storing characters and for all other data types.(this norm is compulsory in FORTRAN 77)

- Entities for storing characters must not be EQUIVALENCEd with entities of other data types.(also this norm is compulsory in FORTRAN 77)

- It is suggested that the entities designed to contain characters are of type INTEGER.

- If initial values are requested they have to be specified in the apostophe notation which is the only form accepted in FORTRAN 77.

- For better clarity, all declarative statements referring to such entities should be grouped together and the use of logical parentheses to make them highly visible is strongly recommended. This may be obtained by comment lines with a fixed texts such as:

```
C  BELOW-CHARACTER ENTITIES
...
...
C  ABOVE-CHARACTER ENTITIES
```

- Finally all the functions to be performed on these entities must be committed to special subprograms which hide from the application program the concept of addressing character within a string.
  The implementation of similar subprograms is the temporary solution which allows the application program to work before the installation of a FORTRAN 77 compiler.
  Therefore, they may be coded using the well known "trick" of equivalencing INTEGER and LOGICAL*1 entities. A detailed description of a set of such subprograms follows in the next paragraph.

In the subsequent phase of adaptation to FORTRAN 77, the

modifications of the application program, necessary to match the CHARACTER type will be:

- Replacement of the declarative segment with the suitable CHARACTER Type-statements.

- Substitution of the statements which make reference to a character handling subprogram with appropriate FORTRAN 77 statements.

HOW TO USE THE FORTRAN AIDS LIBRARY

This library is planned to contain routines which help the users in solving small, well identified and general problems. It is named SYS1.LIBFTAID and can be concatenaited to the FORTRAN Library as a private library using the parameterization facility of the catalogued procedures as shown in the following line.

```
//SIMF77 EXEC FTG1CLG,ULB=DISK,VLB=COPICA,PRN=FTAID
```

For the moment it contains five subprograms implementing three primitives which directly simulate features available in FORTRAN 77:

- search a substring within a given string,

- compare two strings for equality,

- move the contents of a string into another string,

plus two more sophisticated primitives easly programmable in a standard way using FORTRAN 77:

- locate the beginning of a field which follows a given separator,

- replace a substring with a new string of different length.

The design of the argument lists and the underlying algorithms of the subprograms was drawn from the specifications of the feature to be simulated.

The first primitive is an integer function, a complete listing of which is given below as an example of the implemention fashion. The heading segment also describes the use of the function.

```
CG
      FUNCTION INDXCH(A,IA,LA,B,IB,LB)
CG
CG   THIS FUNCTION LOOKS FOR THE FIRST OCCURRENCE OF THE SUBSTRING
CG   CONTAINED IN THE RANGE (EXTREMES INCLUDED)                  IB - LB
CG   OF THE GIVEN STRING                                           B
CG   IN THE RANGE BETWEEN THE LOCATIONS (EXTREMES INCLUDED)      IA - LA
CG   OF THE INSPECTED STRING                                       A
CG
CG   THE FUNCTION MAY ASSUME THE VALUE EITHER OF :
CG
CG         - A POSITIVE INTEGER WHICH GIVES THE LOCATION OF THE
CG             LEFTMOST CHARACTER OF THE SUBSTRING WITHIN THE INSPECTED
CG             STRING. (THE LOCATIONS ARE COUNTED STARTING FROM     IA)
CG
CG         - OR  ZERO  IF THE SUBSTRING DOES NOT EXIST
CG                     OR IF THE SPECIFIED RANGE FOR                 A
CG                       IS SHORTER THAN THE ONE SPECIFIED FOR        B
CG                     OR IF THE PARAMETERS ARE INCONSISTENT
CG                        (E.G.  IA.GT.LA)
CG
      DIMENSION A(LA),B(LB)
      LOGICAL*1 A,B
      LOGICAL EQCOMP
      INTEGER PR
      DATA    PR/6/
C
CD    F O R M A T S   FOR MESSAGES ON PRINTER
C
   61 FORMAT('0 ERROR 01 INVALID STRING BOUNDARIES')
   62 FORMAT('0 ERROR 02 EMPTY STRING NOT ALLOWED')
C
CX    PARAMETERS ARE CHECKED FOR CONSISTENCY
C
      IF(IA.LE.LA.AND.IB.LE.LB) GO TO 1
         WRITE(PR,61)
         GO TO 20
    1 IF(LA.GT.0.AND.LB.GT.0) GO TO 10
         WRITE(PR,62)
         GO TO 20
   10 CONTINUE
C
CX    COMPUTE AND VERIFY THE LENGTHS OF BOTH STRINGS
C
      NCHA=LA-IA+1
      NCHB=LB-IB+1
      IF(NCHA.LT.NCHB) GO TO 20
```

```
C
CX       LOOK THE INSPECTED STRING FOR THE OCCURRENCE OF THE SUBSTRING
C
         LAST=LA-NCHB+1
         DO  14 I=IA,LAST
            LF=I+NCHB-1
            IF(EQCOMP(A,I,LF,B,IB,LB)) GO TO 22
   14    CONTINUE
C
CX    ASSIGN VALUE TO THE FUNCTION
C
   20 CONTINUE
      INDXCH=0
      GO TO 29
   22 CONTINUE
      INDXCH=I-IA+1
   29 RETURN
      END
```

