Chapter 26: Sequential File Organization

Author's Note: *This chapter is copied almost verbatim from the material in Chapter 18 of the textbook by Peter Abel. It is used by permission.*

In this chapter, you examine sequential file organization for DOS and OS and learn how to create and read such files. You will also examine the definition and processing of variable-length records.

The processing of sequential files involves the same imperative macros used up to now: OPEN, CLOSE, GET, and PUT. IOCS (data management) handles all the necessary label processing and blocking and deblocking of records. Other than job control commands, the only major difference is the use of blocked records.

An installation has to make a (perhaps arbitrary) choice of a blocking factor when a file is created, and all programs that subsequently process the file define the same blocking factor. A program may also define one or more I/O buffers; if records are highly blocked, a second buffer involves more space in main storage with perhaps little gained in processing speed.

CREATING A TAPE FILE

The first two examples create a tape file for DOS and OS. The programs accept input data from the system reader and write four records per block onto tape.

For both programs, OPEN checks the volume label and header label, and CLOSE writes the last block (even if it contains fewer than four records) and writes a trailer label.

DOS Program to Create a Tape File

The DOS DTFMT file definition macro defines a magnetic tape file. You define a DTFMT macro with a unique name for each tape input or output file that the program processes. The parameters that you code are similar to those for the DTFCD and DTFPR macros.

In Fig. 26–1, the program reads records into RECDlN and transfers required fields to a tape work area named TAPEWORK. The program then writes this work area to a tape output file named FILOTP. Based on the BLKSIZE entry in the DTFMT, IOCS blocks four records before physically writing the block onto tape. Thus for every four input records that the program reads, IOCS writes one block of four records onto tape.

The following explains the DTFMT entries:

BLKSIZE=360 means that each block to be written from the IOAREA is 360 bytes long, based on four records at 90 bytes each.

DEVADDR= SYS025 denotes the logical address of the tape device to write the file.

FILABL = STD indicates that the tape file contains standard labels, described in Chapter 25.

IOAREAl and **IOAREA2** are the two IOCS buffers, each defined with the same length (360) as BLKSIZE. If your blocks are especially large, you may omit defining a second buffer to reduce program size.

RECFORM=FIXBLK defines output records as fixed-length and blocked. Records on tape and disk may also be variable–length or unblocked.

RECSIZE= 90 means that each fixed-length record is 90 bytes in length, the same as the work area.

TYPEFLE=OUTPUT means that the file is output, that is, for writing only. Other options are INPUT and WORK (for a work file). .

WORKA=YES means that the program is to process output records in awork area. In this program, TAPEWORK is the work area and has the same length as RECSIZE, 90 bytes. Alternatively, you may code IOREG and use the macro PUT FILEOTP with no work area coded in the operand.

The DTFMT file definition macro for tape input requires an entry EOFADDR=**address** to indicate the name of the routine where IOCS links on reaching the end of the tape file.

OS Program to Create a Tape File

For OS, you define a DCB macro with a unique name for each tape input or output file that the program processes. The parameters that you code are similar to those for the DCB macros covered earlier.

In Fig. 26–2, the program reads records into RECDIN and transfers required fields to a tape work area named TAPEWORK. The program then writes this work area to a tape output file named FILOTP. Based on the BLKSIZE entry in job control, the system blocks four records before physically writing the block onto tape. Thus for every four input records that the program reads, the system writes one block of four records.

The DD job commands for the files appear first in the job stream and provide some entries that could also appear in the DCB. This common practice enables users to change entries without reassembling programs. The DD entries for the tape file, TAPEOT, are as follows:

DSNAME=TRFILE provides the data set name.

DISP=(NEW,PASS) means that the file is new (to be created) and is to be kept temporarily. Note that "(NEW,PASS)" is written without spaces.

UNIT=3420 provides the tape drive model.

BLKSIZE=360 means that each block to be written from-the IOAREA is 360 bytes long, based on four records at 90 bytes each.

RECFM=FB defines output records as fixed–length and blocked. Records on tape and disk may also be variable–length (V) or unblocked.

DEN= 3 indicates tape density as 1,600 bpi. (DEN=2 would mean 800 bpi.)

The following explains the DCB entries:

DDNAME=TAPEOT relates to the same name in the DD job control command:

```
                    //GO.TAPEOT …
```

DSORG= PS defines output as physical sequential.

LRECL=90 provides the logical record length for each record.

MACRF=(PM) defines the type of output operation as put and move from a work area. **MACRF=(PL)** would allow you to use locate mode to process records directly in the buffers.

The DCB file definition macro for tape input requires an entry EOFADDR=**address** to indicate the name of the routine where IOCS links on reaching the end of the tape file.

Also, another DCB entry, EROPT, provides for an action if an input operation encounters problems. The options are as follows:

=ACC Accept the possibly erroneous block of data.

=SKP Skip the data block entirely and resume with the next one.

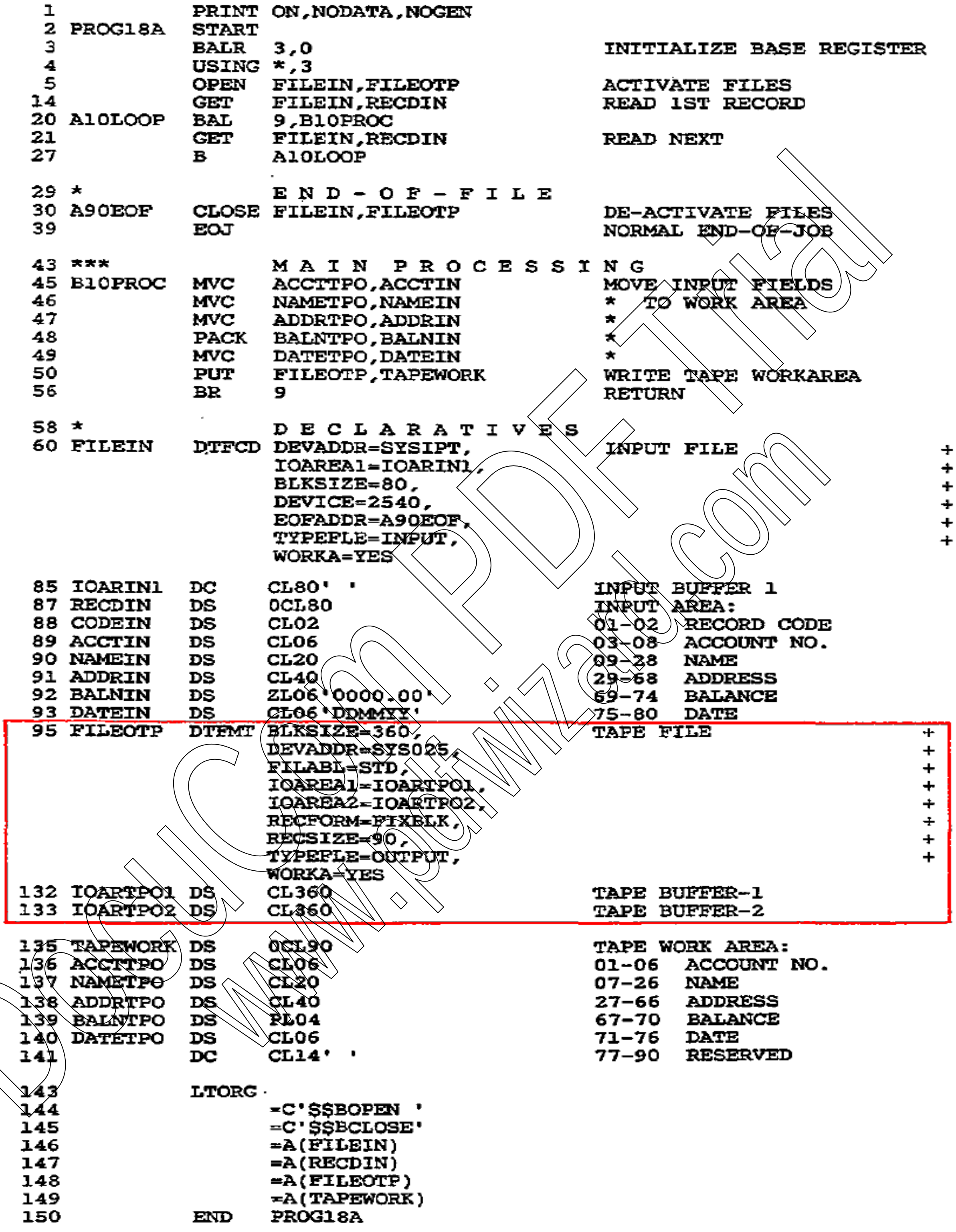
=ABE Abend (abnormal end of program execution), the standard default if you omit the entry.

