JADE Computer Note 7

P. Dittmann

4.9.78

IBM <==> NORD

Data transfer via magnetic tape

Two programs have been written at the NORD to transfer data between the two machines:

IBM-DATA transfers binary data (i.e. events)

IBM-CHAR converts character data (i.e. program code).

The tape units at the IBM and at the NORD are compatible (9 tracks, 1600 bpi (DEN=3)). Also both machines have 8bit-bytes. The problems left are:

- Tape lables. They are not supported at the NORD.
  However, if the programs detect an IBM-Label (starting with 'VOL1' in EBCDIC) a skip to the next EOF is issued.
- EBCDIC vs. ASCII code. The conversion is done using the JADE-LIB routine YASEB. Only the lower 6 bits are used, so parity bits don't play any role. Character data on tape are always read or written in EBCDIC-code, on all other devices at the NORD in ASCII-code.
- Record formats. The record length on tape is assumed to be constant and less than 6400 bytes. This is achieved with the IBM record formats RECFM=F and RECFM=VBS. Details are given in the appendix.
- Multi-file data. Many files on one tape is a problem at the IBM. Therefore tapes are assumed to have only one file. There is a possibility to write more than one NORD-SINTRAN file to the tape, but there will be no EOF's written onto tape in between.
- Floating point numbers. There was no attempt to convert 'REAL's at the NORD. A conversion routine exists at the IBM.

The programs are written in FORTRAN. If somebody wants to write his own tape I/O-routines he may easily consult the source listing.

A. IBM-CHAR Program description

1. From IBM to NORD

   The example shows how to transfer program coding written under NEWLIB control from the IBM to the NORD.

a. At the IBM:

```
// EXEC NEWLIB, PS='sourcelib'
./ COPY name1 [name2]  (see NEWLIB manual)
// COPY DD UNIT=TAPE, DSN=what you like, VOL=SER=volume number,
// DCB=(DEN=3, RECFM=F, BLKSIZE=80)[, LABEL=(,NL)]
```

Instead of ./ COPY one may use ./ EXPAND if one has MACROS. The LABEL parameter is only necessary if the tape is not IBM standard-labelled. The BLKSIZE parameter must be ≤ 136.

b. At the NORD:

Load the tape on the tape unit, press ONLINE.

@ <u>(J-P)IBM-CHAR</u>

INPUT FILE: <u>M-T-1</u>

OUTPUT FILE: <u>"file-name"</u>

The file names may be typed in SINTRAN convention. The quotes must be omitted if the file already exists. In this case it gets over-written. If one types L-P on output file one gets a copy on the line-printer.

LIST INPUT: <u>10</u> number of lines to be listed on terminal

The program checks now, if the tape is mounted, and skips an IBM-Label, if it is there.

RECORD LENGTH ON TAPE IN BYTES: <u>80</u> 10-136 possible.

SKIP COLUMNS 73-80: <u>YES</u> contains useless NEWLIB information.

The program starts copying. Each byte is translated from EBCDIC-code to ASCII-code. In addition, trailing blanks are removed from the record.

END OF FILE

NUMBER OF RECORDS WRITTEN

The file can now be accessed from the NORD editor. If any error messages appear in the program, try to see a NORD expert.

- 3 -

2. From NORD to IBM

The example shows how to transfer program coding written at the NORD to NEWLIB at the IBM.

a. At the NORD:

Load an unlabelled tape, otherwise you run into problems since there will be no trailing labels written.

```
@(J-P)IBM-CHAR
INPUT FILE: file-name
OUTPUT FILE: M-T-1
) see above
END OF FILE. COPY ANOTHER FILE: YES or RETURN
```

There will be no EOF between the files on the tape.

b. At the IBM:

```
// EXEC NEWLIB, PS='sourcelib', TO=DISK
// SYSINØ DD *
./ ADD name,
// SYSIN DD UNIT=TAPE, DSN=something, VOL=SER=volume#,
// DCB = (DEN=3, RECFM=F, BLKSIZE=80), LABEL=(,NL)
```

B. IBM - DATA Program description

1. From IBM to NORD

The example shows how to transfer Monte-Carlo events from the IBM to the NORD.

a. At the IBM

```
// EXEC DUPDAT
// PRINT SYSOUT=A,
// INPUT DSN=F11BAR.NEWT, DISP=OLD
// OUTPUT UNIT=TAPE, DSN=anything, VOL=SER=volume#,
// DCB=DEN=3 [, LABEL=(,NL)]
```

Here it is assumed that the INPUT tape is written with RECFM=VBS.

b. At the NORD (for more details see example A.1.b)

Mount tape and press ONLINE

- 4 -

```
@(J-P)IBM-DATA
INPUT-FILE: M-T-1
OUTPUT-FILE: "file-name"
```

If the output file does not yet exist, one should create a continuous file before program starts. Ask NORD experts how to do this. If the output file is L-P an octal dump is written to the line-printer (BREAK, if you think it is enough).

```
RECORD FORMAT ON TAPE: F or VBS
```

The record length is read from the data on tape.

```
RECORD FORMAT ON DISK: must be the same as above:

END OF FILE
```

The data can now be read by the JADE-LIB routine YRVBS.