The following statement shows an example of reference to the INDXCH function:

```
INIT=INDXCH(REC,ISR,IER,FLD1,ISF,IEF)
```

the same action may be produced in FORTRAN 77 by the statement:

```
INIT=INDEX(REC(ISR:IER),FLD1(ISF:IEF))
```

The comparison primitive has been implemented only for checking identity of strings, while all the relational operators are made available by the standard. However, only the equality or inequality are independent from the collating sequence and therefore, compatible on different computers. Furthermore, the latter can be expressed as the complement of the former.
This primitive has the form of a logical function and is described and referenced as follows.

```
  LOGICAL FUNCTION EQCOMP(A,IA,LA,B,IB,LB)

THIS FUNCTION PERFORMS A COMPARISON, CHARACTER BY CHARACTER,
BETWEEN A FIELD DEFINED IN THE RANGE (EXTREMES INCLUDED)   IA - LA
OF A GIVEN CHARACTER STRING                                    A
AND ANOTHER FIELD DEFINED IN THE RANGE (EXTREMES INCLUDED) IB - LB
OF A SECOND CHARACTER STRING                                   B

THE FUNCTION RETURNS A LOGICAL VALUE WHICH IS :

     .TRUE.    IF THE COMPARED FIELDS ARE EQUAL
     .FALSE.   EITHER IF THE TWO FIELDS ARE DIFFERENT
               OR IF THE SPECIFIED RANGES ARE INCONSISTENT

REMARK -  IF THE TWO FIELDS DO NOT MATCH IN LENGTH, THE COMPARISON
          GOES ON UP TO THE END OF THE LONGER ONE.
          THE SHORTER FIELD IS EXTENDED TO THE RIGHT WITH AS MANY
          BLANKS AS NECESSARY.

  DOCUMENTATION IS COMPLETED FOR UNIT EQCOMP
```

Example of use:

```
      IF (EQCOMP(FLD1,IS1,IE1,FLD2,IS2,IE2)) GO TO 10
         (block α)
         GO TO 20
   10 CONTINUE
         (block β)
   20 CONTINUE
```

This will become:

```
      IF (FLD1(IS1:IE1).NE.FLD2(IS2:IE2)) THEN
         (block α)
      ELSE
         (block β)
      ENDIF
```