ACC and SKP can use a SYNAD entry for printing an error message and continue processing. If the error message routine is named RIOTPERR, the DCB coding could be

```
EROPT=SKP,
SYNAD=Rl0TPERR
```

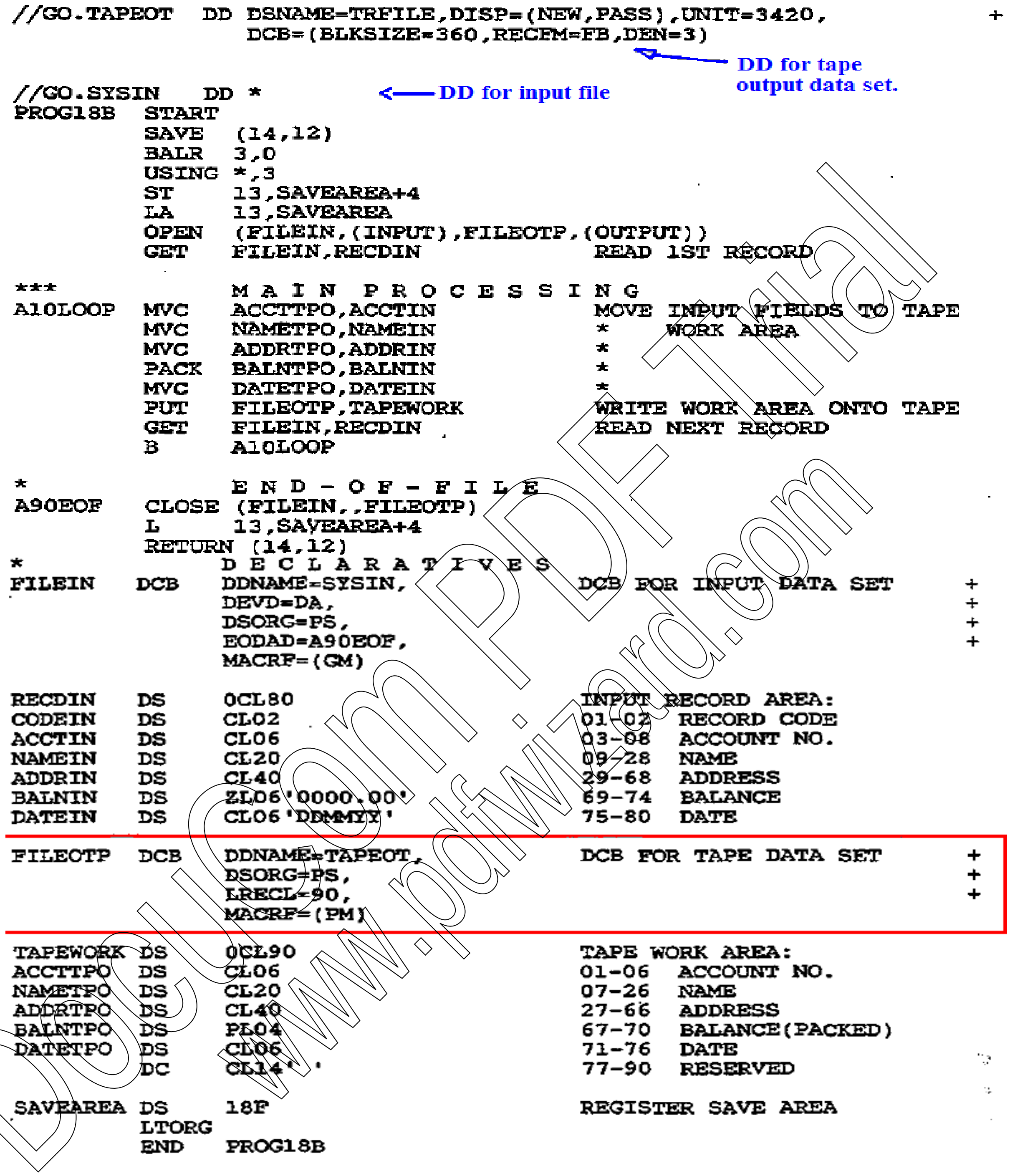
Since the use of ACC and SKP may cause invalid results, it may be preferable for important production jobs to use ABE (or allow it to default). See the OS supervisor manuals for other DCB options.

