KEK-TOPAZ-OFFLINE-86-4

TRISTAN BANK SYSTEM

User Manual *

Version 4.1

April 1986
TOPAZ Offline Group

*Original manual written by ATF of FACOM was converted to LaTeX document by A. Miyamoto (*e-mail address:miyamoto@kekux1.kek.jp*) on June 1994 for use of TBS on UNIX system.

Contents

1 Overview of the TRISTAN BANK SYSTEM

When we analyze experimental data, we have to handle many kinds of data, for example, experimental raw data and calibration constants. Raw data consists of a begin run record, many event records and an end run record and they are saved in the magnetic tape sequentially. Data like calibration constants are stored in a disk file and used for correcting the raw data of all event records. If we can handle these different kinds of data sets by only one data handling system, it might be very useful. The purpose of this program package is to be a good tool for data handling and data communication among different sub-detectors and among different analysis stages.

Data are stored in a *data buffer* of the common /TBSBUF/. This data buffer is divided into several *data sub-buffers* (*DSB*). The number of DSBs is defined by the user. The maximum number of DSBs is 20.

The data of the bank system is organized by a tree structure of three levels, i.e. **record**, **bank** and **bank element**. One record is stored in one DSB. A record consists of many banks and each bank consists of many bank elements. Input/output of the bank data from/to a mass storage device is done by the record unit. The kind of record is identified by the record name. In the previous example, there are four kinds of records and their names may be "Begin Run", "Event", "End Run" and "Calibration constants". Since each kind of record requires one DSB, there are four DSBs. The allocation of a DSB to a record is done by calling routine TBALOC.

It might be required ,because several kinds of records are not used at a time, that only one DSB is assigned to them for the sake of buffer space economy. In our example, the begin run record and the end run record are those kinds of records. For this purpose we support a time sharing use of a DSB, by which one DSB can be used by several different records at different times. The usage of a DSB is defined by routines TBALOC and TBSHAR as described in section 2. This definition of usage should be done only one time at the beginning of a job. A templet for our example is shown in Appendix B.

The input/output data of this bank system consists of two I/O records, one is for the control block and another for the data block. The control block contains the record name, number of banks, bank names, number of elements and element numbers, etc. The data block is the image of data in the DSB. When we read the data from a mass storage device, first the control block is read in. If the record name in the control block coincides with any record names defined by TBALOC and TBSHAR, the control block data are copied to a labelled common /TBSBNK/. The data block,which comes after the control block, is copied into the DSB, which is allocated by TBALOC or defined to be a time sharing DSB by TBALOC and TBSHAR. The content of the common /TBSBNK/ is described in Appendix A.

It may happen that several kinds of records are not needed in the current analysis job but required in later analysis. In such a case, we write these records on the output device as well as the result of the current analysis. For this purpose, the system has a facility which passes the records, defined by TBPASS, directly to the output device.

After initialization of the bank system, the user program steps into an analysis loop. At the beginning of each loop, each DSB should be initialized except for the DSB where data like calibration constants are saved. The initialization of the DSB is done by

TBINIT or by reading bank data into a DSB by TBREAD. It should be kept in mind that the DSB, defined as a time sharing one, cannot be initialized by TBINIT, but can be initialized only by TBREAD. The description of DSB initialization is given in section 3.

After reading data into a DSB or initialization of DSBs, banks can be created and deleted and any data can be put/get into/from a bank element. Routines to handle data are described in section 4. Output of a record to a mass storage device is always possible as long as the record is in a DSB. After the output of a record, the contents of the record in a DSB are not altered. Thus user can use them unless they are replaced by a new one or initialized by TBREAD or TBINIT.

The system also has listing routines, by which a user can print information about the data buffer, DSB, bank, or bank elements. These listing routines are described in section 6.

2 Initialization of the Bank system

2.1 Data Buffer

A labelled common `/TBSBUF/` is used for the *data buffer* of the bank system. Prior to using the bank system, the user should define the common `/TBSBUF/` and its size in I*4 word units. To *define the size of data buffer*, call TBSIZE.

```
PARAMETER ( NSIZE = 100000 )
COMMON /TBSBUF/ IBUFF( NSIZE )

CALL TBSIZE( NSIZE )
```

In the bank system, the data buffer can be divided into several *data sub-buffers*.

2.2 Data Sub-Buffer(DSB) and Record

To allocate a DSB to a record is done by calling a subroutine TBALOC as follows:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      CALL TBALOC( RECNAME, IDALOC, MXBNK, MXELM, NWALOC, IRET )
--------------------------------------------------------------
 (Input)
      RECNAME = Record name to which the DSB is allocated.
      IDALOC  = ID number of the record
      MXBNK   = Maximum number of banks in the record
      MXELM   = Maximum number of bank elements in the record
      NWALOC  = Allocated size of the record
 (Output)
      IRET    = 0 ; Normal return. In any other case, the program
                    will be stopped.
 (Caution)
```