The "move" primitive is a subroutine which satisfies the following specifications.

```
  SUBROUTINE MOVCH(A,IA,LA,B,IB,LB,IER)

THIS SUBROUTINE  MOVES  CHARACTERS  FROM A SOURCE STRING          B
ACCORDING TO THE SPECIFICATION OF THE RANGE                  IB - LB
TO A TARGET STRING                                               A
ACCORDING TO THE SPECIFICATION OF THE RANGE                  IA - LA

IF THE TARGET RANGE IS LARGER THAN THE SOURCE ONE,
THE FORMER WILL BE FILLED TO THE RIGHT WITH BLANK(S)

IF THE TARGET RANGE IS SMALLER THAN THE SOURCE ONE,
THE MOVED CHARACTERS WILL BE TRUNCATED TO THE RIGHT

CHARACTERS ARE NOT MOVED IF:

     - IA.GT.LA            OR
     - IB.GT.LB            OR
     - LA.LE.0             OR
     - LB.LE.0

   R E M A R K    EXPECTED MODIFICATIONS OF GLOBAL VARIABLES:
                  - THE TARGET STRING      A      MAY CHANGE
                       ITS CONTENT PARTLY OR TOTALLY
                  - THE VARIABLE     IER     IS ASSIGNED TO

  DOCUMENTATION IS COMPLETED FOR UNIT MOVCH
```

This subroutine may be called now by using:

```
CALL MOVCH(FLD1,IS1,IE1,FLD2,IS2,IE2,IER)
```

and replaced later on by the character assignement statement:

```
FLD1(IS1:IE1)= FLD2(IS2:IE2)
```

The replacements shown for these three primitives look so simple that a special tool providing them automatically could also be forseen.

A variant of the function INDXCH is also provided with the following specifications.

```
   FUNCTION LOCFLD(A,IA,LA,CH)

 THIS FUNCTION LOOKS FOR THE FIRST OCCURRENCE OF THE CHARACTER   CH
 IN THE RANGE BETWEEN THE LOCATIONS (EXTREMES INCLUDED)      IA - LA
 OF THE CHARACTER STRING                                        A

 THE FUNCTION MAY ASSUME DIFFERENT VALUES WHICH HAVE THE FOLLOWING
 MEANINGS:
          ZERO
              =]  CH     IS ABSENT WITHIN THE SPECIFIED RANGE
          INT (NEGATIVE)
              =]  FIRST OCCURRENCE OF    CH    AT LOCATION  IABS(INT)
                  FOLLOWED BY THE SAME CHARACTER UP TO THE END OF
                  THE SPECIFIED RANGE (NO FIELD FOLLOWS)
          INT (POSITIVE)
              =]  THE FIELD FOLLOWING THE FIRST OCCURRENCE
                  (SINGLE OR MULTIPLE) OF      CH
                  BEGINS AT LOCATION          INT
          NOTICE THAT THE LOCATION RETURNED BY THE FUNCTION
          IS ALWAYS COUNTED STARTING FROM THE LEFT BOUND     IA .

   DOCUMENTATION IS COMPLETED FOR UNIT LOCFLD
```

This function has to be recoded in Fortran 77, for instance as follows.

```
      FUNCTION LOCFLD(A,CH)
C
C     ...
C
      CHARACTER A(*),CH
C
      LOC=INDEX(A,CH)
      IF (LOC.NE.0) THEN
         I=LOC
         LOC=-LOC
    1    IF (I.LT.LEN(A)) THEN
            I=I+1
            IF (A(I:I).EQ.CH) GO TO 1
            LOC=I
         ENDIF
      ENDIF
      LOCFLD=LOC
      RETURN
      END
```