Figures 26–1 and 26–2 now follow, one per page.
The rest of this page is left blank.



```
  1           PRINT ON,NODATA,NOGEN
  2 PROG18A   START
  3           BALR  3,0                        INITIALIZE BASE REGISTER
  4           USING *,3
  5           OPEN  FILEIN,FILEOTP             ACTIVATE FILES
 14           GET   FILEIN,RECDIN              READ 1ST RECORD
 20 A10LOOP   BAL   9,B10PROC
 21           GET   FILEIN,RECDIN              READ NEXT
 27           B     A10LOOP

 29 *               END-OF-FILE
 30 A90EOF    CLOSE FILEIN,FILEOTP             DE-ACTIVATE FILES
 39           EOJ                              NORMAL END-OF-JOB

 43 ***             MAIN PROCESSING
 45 B10PROC   MVC   ACCTTPO,ACCTIN             MOVE INPUT FIELDS
 46           MVC   NAMETPO,NAMEIN             *  TO WORK AREA
 47           MVC   ADDRTPO,ADDRIN             *
 48           PACK  BALNTPO,BALNIN             *
 49           MVC   DATETPO,DATEIN             *
 50           PUT   FILEOTP,TAPEWORK           WRITE TAPE WORKAREA
 56           BR    9                          RETURN

 58 *               DECLARATIVES
 60 FILEIN    DTFCD DEVADDR=SYSIPT,            INPUT FILE              +
                    IOAREA1=IOARIN1,                                   +
                    BLKSIZE=80,                                        +
                    DEVICE=2540,                                       +
                    EOFADDR=A90EOF,                                    +
                    TYPEFLE=INPUT,                                     +
                    WORKA=YES

 85 IOARIN1  DC    CL80' '                    INPUT BUFFER 1
 87 RECDIN   DS    0CL80                      INPUT AREA:
 88 CODEIN   DS    CL02                       01-02 RECORD CODE
 89 ACCTIN   DS    CL06                       03-08 ACCOUNT NO.
 90 NAMEIN   DS    CL20                       09-28 NAME
 91 ADDRIN   DS    CL40                       29-68 ADDRESS
 92 BALNIN   DS    ZL06'0000.00'              69-74 BALANCE
 93 DATEIN   DS    CL06'DDMMYY'               75-80 DATE
 95 FILEOTP  DTFMT BLKSIZE=360,               TAPE FILE               +
                   DEVADDR=SYS025,                                    +
                   FILABL=STD,                                        +
                   IOAREA1=IOARTPO1,                                  +
                   IOAREA2=IOARTPO2,                                  +
                   RECFORM=FIXBLK,                                    +
                   RECSIZE=90,                                        +
                   TYPEFLE=OUTPUT,                                    +
                   WORKA=YES
132 IOARTPO1 DS    CL360                      TAPE BUFFER-1
133 IOARTPO2 DS    CL360                      TAPE BUFFER-2

135 TAPEWORK DS    0CL90                      TAPE WORK AREA:
136 ACCTTPO  DS    CL06                       01-06  ACCOUNT NO.
137 NAMETPO  DS    CL20                       07-26  NAME
138 ADDRTPO  DS    CL40                       27-66  ADDRESS
139 BALNTPO  DS    PL04                       67-70  BALANCE
140 DATETPO  DS    CL06                       71-76  DATE
141          DC    CL14' '                    77-90  RESERVED

143          LTORG
144                =C'$$BOPEN '
145                =C'$$BCLOSE'
146                =A(FILEIN)
147                =A(RECDIN)
148                =A(FILEOTP)
149                =A(TAPEWORK)
150          END   PROG18A
```

Figure 26 – 1 Program: Writing a Tape File under DOS



```
//GO.TAPEOT   DD DSNAME=TRFILE,DISP=(NEW,PASS),UNIT=3420,          +
                 DCB=(BLKSIZE=360,RECFM=FB,DEN=3)

//GO.SYSIN    DD *
PROG18B  START
         SAVE  (14,12)
         BALR  3,0
         USING *,3
         ST    13,SAVEAREA+4
         LA    13,SAVEAREA
         OPEN  (FILEIN,(INPUT),FILEOTP,(OUTPUT))
         GET   FILEIN,RECDIN               READ 1ST RECORD

***            M A I N  P R O C E S S I N G
A10LOOP  MVC   ACCTTPO,ACCTIN              MOVE INPUT FIELDS TO TAPE
         MVC   NAMETPO,NAMEIN              *    WORK AREA
         MVC   ADDRTPO,ADDRIN              *
         PACK  BALNTPO,BALNIN              *
         MVC   DATETPO,DATEIN              *
         PUT   FILEOTP,TAPEWORK            WRITE WORK AREA ONTO TAPE
         GET   FILEIN,RECDIN               READ NEXT RECORD
         B     A10LOOP

*              E N D - O F - F I L E
A90EOF   CLOSE (FILEIN,,FILEOTP)
         L     13,SAVEAREA+4
         RETURN (14,12)
*             D E C L A R A T I V E S
FILEIN   DCB  DDNAME=SYSIN,                DCB FOR INPUT DATA SET     +
              DEVD=DA,                                                +
              DSORG=PS,                                               +
              EODAD=A90EOF,                                           +
              MACRF=(GM)

RECDIN   DS   0CL80                        INPUT RECORD AREA:
CODEIN   DS   CL02                         01-02  RECORD CODE
ACCTIN   DS   CL06                         03-08  ACCOUNT NO.
NAMEIN   DS   CL20                         09-28  NAME
ADDRIN   DS   CL40                         29-68  ADDRESS
BALNIN   DS   ZL06'0000.00'                69-74  BALANCE
DATEIN   DS   CL06'DDMMYY'                 75-80  DATE

FILEOTP  DCB  DDNAME=TAPEOT,               DCB FOR TAPE DATA SET      +
              DSORG=PS,                                               +
              LRECL=90,                                               +
              MACRF=(PM)

TAPEWORK DS   0CL90                        TAPE WORK AREA:
ACCTTPO  DS   CL06                         01-06  ACCOUNT NO.
NAMETPO  DS   CL20                         07-26  NAME
ADDRTPO  DS   CL40                         27-66  ADDRESS
BALNTPO  DS   PL04                         67-70  BALANCE(PACKED)
DATETPO  DS   CL06                         71-76  DATE
         DC   CL14' '                      77-90  RESERVED

SAVEAREA DS   18F                          REGISTER SAVE AREA
         LTORG
         END  PROG18B
```

DD for tape output data set.

DD for input file

Figure 26 – 2 Program: Writing a Tape File under OS

CREATING A SEQUENTIAL DISK FILE

The next two examples create a disk file for DOS and OS. The programs accept input data from the system reader and write four records per block onto disk. For both programs, OPEN checks the disk label, and CLOSE writes the last data block (even if it contains fewer than four records) and writes a last dummy block with zero length.

DOS Program to Create a Sequential Disk File

The DOS file definition macro that defines a sequential disk file is DTFSD. The parameters that you code are similar to those for the DTFMT macro.