2. From NORD to IBM
   This example shows how to transfer constants from the NORD to the IBM. For more details see the other examples.

a. At the NORD:
   Mount an unlabelled tape, press ONLINE.

```
@(J-P)IBM-DATA
INPUT-FILE: file-name
OUTPUT-FILE: M-T-1
RECORD FORMAT ON TAPE: F
RECORD FORMAT ON DISK: F
RECORD LENGTH ON DISK IN BYTES: 2048* (< 6400)

END OF FILE. COPY ANOTHER FILE: YES or RETURN
```

b. At the IBM

```
DIMENSION IA(512 )
:
:
READ(1) IA
:
:
//GO,FT01F001 DD UNIT=TAPE, DSN=something, VOL=SER=volume#,
// DCB=(DEN=3, RECFM=F, BLKSIZE=2048*), LABEL=(,NL)
```

The numbers marked with a * must be the same.
The routine to write VBS records at the NORD is not yet written, but anticipated.

- 5 -
(Appendix)

Appendix: IBM Record formats F and VBS

Record format F:

FORMAT CONTROL

The following discussion provides information on records written under control of a FORMAT statement.

UNBLOCKED RECORDS: For fixed-length and undefined records, the record length and buffer length are specified in the BLKSIZE subparameter. For variable-length records, the record length is specified in the LRECL subparameter; the buffer length in the BLKSIZE subparameter. The information coded in a FORMAT statement indicates the FORTRAN record length (in bytes).

F ed-Length Records: For unblocked fixed- gth records written under FORMAT con- l, the FORTRAN record length must not exceed BLKSIZE (see Figure 32).

Example: Assume BLKSIZE=44

```
10  FORMAT(F10.5,I6,2F12.5,'SUMS')
    WRITE(20,10)AB,NA,AC,AD
```

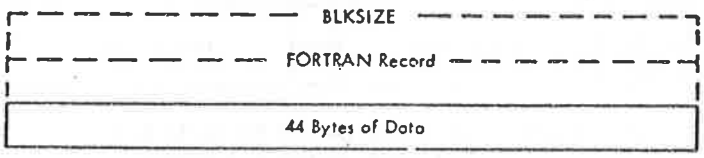

Figure 32. FORTRAN Record (FORMAT Control) Fixed-Length Specification

If the FORTRAN record length is less than BLKSIZE, the record is padded with blanks to fill the remainder of the buffer (see Figure 33). The entire buffer is written.

Example: Assume BLKSIZE=56

```
5   FORMAT(F10.5,I6,F12.5,'TOTAL')
    WRITE(15,5)BC,NB,BD
```

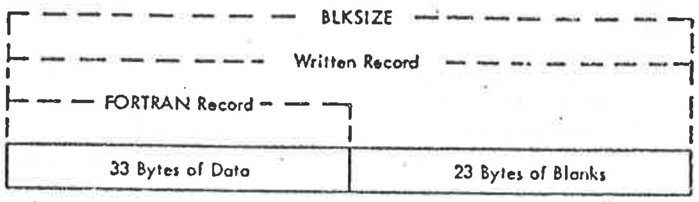

Figure 33. FORTRAN Record (FORMAT Control) Fixed-Length Specification and FORTRAN Record Length Less Than BLKSIZE

Record format VBS:

UNFORMATTED CONTROL

Only variable-length records can be written without format control, i.e., the RECFM subparameter must be VBS. (If nothing is specified, VBS is assumed.)

Records written with no FORMAT control have the following properties:

- The length of the logical record is controlled by the type and number of variables in the input/output list of its associated READ or WRITE statement.
- A logical record can be physically recorded on an external medium as one or more record segments. Not all segments of a logical record must fit into the same physical record (block).
- Two quantities control the manner in which records are placed on an external medium: the block size (as specified by the BLKSIZE parameter), and the logical record (as defined by the length of the I/O list). BLKSIZE is specified as part of the DCB parameter of the data definition (DD) statement. If not specified, FORTRAN provides default values.