And the current form for referencing it:

```
LOCFLD(STRING,ISS,IES,SEP)
```

will have to be, modified to:

```
LOCFLD(STRING(ISS:IES),SEP)
```

The replacement of a substring shifting the portion of the initial string which follows it is possible by calling the subroutine REPLCH which works as described below:

```
  SUBROUTINE REPLCH(A,IA,LA,ISUBS,LSUBS,B,IB,LB,RPL)

THIS SUBROUTINE REPLACES THE CHARACTERS CONTAINED IN THE SUBSTRING
DEFINED BY THE LOWER AND UPPER BOUNDS (INCLUDED)      ISUBS - LSUBS
OF A GIVEN STRING                                                  A
WITH THE CHARACTERS IN THE RANGE (EXTERMES INCLUDED)        IB - LB
OF A SECOND STRING                                                 B

NOTICE THAT THE TARGET SUBSTRING MUST BE COMPLETELY CONTAINED
WITHIN THE RANGE OF THE STRING   A   SPECIFIED BY           IA - LA

THE REPLACEMENT MAY IMPLY EITHER A COMPRESSION TO THE LEFT OR
A TRUNCATION TO THE RIGHT OF THE TARGET STRING   A .
HOWEVER THE TRUNCATION IS ONLY ALLOWED IF THE LOST CHARACTER(S)
IS(ARE) EXCLUSIVELY BLANK(S).

   R E M A R K    EXPECTED MODIFICATION OF GLOBAL VARIABLES:
                  - THE TARGET STRING     A       MAY CHANGE
                       ITS CONTENT, PARTLY OR TOTALLY
                  - THE LOGICAL VARIABLE   RPL   IS ASSIGNED TO
                       THE VALUE  .TRUE.  IF THE REPLACEMENT
                       SUCCEEDED, OR  .FALSE.  OTHERWISE.

  DOCUMENTATION IS COMPLETED FOR UNIT REPLCH
```

Also this subroutine will need to be converted into FORTRAN 77. Statements calling the subroutine will have to be changed from the form:

```
  CALL REPLCH(CARD,ISC,IEC,INIT,LAST,WORD,ISW,IEW,REPL)
```

to the form:

```
  CALL REPLCH(CARD(ISC:IEC),INIT,LAST,WORD(ISW:IEW),REPL)
```

In conclusion as an example of one application of the suggested way for handling characters is included. In appendix A, at the end of this article, is given the listing of program FTDOC which was used to abstract the explanatory headings reported above as description of the library subprograms.

To have a global look at the transformations involved the list of what might be a FORTRAN 77 version is included as appendix B. Note: the program listed in appendix B has only had a "manual" syntax check.