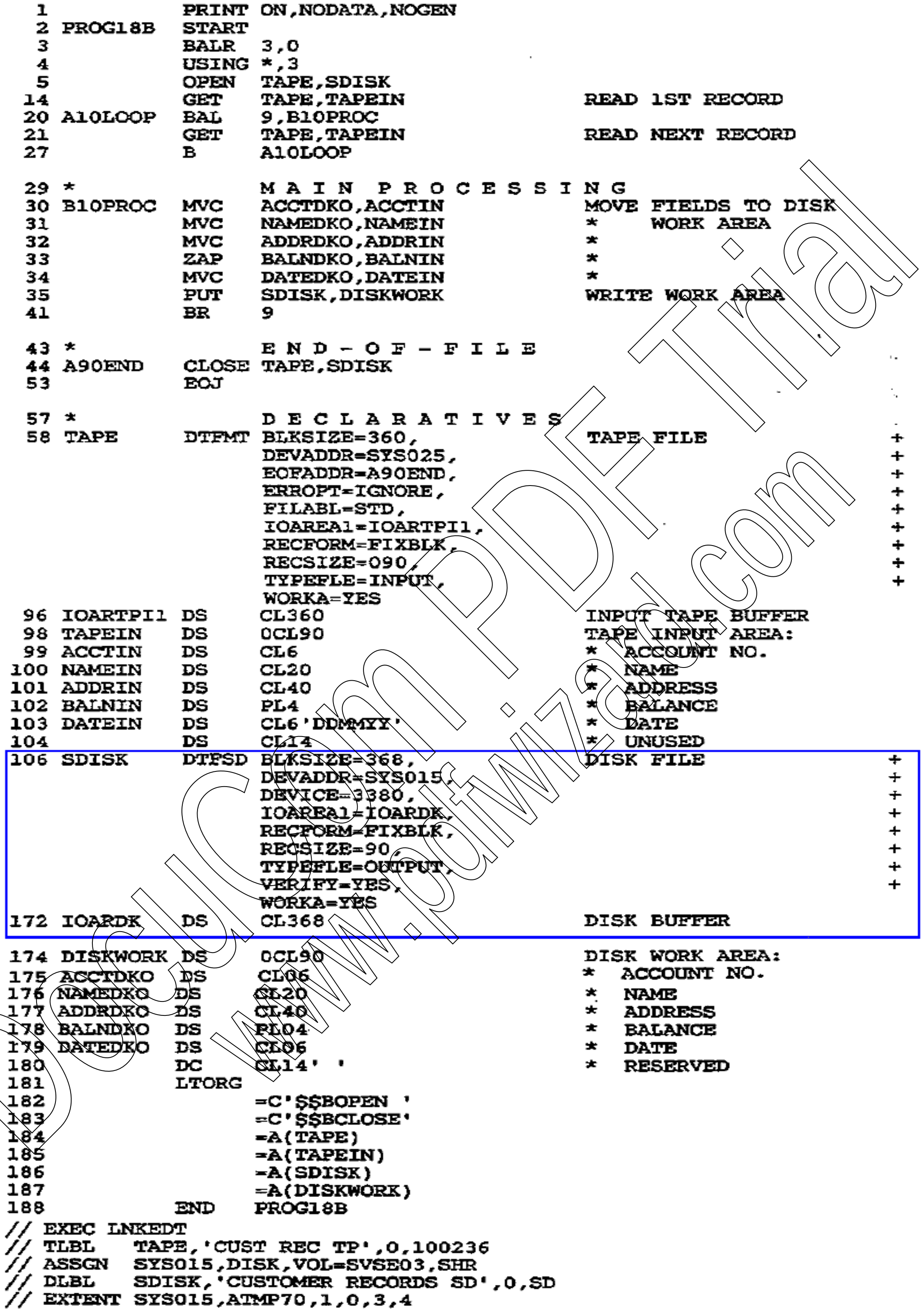
The program in Fig. 26–3 reads the tape records from the file created in Fig. 26–1 and transfers required fields to a disk work area named DISKWORK. The program then writes this work area named SDISK. Based on the BLKSIZE entry in the DTFMT and DTFSD, the system both reads and writes blocks of four records, though the blocking factor need not be the same. The following explains the DTFSD entries.

BLKSIZE=368 means that the block size for output is 360 bytes (4 x 90) plus 8 bytes for the system to construct a count field. You provide for the extra 8 bytes only for output; for input, the entry would be 360.

DEVICE= 3380 means that the program is to write blocks on a 3380 disk device.

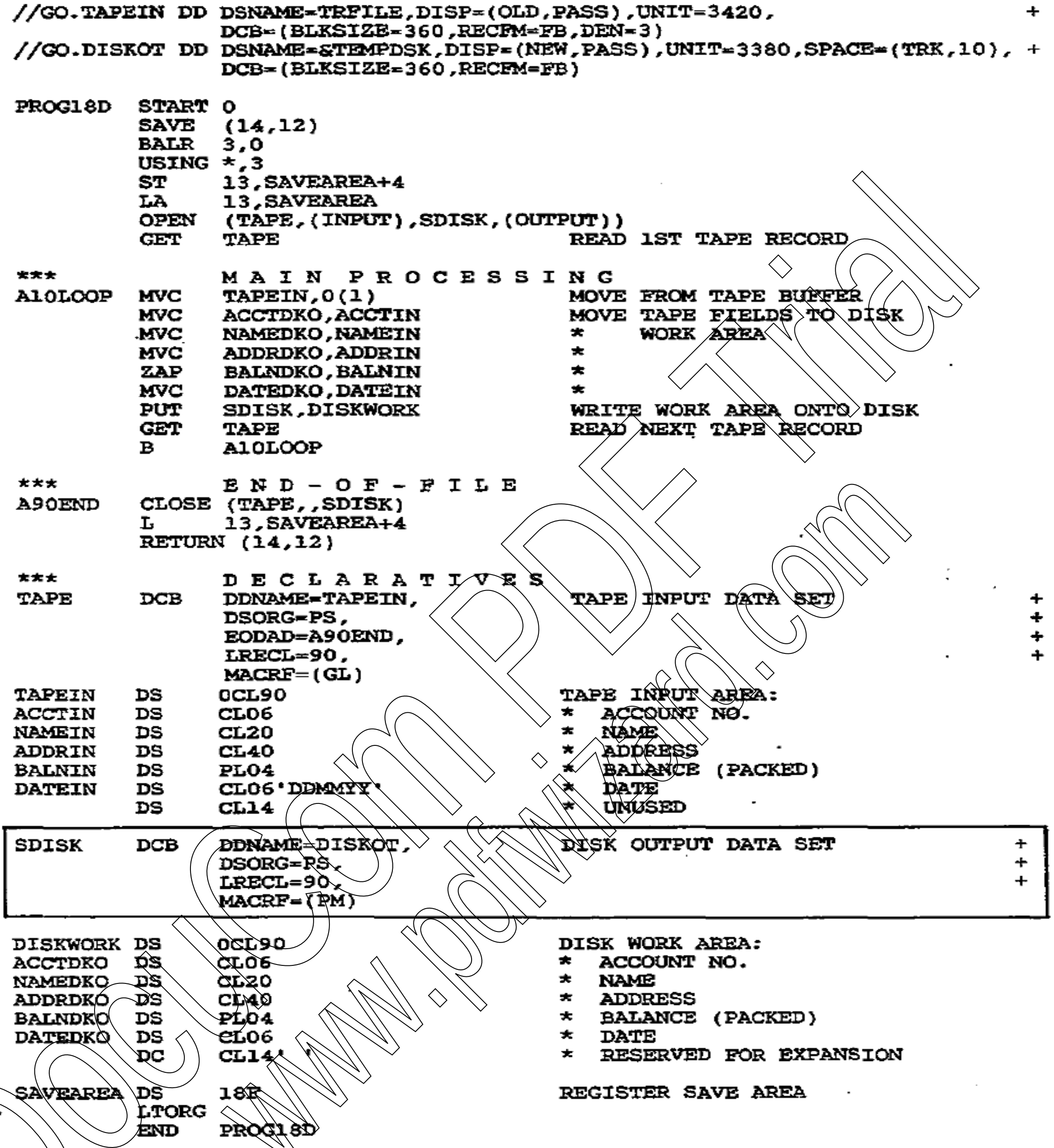
VERIFY = YES tells the system to reread each output record to check its validity. If the record when reread is not identical to the record that was supposed to be written, the system rewrites the record and performs another reread. If the system eventually cannot perform a valid write, it may advance to another area on the disk surface. Although this operation involves more accessing time, it helps ensure the accuracy of the written records.