```
      When a user gives a record name previously allocated,
      the program will be stopped.
```

When several records are defined, none of these record names and record ID numbers should be duplicated. When any record name or ID number is already in use, this call will be neglected. The maximum number of DSBs is 20.

The input/output data of this bank system consists of two blocks, one is the control block and another is the data block. When we read the data by TBREAD, the control block is read first. The read routine looks for the record name in the allocated record name table, which coincides with the name in the control block. If there is such a record name, the next data block is copied to the allocated DSB. If not, the read routine skips the record and reads the next control block. One advantage of this bank system is that we can define several independent DSBs. We can use one DSB for raw data and other DSBs for several independent calibration constants. In this way, raw data changes event by event, but constants can be fixed throughout an analysis job.

When the size of the input data is larger than allocated, the size of the DSB will be extended. The size of extension unit for a DSB is defined by the routine TBEXTD.

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBEXTD( IDALOC, NWEXTD, IRET )
   --------------------------------------
 (Input)
      IDALOC  = Record ID of the allocated record to a DSB by TBALOC.
      NWEXTD  = Size of extension unit
 (Output)
      IRET = 1 : Normal return
      IRET < 0 : Warning.  This call is neglected.
            -1 ; No such record ID is found.
            -2 ; extension size is too large.
```

If the user does not define the extension unit for a DSB, the DSB is extended by the amount of space required. When the extension of a DSB occurs, the other DSBs, located after the extended DSB in the data buffer, are moved backward just by the amount extended. This DSB move is time consumming.

It is sometimes required, because several kinds of data are not used at a time, that only one DSB is assigned to these data for the sake of buffer space economy. For example, TOPAZ on-line data have several types of records, like begin run record, monitor record, end run record and event record, etc. These records are copied into a mass storage device in the order they were generated. We desired to assign one DSB for the event record and other one DSB for the begin run record, monitor record and end run record. For this purpose the following routine TBSHAR is prepared:

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBSHAR( IDALOC, RECSHR, IDSHR, IRET )
   ---------------------------------------------
 (Input)
      IDALOC  = Record ID of the allocated record to a DSB by TBALOC.
```

```
      RECSHR  = Record name whose DSB is used by time sharing with
                the allocated record.  The maximum number of sharing
                records is 20 per one DSB.
      IDSHAR  = Record ID whose DSB is used by time sharing with
                the allocated record.
 (Output)
      IRET = 1 : Normal return
      IRET < 0 : Warning.  This call is neglected.
            -1 ; No such record ID is found.
            -2 ; Record name has already been defined.
            -3 ; Record ID has already been used.
            -4 ; Too many DSBs were defined to be time sharing.
            -5 ; Too many records were assigned to a DSB.
```

The maximum number of DSBs for time sharing use is 10. If the record name RECSHR or IDSHAR have already defined, this call is neglected.

It may happen that a kind of record is not needed in the current analysis job but required in later analysis. In this case we read data and write them immediately to the output file. This is possible by using routines TBREAD and TBWRIT in principle. But the routine TBREAD unpacks the control block data and TBWRIT packs them again. Instead of this we recommend using the data passing function. Use of the routine TBPASS is the following:

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBPASS( PASSNAM, LUOUT, IRET )
   --------------------------------------
 (Input)
      PASSNAM = Record name which is passed to the file of the logical
                unit number LUOUT. ( up to 64 characters )
      LUOUT   = Logical unit number of the output file.
 (Output)
      IRET = 1 : Normal return
      IRET < 0 : Warning.  This call is neglected.
            -1 ; The record name exists already.
            -2 ; Too many kinds of records defined as passed one.
```

When the record name PASSNAM is the same as any of the record names prior to this call, this call is neglected. The maximum number of passed record names is 20.

3 Initialization of Data Sub-buffer and Data Read

After allocation of a DSB to a kind of record, the DSB should be initialized. The initialization of the DSB can be done in two different ways, one by calling TBINIT and the other by calling TBREAD.

3.1 Data Read

Subroutine TBREAD can be used to read the existing bank data from a file. The calling sequence of the routine is,

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   CALL TBREAD( LUNIT, IRET )
 ------------------------------
(Input)
     LUNIT   = Logical unit number of the input device.
(Output)
     IRET = ID    : Normal return. IRET gives the input record ID.
          = -1    : Read error.
          = -1000 : EOF found.
```

The bank data on the mass storage device consists of a control-block and a data-block. The data-block is an image of the data on a DSB. The control-block has information to control the data-block, i.e. record name, bank-names and pointer tables etc. A pair of a control-block and data-block is stored in the separate I/O records, thus two I/O records correspond to one record of the bank system. The read routine TBREAD reads the control-block first to see the record name of the data. If that record name is allocated to a DSB or is one of the shared record names which use one time sharing DSB, the control-block is unpacked to prepare the proper pointers and bank names for the record. Then the data-block is read into the relevant DSB and the return flag IRET is set equal to the record ID number. If no DSB is matched to the input record name, the data-block is skipped. Then the next control-block is read and tested until the input record name matches to a DSB.

3.2 DSB Initialization routine

The DSB can be initialized by the subroutine TBINIT, which is used when creating banks and putting data into bank elements, instead of reading data by TBREAD. The calling sequence of TBINIT is:

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   CALL TBINIT( IDALOC, IRET )
 -------------------------------
(Input)
     IDALOC  = Record ID of the allocated record to a DSB by TBALOC.
(Output)
     IRET = 1 : Normal return.
     IRET < 0 : Warning. This call is neglected.
           -1 ; No such record ID is found.
           -2 ; The queried DSB is for the time sharing use.
```

However, there is a limitation for TBINIT that if the DSB is defined to be used by more than two types of records, the DSB cannot be initialized by TBINIT. Such a DSB can be initialized only by reading data to the DSB by TBREAD.