Appendix A
Listing of FORTRAN IV program FTDOC

```
CG     PROGRAM   F T D O C
CG
CG   THIS PROGRAM READS A FORTRAN SOURCE CODE AND LISTS ONLY THE
CG   INSERTED COMMENT LINES, ACCORDING TO A KEY WHICH MAY BE SUPPLIED
CG   AS FIRST CARD.  POSSIBLE KEYS, AS USED IN  THIS PROGRAM ARE:
CG
CG            -  CG  FOR GENERAL INFORMATION
CG            -  CD  FOR DECLARATIVE SEGMENTS
CG            -  CX  FOR EXECUTABLE SEGMENTS
CG            -  C   (TAKEN BY DEFAULT) FOR ALL COMMENT LINES
CG
CG   THE PURPOSE IS TO PROVIDE A LIST OF CONDENSED AND MEANINGFUL
CG   TEXTS WHICH DESCRIBE THE USE AND THE FUNCTION OF EACH PROGRAM
CG   UNIT. THIS GIVES EVIDENCE TO THE INTERNAL CODE DOCUMENTATION
CG   PROVIDED BY THE AUTHOR BY MEANS OF IN-LINE COMMENTS.
CG
CD     BELOW - CHARACTER ENTITIES
C
       INTEGER LINE(20),SUBR(3),FUNC(2),ENDS,END(9),NAME(2),KEY
       INTEGER BLANK(2),LPAR
       DATA SUBR(1),SUBR(2),SUBR(3)/'SUBR','OUTI','NE  '/
       DATA FUNC(1),FUNC(2)/'FUNC','TION'/
       DATA END(1),END(2),END(3),END(4),END(5),END(6),END(7),END(8),
      1     END(9)/'DOCU','MENT','ATIO','N IS',' COM','PLET','ED F',
      2            'OR U','NIT '/
       DATA NAME(1),NAME(2)/'MAIN','    '/,ENDS/'END '/,KEY/'C   '/
       DATA BLANK(1),BLANK(2)/'    ','    '/,LPAR/'(   '/
C
CD     ABOVE - CHARACTER ENTITIES
C
CD     F O R M A T S
C
    50 FORMAT(20A4)
    60 FORMAT('1')
    61 FORMAT(10X,20A4)
    68 FORMAT(///' ACQUISITION OF CONTROL KEY DID NOT SUCCEED')
    69 FORMAT(///' SUBROUTINE OR FUNCTION NAME NOT FOUND IN LINE'
      1 //10X,20A4)
C
CD     MAIN INPUT AND OUTPUT UNITS PARAMETERIZED AS  5  AND  6 .
C
       INTEGER CD,   PR
       DATA    CD/5/,PR/6/
C
CD     DECLARATION OF THE LIBRARY FUNCTION   EQCOMP   AS LOGICAL
C
       LOGICAL EQCOMP
C
       LOGICAL ANALZD,REPL,HEADER
       DATA                HEADER/.TRUE./
C
CD     END OF THE DECLARATIVE SEGMENT
C
C
CX     LOOK FOR A SPECIFIC CONTROL KEY
```

```
C
CX    LOOK FOR A SPECIFIC CONTROL KEY
C
      READ(CD,50,END=99) LINE
      INIT=1
      ISEP=LOCFLD(LINE,INIT,80,BLANK)
      IF (ISEP.GE.0) GO TO 1
         LAST=-ISEP-1
         IF (LAST.GT.4) GO TO 1
            CALL MOVCH(KEY,INIT,LAST,LINE,INIT,LAST,IER)
            IF (IER.NE.0) GO TO 91
               READ(CD,50,END=99) LINE
               GO TO 2
    1 CONTINUE
         LAST=1
    2 CONTINUE
C
CX    SCANNING OF EACH PROGRAM UNIT BEGINS HEREAFTER
C
      WRITE(PR,60)
      JUMP=2
    3 CONTINUE
C
CX    ANALISYS OF EACH LINE IS PERFORMED HEREAFTER
C
      ANALZD=.TRUE.
      CALL MOVCH(LINE,73,80,BLANK,1,8,IER)
      IF (.NOT.EQCOMP(LINE,INIT,LAST,KEY,INIT,LAST)) GO TO 4