DEVADDR, IOAREA1, RECFORM, RECSIZE, TYPEFLE, and **WORKA** are the same as for previous D1Fs. You omit the FILABL entry because disk labels must be standard. If you omit the entry for DEVADDR, the system uses the SYSnnn address from the job control entry.



```
  1              PRINT ON,NODATA,NOGEN
  2 PROG18B      START
  3              BALR  3,0
  4              USING *,3
  5              OPEN  TAPE,SDISK
 14              GET   TAPE,TAPEIN               READ 1ST RECORD
 20 A10LOOP      BAL   9,B10PROC
 21              GET   TAPE,TAPEIN               READ NEXT RECORD
 27              B     A10LOOP

 29 *                  M A I N  P R O C E S S I N G
 30 B10PROC      MVC   ACCTDKO,ACCTIN            MOVE FIELDS TO DISK
 31              MVC   NAMEDKO,NAMEIN            *    WORK AREA
 32              MVC   ADDRDKO,ADDRIN            *
 33              ZAP   BALNDKO,BALNIN            *
 34              MVC   DATEDKO,DATEIN            *
 35              PUT   SDISK,DISKWORK            WRITE WORK AREA
 41              BR    9

 43 *                  E N D - O F - F I L E
 44 A90END       CLOSE TAPE,SDISK
 53              EOJ

 57 *                  D E C L A R A T I V E S
 58 TAPE         DTFMT BLKSIZE=360,              TAPE FILE              +
                       DEVADDR=SYS025,                                  +
                       EOFADDR=A90END,                                  +
                       ERROPT=IGNORE,                                   +
                       FILABL=STD,                                      +
                       IOAREA1=IOARTPI1,                                +
                       RECFORM=FIXBLK,                                  +
                       RECSIZE=090,                                     +
                       TYPEFLE=INPUT,                                   +
                       WORKA=YES
 96 IOARTPI1     DS    CL360                     INPUT TAPE BUFFER
 98 TAPEIN       DS    0CL90                     TAPE INPUT AREA:
 99 ACCTIN       DS    CL6                       *  ACCOUNT NO.
100 NAMEIN       DS    CL20                      *  NAME
101 ADDRIN       DS    CL40                      *  ADDRESS
102 BALNIN       DS    PL4                       *  BALANCE
103 DATEIN       DS    CL6'DDMMYY'               *  DATE
104              DS    CL14                      *  UNUSED
106 SDISK        DTFSD BLKSIZE=368,              DISK FILE              +
                       DEVADDR=SYS015,                                  +
                       DEVICE=3380,                                     +
                       IOAREA1=IOARDK,                                  +
                       RECFORM=FIXBLK,                                  +
                       RECSIZE=90,                                      +
                       TYPEFLE=OUTPUT,                                  +
                       VERIFY=YES,                                      +
                       WORKA=YES
172 IOARDK       DS    CL368                     DISK BUFFER

174 DISKWORK DS        0CL90                     DISK WORK AREA:
175 ACCTDKO  DS        CL06                      *  ACCOUNT NO.
176 NAMEDKO  DS        CL20                      *  NAME
177 ADDRDKO  DS        CL40                      *  ADDRESS
178 BALNDKO  DS        PL04                      *  BALANCE
179 DATEDKO  DS        CL06                      *  DATE
180          DC        CL14' '                   *  RESERVED
181          LTORG
182                    =C'$$BOPEN '
183                    =C'$$BCLOSE'
184                    =A(TAPE)
185                    =A(TAPEIN)
186                    =A(SDISK)
187                    =A(DISKWORK)
188          END       PROG18B
// EXEC LNKEDT
// TLBL   TAPE,'CUST REC TP',0,100236
// ASSGN  SYS015,DISK,VOL=SVSE03,SHR
// DLBL   SDISK,'CUSTOMER RECORDS SD',0,SD
// EXTENT SYS015,ATMP70,1,0,3,4
```

Figure 26–3 Program: Writing a sequential disk file under DOS



```
//GO.TAPEIN DD DSNAME=TRFILE,DISP=(OLD,PASS),UNIT=3420,                     +
               DCB=(BLKSIZE=360,RECFM=FB,DEN=3)
//GO.DISKOT DD DSNAME=&TEMPDSK,DISP=(NEW,PASS),UNIT=3380,SPACE=(TRK,10),   +
               DCB=(BLKSIZE=360,RECFM=FB)

PROG18D  START 0
         SAVE  (14,12)
         BALR  3,0
         USING *,3
         ST    13,SAVEAREA+4
         LA    13,SAVEAREA
         OPEN  (TAPE,(INPUT),SDISK,(OUTPUT))
         GET   TAPE                        READ 1ST TAPE RECORD

***            M A I N   P R O C E S S I N G
A10LOOP  MVC   TAPEIN,0(1)                 MOVE FROM TAPE BUFFER
         MVC   ACCTDKO,ACCTIN              MOVE TAPE FIELDS TO DISK
         MVC   NAMEDKO,NAMEIN              *    WORK AREA
         MVC   ADDRDKO,ADDRIN              *
         ZAP   BALNDKO,BALNIN              *
         MVC   DATEDKO,DATEIN              *
         PUT   SDISK,DISKWORK              WRITE WORK AREA ONTO DISK
         GET   TAPE                        READ NEXT TAPE RECORD
         B     A10LOOP

***            E N D - O F - F I L E
A90END   CLOSE (TAPE,,SDISK)
         L     13,SAVEAREA+4
         RETURN (14,12)

***            D E C L A R A T I V E S
TAPE     DCB   DDNAME=TAPEIN,              TAPE INPUT DATA SET             +
               DSORG=PS,                                                   +
               EODAD=A90END,                                               +
               LRECL=90,                                                   +
               MACRF=(GL)
TAPEIN   DS    0CL90                      TAPE INPUT AREA:
ACCTIN   DS    CL06                       *  ACCOUNT NO.
NAMEIN   DS    CL20                       *  NAME
ADDRIN   DS    CL40                       *  ADDRESS
BALNIN   DS    PL04                       *  BALANCE (PACKED)
DATEIN   DS    CL06'DDMMYY'               *  DATE
         DS    CL14                       *  UNUSED

SDISK    DCB   DDNAME=DISKOT,             DISK OUTPUT DATA SET            +
               DSORG=PS,                                                  +
               LRECL=90,                                                  +
               MACRF=(PM)

DISKWORK DS    0CL90                      DISK WORK AREA:
ACCTDKO  DS    CL06                       *  ACCOUNT NO.
NAMEDKO  DS    CL20                       *  NAME
ADDRDKO  DS    CL40                       *  ADDRESS
BALNDKO  DS    PL04                       *  BALANCE (PACKED)
DATEDKO  DS    CL06                       *  DATE
         DC    CL14' '                    *  RESERVED FOR EXPANSION

SAVEAREA DS    18F                        REGISTER SAVE AREA
         LTORG
         END   PROG18D
```

Figure 26–4   Program: Writing a sequential disk file under OS

OS Program to Create a Sequential Disk File

For OS, you define a DCB macro with a unique name for each disk input or output file that the program processes. The parameters that you code are similar to those for the DCB macros covered earlier.