Each block begins with a 4-byte block descriptor word (BDW); each segment begins with a 4-byte segment descriptor word (SDW). The SDWs and BDWs are provided by the system. Each buffer begins with a 4-byte block descriptor word (BDW). The SDWs and BDWs are provided by the system.

The format of a BDW is given in Figure

block-length	reserved
2 bytes	2 bytes

Figure 39. Format of a Block Descriptor Word (BDW)

where:

block-length
is a binary count of the total number of bytes of information in the block. This includes four bytes for the BDW plus the sum of the segment lengths specified in each SDW in the block. (The permissible range is from 8 to 32,760 bytes.)

reserved
is two bytes of zeros reserved for system use.

The format of an SDW is given in Figure 0.

segment-length	code	reserved
2 bytes	1 byte	1 byte

Figure 40. Format of a Segment Descriptor Word (SDW)

where:

segment-length
is a binary count of the number of bytes in the SDW (four bytes) plus the number of bytes in the data portion of the segment following the SDW. (The permissible range is from 4 to 32,756 bytes.)

code
indicates the position of the segment with respect to the other segments (if any) of the record. Bits 0 through 5 are reserved for system use and are set to 0. Bits ~~6~~ and ~~7~~ contain the codes: 8 9

Code	Meaning
00	This segment is not followed or preceded by another segment of the record.
01	This segment is the first of a multisegment record.
10	This segment is the last of a multisegment record.
11	This segment is neither the first nor last of a multi-segment record.

reserved
is a byte of zeros reserved for system use.

Blocked Records: For blocked records, if the logical record length is less than or equal to the length of the block (allowing four bytes for the BDW and four bytes for the SDW), the block will contain the entire logical record. The next record, preceded by its SDW, begins in the same block (see Figure 40.3).

If the logical record is greater than the length of the block, the record is divided into record segments. The number of segments is determined from the READ/WRITE statement and the BLKSIZE specification. The next record, preceded by its SDW, begins in the same block. If the length of the second record exceeds the remainder of the block it too is segmented (see Figure 40.4).

Example: Assume BLKSIZE=44

```
     REAL*8 A,B,C,D
       .
       .
       .
     WRITE(9) A,B
       .
       .
       .
     WRITE(9) C,D
```

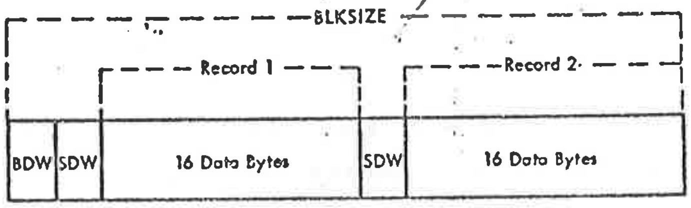

Figure 40.3. Blocked Records Written Without FORMAT Control

Example: Assume BLKSIZE=32

```
     REAL*8 A,B,C,D,E,F,G,H
     WRITE(9) A,B,C,D
     WRITE(9) E,F,G,H
```

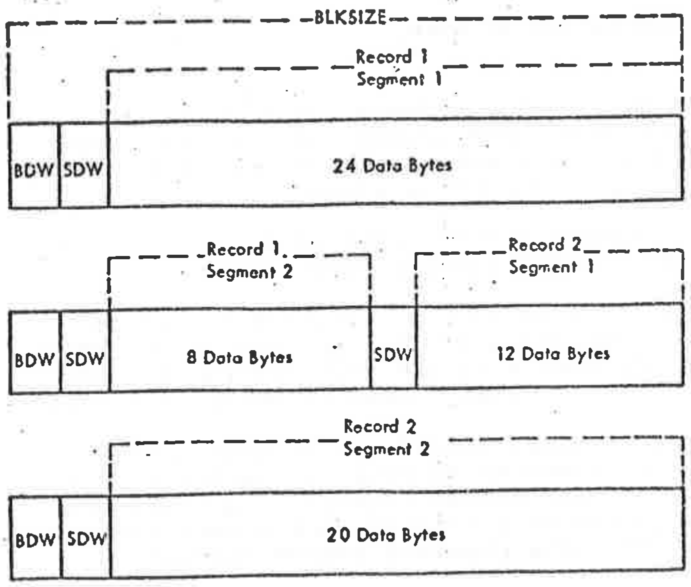

Figure 40.4. Blocked Segmented Records Written Without FORMAT Control