         CALL MOVCH(LINE,INIT,LAST,BLANK,INIT,LAST,IER)
C
CX       ISSUE ONE DOCUMENTATION LINE
C
         WRITE(PR,61) LINE
         GO TO 13
    4 CONTINUE
         IF (.NOT.HEADER) GO TO 12
C
CX       LOOK FOR A SUBROUTINE OR FUNCTION STATEMENT
C
         ISTAT=INDXCH(LINE,INIT,80,SUBR,INIT,10)
         IF (ISTAT.LT.7) GO TO 5
            LSTAT=ISTAT+9
            GO TO 6
    5    ISTAT=INDXCH(LINE,INIT,80,FUNC,INIT,8)
         IF (ISTAT.LT.7) GO TO 11
            LSTAT=ISTAT+7
    6    CONTINUE
C
CX          LOOK FOR THE UNIT NAME AND PRINT THE STATEMENT
C
            ISCAN=LSTAT+1
            INAM=LOCFLD(LINE,ISCAN,80,BLANK)
            IF (INAM.LE.0) GO TO 92
```

```
                INAM=LSTAT+INAM
                LNAM=INAM+5
                ISCAN=INAM+1
                DO 7 ICH=ISCAN,LNAM
                   IF (EQCOMP(LINE,ICH,ICH,BLANK,1,1).OR.
     C                  EQCOMP(LINE,ICH,ICH,LPAR,1,1))     GO TO 8
     7          CONTINUE
                GO TO 9
     8          LNAM=ICH-1
     9          LTH=LNAM-INAM+1
                CALL MOVCH(NAME,INIT,LTH,LINE,INAM,LNAM,IER)
C
CX              PRINT THE HEADER STATEMENT OF THE PROGRAM UNIT
C
    10          CONTINUE
                   WRITE(PR,61) LINE
                   READ(CD,50,END=99) LINE
                   CALL MOVCH(LINE,73,80,BLANK,1,8,IER)
                   IF (LOCFLD(LINE,1,80,BLANK).EQ.6) GO TO 10
                      ANALZD=.FALSE.
    11          CONTINUE
                   HEADER=.FALSE.
                   GO TO 13
    12       CONTINUE
C
CX             LOOK FOR THE END STATEMENT
C
               IF (.NOT.EQCOMP(LINE,INIT,6,BLANK,INIT,6)) GO TO 13
                  ISTAT=LOCFLD(LINE,INIT,80,BLANK)
                  IF (ISTAT.LT.7) GO TO 13
                     LSTAT=ISTAT+2
                     IF (.NOT.EQCOMP(LINE,ISTAT,LSTAT,ENDS,1,4)) GO TO 13
                        IF (LOCFLD(LINE,LSTAT,80,BLANK).NE.(-2)) GO TO 13
                           CALL REPLCH(LINE,1,80,ISTAT,LSTAT,END,1,36,REPL)
                           INAM=LSTAT+2
                           IF (REPL) INAM=ISTAT+36
                           CALL MOVCH(LINE,INAM,INAM+5,NAME,1,6,IER)
                           IF (IER.EQ.0) WRITE(PR,61) LINE
                           HEADER=.TRUE.
                           CALL MOVCH(NAME,INIT,6,BLANK,INIT,6,IER)
                           JUMP=1
    13 CONTINUE
C
CX          READ NEXT LINE AND REPEAT THE OPERATIONS
C
            IF (ANALZD) READ(CD,50,END=99) LINE
          GO TO (2,3),JUMP
C
CX       ISSUE ERROR MESSAGES
C
    91 WRITE(PR,68)
       GO TO 99
    92 WRITE(PR,69) LINE
    99 STOP
       END
```