The program in Fig. 26–4 reads the tape records from the file created in Fig. 26–2 and transfers required fields to a disk work area named DISKWORK. The program then writes this work area to a disk output file named SDISK. Based on the BLKSIZE entry in job control, the system both reads and writes blocks of four records, although the two blocking factors need not be the same.

The DD entries for the disk file, DISKOT, are as follows:

DSNAME=&TEMPDSK provides the data set name.

DISP=(NEW,PASS) means that the file is new and is to be kept temporarily.

UNIT= 3380 provides the disk drive model.

SPACE= (TRK,10) allocates ten tracks for this file.

BLKSIZE= 360 means that each block to be written from the buffer is 360

bytes long, based on four records at 90 bytes each.

RECFM= FB defines output records as fixed-length and blocked. Records on disk may also be variable-length (V) or unblocked.

The following explains the DCB entries:

DDNAME=DISKOT relates to the same name in the DD job control command:

```
//GO.DISKOT
```

DSORG=PS defines output as physical sequential.

LRECL= 90 provides the logical record length for each record.

MACRF=(PM) defines the type of output operation as put and move from a work area. MACRF=(PL) would allow you to use locate mode to process records directly in the buffers.

The DCB file definition macro for disk input requires an entry EOFADDR=**address** to indicate the name of the routine where the system links on reaching the end of the disk file.

VARIABLE-LENGTH RECORDS

Tape and disk files provide for variable-length records, either unblocked or blocked. The use of variable-length records may significantly reduce the amount of space required to store a file. However, beware of trivial applications in which variations in record size are small or the file itself is small, because the system generates overhead that may defeat any expected savings. A record may contain one or more variable–length fields or a variable number of fixed–length fields.

1. ***Variable–Length Fields.*** For fields such as customer name and address that vary considerably in length, a program could store only significant characters and delete trailing blanks. One approach is to follow each variable field with a special delimiter character such as an asterisk.

The following example illustrates fixed-length name and address of 20 characters each, compressed into variable length with an asterisk replacing trailing blanks:

Fixed length: `Norman Bates        Bates Motel`

Variable length: `Norman Bates*Bates Motel*`

(ELB – Does anybody remember the Alfred Hitchcock movie ***Psycho***?)

To find the end of the field, the program may use a TRT instruction to scan for the delimiter. Another technique stores a count of the field length immediately preceding each variable-length field. For the preceding record, the count for the name would be 12 and the count for the address would be 11: |**12**|`Norman Bates|11|Bates Motel|`

2. *Variable Number of Fixed-Length Fields.* Records may contain a variable number of fields. For example, an electric utility company may maintain a large file of customer records with a fixed portion containing the customer name and address and optional subrecords for their electric account, natural gas account, and budget account.

VARIABLE·LENGTH RECORD FORMAT

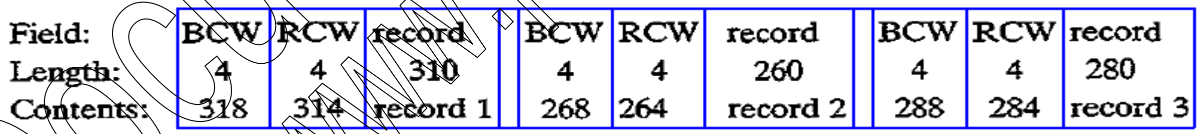
Immediately preceding each variable-length record on tape or disk is a 4-byte record control word (RCW) that supplies the length of the record. Immediately preceding each block is a 4-byte block control word (BCW) that supplies the length of the block. As a consequence, both records and blocks may be variable length. You have to supply a maximum block size into which the system is to fit as many records as possible.

Unblocked Records

Variable-length records that are unblocked contain a BCW and an RCW before each block. Here are three unblocked records:

```
|BCW|RCW|Record 1|•••|BCW|RCW|Record 2|•••|BCW|RCW|Record 3|
```

Suppose that three records are to be stored as variable-length unblocked. Their lengths are 310, 260, and 280 bytes, respectively:



Field:	BCW	RCW	record		BCW	RCW	record		BCW	RCW	record
Length:	4	4	310		4	4	260		4	4	280
Contents:	318	314	record 1		268	264	record 2		288	284	record 3

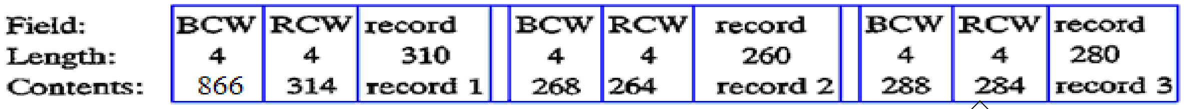
The RCW contains the length of the record plus its own length of 4. Since the first record has a length of 310, its RCW contains 314. The BCW contains the length of the RCW(s) plus its own length of 4. Since the only RCW contains a length of 314, the BCW contains 318.

Blocked Records

Variable-length records that are blocked contain a BCW before each block and an RCW before each record. The following shows a block of three records:

```
|BCW|RCW|Record 1|BCW|RCW|Record 2|BCW|RCW|Record 3|
```

Suppose that the same three records with lengths of 310, 260, and 280 bytes are to be stored as variable-length blocked and are to fit into a maximum block size of 900 bytes:



Field:	BCW	RCW	record		BCW	RCW	record		BCW	RCW	record
Length:	4	4	310		4	4	260		4	4	280
Contents:	866	314	record 1		268	264	record 2		288	284	record 3

The length of the block is the sum of one BCW, the RCWs, and the record lengths:

Block control word:	4	bytes
Record control words:	12	
Record lengths:	+ 850	
Total length:	866	bytes

The system stores as many records as possible in the block up to (in this example) 900 bytes. Thus a block may contain any number of bytes up to 900, and both blocks and records are variable length. The system automatically handles all blocking, unblocking, and control of BCWs.

Your BLKSIZE entry tells the system the maximum block length. For example, if the BLKSIZE entry in the preceding example specified 800, the system would fit only the first two records in the block, and the third record would begin the next block.