Appendix B
Listing of FORTRAN 77 version of FTDOC

```
CG     PROGRAM   F T D O C
C
C      ...
C
CD     BELOW - CHARACTER ENTITIES
C
       CHARACTER*80 LINE
       CHARACTER*4 ENDS/'END '/,KEY/'C   '/
       CHARACTER LPAR/'('/,BLANK/' '/
       CHARACTER*6 NAME/'MAIN  '/
       CHARACTER*8 FUNC/'FUNCTION'/
       CHARACTER*10 SUBR/'SUBROUTINE'/
       CHARACTER*36 END/'DOCUMENTATION IS COMPLETED FOR UNIT '/
C
CD     ABOVE - CHARACTER ENTITIES
C
CD     F O R M A T S
C
   50 FORMAT(A80)
   60 FORMAT('1')
   61 FORMAT(10X,A80)
   68 FORMAT(///' ACQUISITION OF CONTROL KEY DID NOT SUCCEED')
   69 FORMAT(///' SUBROUTINE OR FUNCTION NAME NOT FOUND IN LINE'
     1 //10X,A80)
C
CD     MAIN INPUT AND OUTPUT UNITS PARAMETERIZED AS  5  AND  6 .
C
       INTEGER CD/5/,PR/6/
C
       LOGICAL ANALZD,REPL,HEADER/.TRUE./
C
CD     END OF THE DECLARATIVE SEGMENT
C
C
CX     LOOK FOR A SPECIFIC CONTROL KEY
C
       READ(CD,50,END=99) LINE
       INIT=1
       ISEP=LOCFLD(LINE,BLANK)
       IF (ISEP.LT.0) THEN
          LAST=-ISEP-1
          IF (LAST.LE.4) THEN
             KEY=LINE(1:LAST)
             READ(CD,50,END=99) LINE
             GO TO 2
          ENDIF
       ENDIF
       LAST=1
    2  CONTINUE
C
```

```
CX     SCANNING OF EACH PROGRAM UNIT BEGINS HEREAFTER
C
       WRITE(PR,60)
       JUMP=2
     3 CONTINUE
C
CX     ANALISYS OF EACH LINE IS PERFORMED HEREAFTER
C
       ANALZD=.TRUE.
       LINE(73:80)=BLANK
       IF (LINE(INIT:LAST).EQ.KEY(INIT:LAST)) THEN
          LINE(INIT:LAST)=BLANK
C
CX        ISSUE ONE DOCUMENTATION LINE
C
          WRITE(PR,61) LINE
       ELSE IF (HEADER) THEN
C
CX        LOOK FOR A SUBROUTINE OR FUNCTION STATEMENT
C
          ISTAT=INDEX(LINE(INIT:80),SUBR)
          IF (ISTAT.GE.7) THEN
             LSTAT=ISTAT+9
          ELSE
             ISTAT=INDEX(LINE(INIT:80),FUNC)
             IF (ISTAT.LT.7) GO TO 11
                LSTAT=ISTAT+7
             ENDIF
C
CX           LOOK FOR THE UNIT NAME AND PRINT THE STATEMENT
C
             ISCAN=LSTAT+1
             INAM=LOCFLD(LINE(ISCAN:80),BLANK)
             IF (INAM.LE.0) GO TO 92
                INAM=LSTAT+INAM
                LNAM=INAM+5
                ISCAN=INAM+1
                DO 7 ICH=ISCAN,LNAM
                   IF (LINE(ICH:ICH).EQ.BLANK.OR.
      C                 LINE(ICH:ICH).EQ.LPAR) GO TO 8
     7          CONTINUE
                GO TO 9
     8          LNAM=ICH-1
     9          NAME=LINE(INAM:LNAM)
C
CX              PRINT THE HEADER STATEMENT OF THE PROGRAM UNIT
C
    10          CONTINUE
                   WRITE(PR,61) LINE
                   READ(CD,50,END=99) LINE
                   LINE(73:80)=BLANK
                   IF (LOCFLD(LINE(1:80),BLANK).EQ.6) GO TO 10
                      ANALZD=.FALSE.
    11    CONTINUE
          HEADER=.FALSE.
       ELSE
```

```
C
CX       LOOK FOR THE END STATEMENT
C
         IF (LINE(INIT:6).EQ.BLANK) THEN
            ISTAT=LOCFLD(LINE(INIT:80),BLANK)
            IF (ISTAT.GE.7) THEN
               LSTAT=ISTAT+2
               IF (LINE(ISTAT:LSTAT).EQ.ENDS) THEN
                  IF (LOCFLD(LINE(LSTAT:80),BLANK).EQ.(-2)) THEN
                     CALL REPLCH(LINE,ISTAT,LSTAT,END,REPL)
                     INAM=LSTAT+2
                     IF (REPL) INAM=ISTAT+36
                     LINE(INAM:INAM+5)=NAME
                     WRITE(PR,61) LINE
                     HEADER=.TRUE.
                     NAME=BLANK
                     JUMP=1
                  ENDIF
               ENDIF
            ENDIF
         ENDIF
      ENDIF
C
CX    READ NEXT LINE AND REPEAT THE OPERATIONS
C
      IF (ANALZD) READ(CD,50,END=99) LINE
      GO TO (2,3),JUMP
C
CX    ISSUE ERROR MESSAGES
C
   92 WRITE(PR,69) LINE
   99 STOP
      END
```