Programming for Variable-Length Records

Although IOCS performs most of the processing for variable–length records, you have to provide the record length. The additional programming steps are concerned with the record and block length:

Record length. As with fixed-length records, a program may process variable–length records in a work area or in the buffers (I/O areas). You define thework area as the length of the largest possible record; including the 4–byte record control word. When creating each record, calculate and store the record length in the record control word field. This field must be 4 bytes long, with the contents in binary format, as

```
VARRCW      DS F
```

DOS uses only the first 2 bytes of this field.

Block length. You define the I/O area as the length of the largest possible block, including the 4-byte block control word. On output, IOCS stores as many complete records in the block as will fit. IOCS performs all blocking and calculating of the block length. On input, IOCS deblocks, all records, similar to its deblocking of fixed-length records.

Sample Program: Reading and Printing Variable-Length Records

Consider a file of disk records that contains variable-length records, with fields defined as follows:

```
01-04          Record length
05-09          Account number
10-82          Variable name and address
```

To indicate the end of a name, it is immediately followed by a delimiter, in this case a plus sign (hex ‘4E’). Another delimiter terminates the next field, the address, and a third terminates the city. Here is a typical case:

```
JP Programmer+1425 North Basin Street+Kingstown+
```

The program in Fig. 26–5 reads and prints these variable-length records. Note that in the DTFSD, RECFORM= VARBLK specifies variable blocked. The program reads each input record and uses TRT and a loop to scan each of the three variable-length fields for the record delimiter. It calculates the length of each field and uses EX to move each field to the output area. The program also checks for the absence of a delimiter.

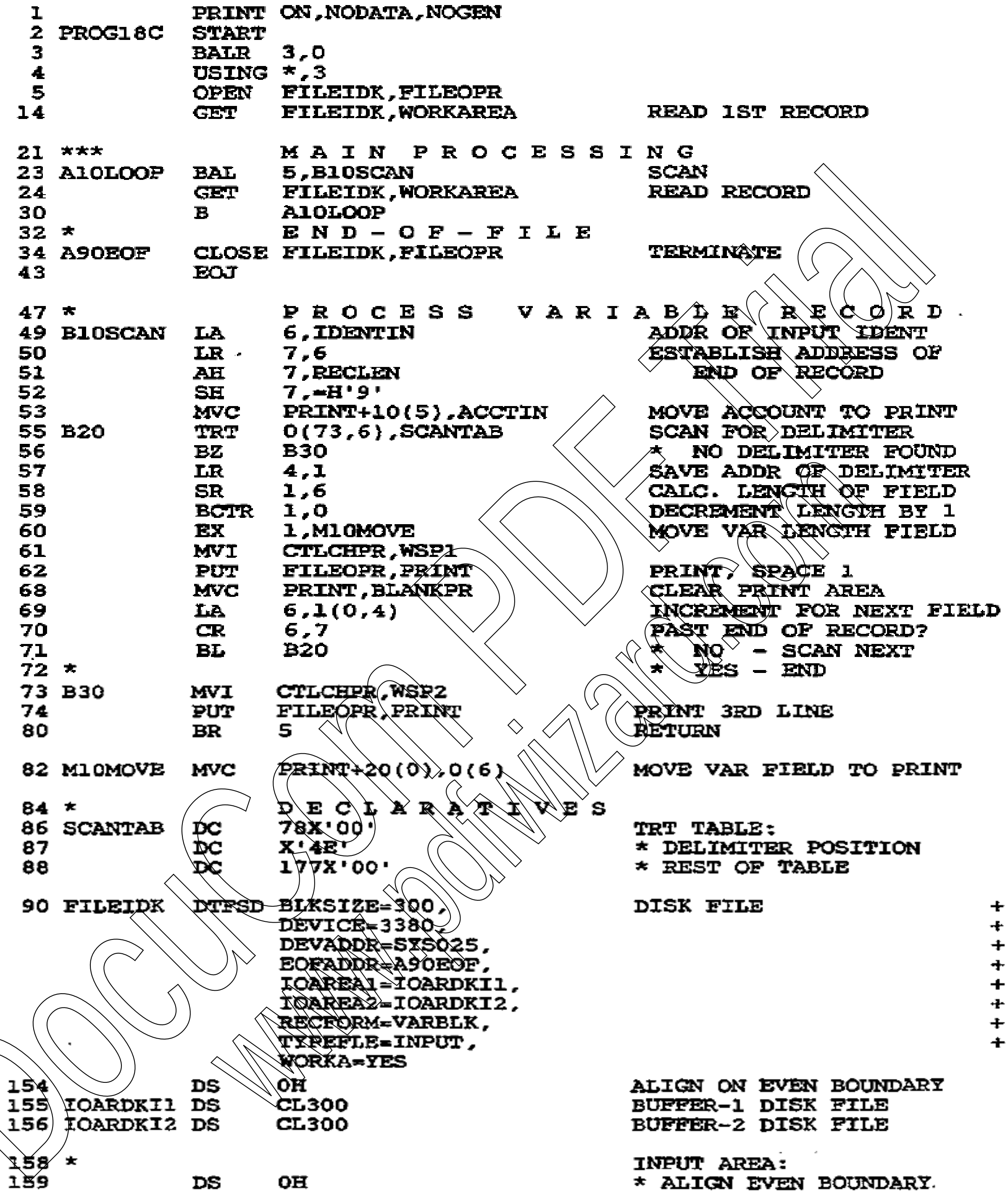
Output would appear as

```
          JP Programmer
          1425 Horth Basin Street
          Kingstown
```

The DTFSD omits RECSIZE because IOCS needs to know only the maximum block length. For OS, the DCB entry for variable blocked format is RECFM= VB. You could devise some records and trace the logic of this program step by step.