REFERENCES

[1] American National Standard Institute
ANS Programming Language FORTRAN. ANSI X3.9-1978
ANSI,Inc New York (1978)

[2] SEAS FORTRAN Project Committee
Minutes of SEAS WPM Brighton 15th January 1979
SEAS Administrative Secretary Nijmegen (1979)

[3] Pollicini, A.A.
A Look at FORTRAN 77
Computing Centre Newsletter No.25(oct.1978)
JRC Ispra internal publication.
[There is a typing error at page 19.
The tabulation descriptors should read:
(T,TL,TR)]

[4] American National Standard Institute
ANS fortran. ANSI X3.9-1966
ANSI,Inc New York (1966)

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

	YEAR 1978	YEAR 1979
General		
Number of working days	22 d	20 d
Work hours from 8.00 to 24.00 for	16.00h	16.00h
Duration of scheduled maintenance	21.00h	16.50h
Duration of unexpected maintenance	15.83h	16.87h
Total maintenance time	36.83h	33.37h
Total exploitation time	334.99h	286.63h
CPU time in problem mode	183.18h	153.24h
Batch Processing		
Number of jobs	8542	7393
Number of cards input	2019000	1453100
Number of lines printed	27663000	24687000
Number of cards punched	116000	200100
CPU time	178.78h	133.65h
Number of I/O (Disk)	19061000	15859000
Number of I/O (Magnetic tape)	4552000	4058000
T.S.O.		
Number of LOGON's	957	3270
Number of messages sent by terminals	43207	193457
Number of messages received by terminals	177315	1133375
CPU time	3.40h	17.91h
Number of I/O (Disk)	674916	2490000
Connect time	434.74h	2125.00h
IMS		
Total time service is available	415.00h	245.17h
CPU time	1.00h	1.68h
Number of I/O (Disk)	501000	442000

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

		IBM 370/165 equivalent time in hours
1.20.2	General Services - Administration - Ispra	41.68
1.20.3	General Services - Technical - Ispra	2.86
1.30.3	Central Workshop	3.82
1.30.4	L.M.A.	-
1.90.0	ESSOR	8.55
1.92.0	Support to the Commission	2.41
2.10.1	Reactor Safety	169.83
2.10.2	Plutonium Fuel and Actinide Research	8.98
2.10.3	Nuclear Materials	5.42
2.20.1	Solar Energy	0.03
2.20.2	Hydrogen	4.62
2.20.4	Design Studies on Thermonuclear Fusion	6.70
2.30.0	Environment and Resources	17.00
2.40.0	METRE	1.83
2.50.1	Informatics	18.26
2.50.2	Training	-
2.50.3	Safeguards	21.47
	TOTAL	313.47
1.94.0	Services to External Users	8.12
	TOTAL	321.59

BATCH PROCESSING DISTRIBUTED BY REQUESTED CORE MEMORY SIZE

	100	200	300	400	600	800	1000	1200	1400	>1400
No. of jobs	1978	2385	1345	850	356	54	2	72	-	-
Elapsed time	58	140	152	166	64	24	3	36	-	-
CPU time	2.8	20.1	34.5	31.5	19.2	8.1	0.9	16.2	-	-
"Equiv" time	20	43	60	67	28	11	1	20	-	-
"Turn" time	0.5	1.4	2.2	2.4	4.4	4.1	16.1	4.3	-	-
I/O (disk)	1740	3062	3544	4480	1216	412	2	526	-	-
I/O (tape)	1780	509	255	1433	33	4	-	10	-	-

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	27	43	61	78	93	99	100	100	100	100
%year 1979	29	46	63	78	93	99	100	100	100	100

Histogram not available this month.

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		P.Tomba		
Substituted in case of abscence by:		A.Binda-Rossetti		
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		G.Gaggero	1874	787
Secretary	Mrs.	G.Hudry	1873	787
Responsible for User Support		H.de Wolde	1883	1259
General Inf./Support Library	Mrs.	A.Cambon (See Note 2)	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 (see note 2) in the Computing Support Library.

NOTE 2. Mrs. Cambon is at present replaced by Mrs. C La Cognata.

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.

NAME .....

ADDRESS .....

.....

.....

TELEPHONE .....