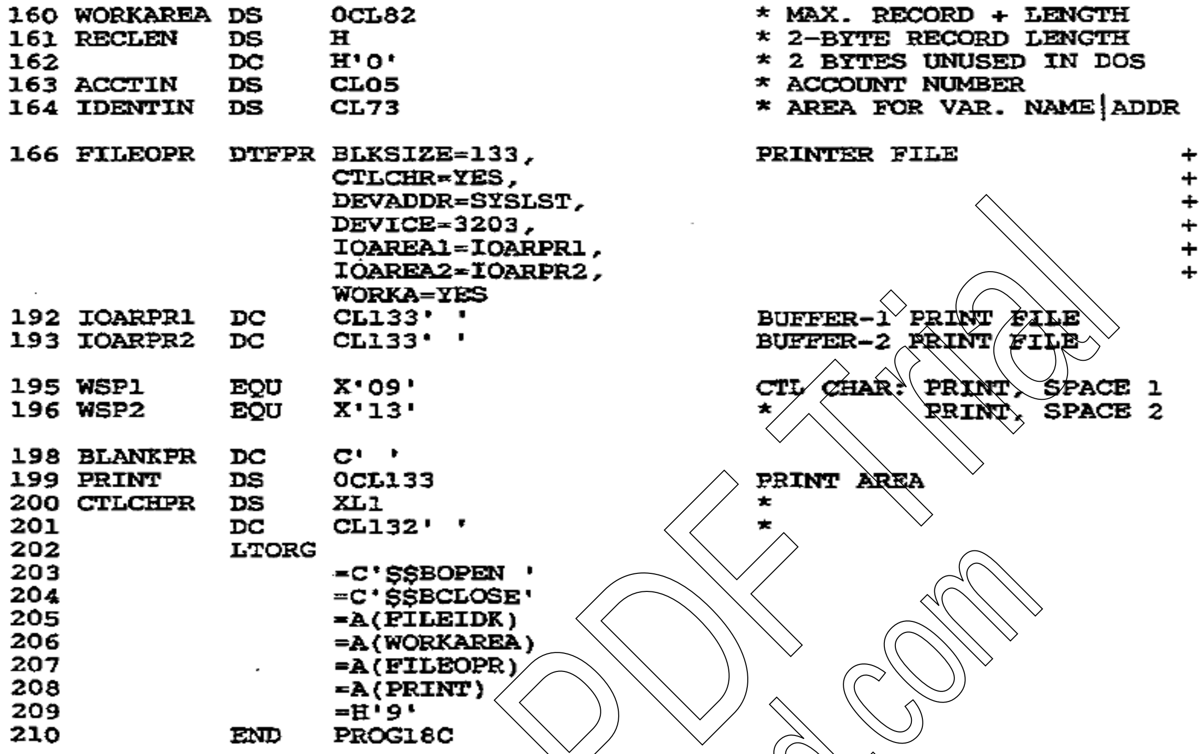
KEY POINTS

- Entries in a program file definition macro should match the job control commands.
- The block size for a file must be a multiple of record size, and all programs that process the file must specify the same record and block size.
- For variable-length files, the work areas and buffers should be aligned on an even boundary. When creating the file, you calculate and store the record length, whereas the system calculates the block length. Your designated maximum block size must equal or exceed the size of any record.



```
  1             PRINT  ON,NODATA,NOGEN
  2 PROG18C     START
  3             BALR   3,0
  4             USING  *,3
  5             OPEN   FILEIDK,FILEOPR
 14             GET    FILEIDK,WORKAREA          READ 1ST RECORD

 21 ***                M A I N   P R O C E S S I N G
 23 A10LOOP     BAL    5,B10SCAN                 SCAN
 24             GET    FILEIDK,WORKAREA          READ RECORD
 30             B      A10LOOP
 32 *                  E N D - O F - F I L E
 34 A90EOF      CLOSE  FILEIDK,FILEOPR           TERMINATE
 43             EOJ

 47 *                  P R O C E S S   V A R I A B L E   R E C O R D
 49 B10SCAN     LA     6,IDENTIN                 ADDR OF INPUT IDENT
 50             LR     7,6                       ESTABLISH ADDRESS OF
 51             AH     7,RECLEN                     END OF RECORD
 52             SH     7,=H'9'
 53             MVC    PRINT+10(5),ACCTIN        MOVE ACCOUNT TO PRINT
 55 B20         TRT    0(73,6),SCANTAB           SCAN FOR DELIMITER
 56             BZ     B30                       *  NO DELIMITER FOUND
 57             LR     4,1                       SAVE ADDR OF DELIMITER
 58             SR     1,6                       CALC. LENGTH OF FIELD
 59             BCTR   1,0                       DECREMENT LENGTH BY 1
 60             EX     1,M10MOVE                 MOVE VAR LENGTH FIELD
 61             MVI    CTLCHPR,WSP1
 62             PUT    FILEOPR,PRINT             PRINT, SPACE 1
 68             MVC    PRINT,BLANKPR             CLEAR PRINT AREA
 69             LA     6,1(0,4)                  INCREMENT FOR NEXT FIELD
 70             CR     6,7                       PAST END OF RECORD?
 71             BL     B20                       *  NO  - SCAN NEXT
 72 *                                            *  YES - END
 73 B30         MVI    CTLCHPR,WSP2
 74             PUT    FILEOPR,PRINT            PRINT 3RD LINE
 80             BR     5                        RETURN

 82 M10MOVE     MVC    PRINT+20(0),0(6)         MOVE VAR FIELD TO PRINT

 84 *                  D E C L A R A T I V E S
 86 SCANTAB     DC     78X'00'                  TRT TABLE:
 87             DC     X'4E'                    * DELIMITER POSITION
 88             DC     177X'00'                 * REST OF TABLE

 90 FILEIDK     DTFSD  BLKSIZE=300,             DISK FILE                +
                       DEVICE=3380,                                      +
                       DEVADDR=SYS025,                                   +
                       EOFADDR=A90EOF,                                   +
                       IOAREA1=IOARDKI1,                                 +
                       IOAREA2=IOARDKI2,                                 +
                       RECFORM=VARBLK,                                   +
                       TYPEFLE=INPUT,                                    +
                       WORKA=YES
154             DS     0H                       ALIGN ON EVEN BOUNDARY
155 IOARDKI1    DS     CL300                    BUFFER-1 DISK FILE
156 IOARDKI2    DS     CL300                    BUFFER-2 DISK FILE

158 *                                           INPUT AREA:
159             DS     0H                       * ALIGN EVEN BOUNDARY.
```

Figure 26–5 Program: Printing variable–length records



```
160 WORKAREA DS     0CL82                    * MAX. RECORD + LENGTH
161 RECLEN   DS     H                        * 2-BYTE RECORD LENGTH
162          DC     H'0'                     * 2 BYTES UNUSED IN DOS
163 ACCTIN   DS     CL05                     * ACCOUNT NUMBER
164 IDENTIN  DS     CL73                     * AREA FOR VAR. NAME|ADDR

166 FILEOPR  DTFPR BLKSIZE=133,              PRINTER FILE           +
                   CTLCHR=YES,                                      +
                   DEVADDR=SYSLST,                                  +
                   DEVICE=3203,                                     +
                   IOAREA1=IOARPR1,                                 +
                   IOAREA2=IOARPR2,                                 +
                   WORKA=YES
192 IOARPR1  DC     CL133' '                 BUFFER-1 PRINT FILE
193 IOARPR2  DC     CL133' '                 BUFFER-2 PRINT FILE

195 WSP1     EQU    X'09'                    CTL CHAR: PRINT, SPACE 1
196 WSP2     EQU    X'13'                    *         PRINT, SPACE 2

198 BLANKPR  DC     C' '
199 PRINT    DS     0CL133                   PRINT AREA
200 CTLCHPR  DS     XL1                      *
201          DC     CL132' '                 *
202          LTORG
203                 =C'$$BOPEN '
204                 =C'$$BCLOSE'
205                 =A(FILEIDK)
206                 =A(WORKAREA)
207                 =A(FILEOPR)
208                 =A(PRINT)
209                 =H'9'
210          END    PROG18C
```

Figure 26–5 Program: Printing variable–length records (Continued)