3.3 Get the Status of a Record

To get the status of a record, a routine TBSTAT is prepared, by which the following information can be obtained:

1. Is the record with queried record ID currently on a DSB ?
2. If the record is on a DSB, the number of created banks and number of bank elements are obtained as well as the size of free space in the DSB.

The calling sequence of the routine TBSTAT is

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBSTAT( IDREC, NOBANK, NOELM, NFREE )
   ---------------------------------------------
 (Input)
      IDREC   = The queried record ID
 (Output)
      NOBANK  = Number of banks in the record.
                If the record does not exist on any DSB, NOBANK is -1.
      NOELM   = Number of bank elements in the record
      NFREE   = Size of the free space in the DSB in I*4 word unit.
```

3.4 Query Record ID

It is sometime useful to know the record ID number associated with record name. For this purpose a routine TBQREC is supported. The usage is

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBQREC( RECNAM, IDREC )
   -------------------------------
 (Input)
      RECNAM  = The queried record name ( 64 characters at maximum)
 (Output)
      IDREC   = The record ID of the queried name.
              = -1 ; No such record name is found.  This call
                     is neglected.
```

4 Data Access Routines

After initialization of DSB, the bank data can be accessed. The bank data is organized by a tree structure of three levels. The record consists of several banks and each bank consists of several bank elements. Access to the bank system is to create/delete a bank or bank element and to read/write data from/into a bank element.

4.1 Creation of a Bank

A user should create a bank before use unless the bank exists in the record. A bank is created by routine TBCRTE.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   CALL TBCRTE( IDREC,'Bank_name', ISALOC, ISEXTD, IRET )
-----------------------------------------------------------
(Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
     ISALOC      = primary allocation size (I*4) of the bank element
     ISEXTD      = Extension size unit (I*4)
(Output)
     IRET = 1 : Normal return
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; The bank with same name exists already.
           -3 ; Too many banks
```

When an element of a bank is first created, the system allocates ISALOC I*4 words of space in the DSB regardless of the actual data size to be written. Next time when a user appends the data to this element, those data will be written in the same area unless the total size of the element exceeds the allocation size. If the total size exceeds the allocation size, the system extends the element size by the unit of IEXTD I*4 words until the size is enough to contain the data. If there exists another element data adjacent the specified element, the entire data of that element will be copied to the end of the last element data in the DSB. At this moment, if there is not enough space at the end of the last element in the DSB, gabage collection for the DSB is done. Even after gabage collection, if there is still not enough space for that element data, this DSB will be extended as long as the data buffer has enough free space. If the data buffer has no more free space, the bank system will be terminated because of lack of space.

4.2 Other routines for the Bank

To delete a bank the routine TBDELB is avialable, calling sequence is the following:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   CALL TBDELB( IDREC,'Bank_name', IRET )
-------------------------------------------
(Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
(Output)
     IRET >=0 : Normal return
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
```

```
          -2 ; No such bank is found.
          -3 ; That bank could not be deleted by any reason.
```

To rename a bank from the old name to a new name is done by calling TBRNAM.

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      CALL TBRNAM( IDREC,'Old_ name','New_name', IRET )
   -----------------------------------------------------
 (Input)
      IDREC      = Record ID
     'Old_name'  = Old_bank_name up to 64 characters
     'New_name'  = New_bank_name up to 64 characters
 (Output)
      IRET = 1 : Normal return
      IRET < 0 : Warning.  This call is neglected.
            -1 ; No such record ID is found.
            -2 ; No old name bank is found.
            -3 ; The new name bank exists already.
            -4 ; Rename error for the lack of the bank name table.
```

To get the record name, number of banks, and bank names of a record, the routine TBQBNK is available. The calling sequence is

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      CALL TBQBNK( IDREC, RECNAM, NOBANK, BNAME )
   -----------------------------------------------
 (Input)
      IDREC      = Record ID
 (Output)
      RECNAM     = Record name ( CHARACTER*64 )
      NOBANK     = Number of banks in the record.
                   If the record does not exist on any DSB, NOBANK = 0
      BNAME      = Array of CHARACTER*64, containing bank names in
                   the record.
```

To delete those banks, which have no bank element, the routine TBSQEZ is prepared. Its usage is

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      CALL TBSQEZ( IDREC, IRET )
   --------------------------------
 (Input)
      IDREC      = Record ID
 (Output)
      IRET >=1 : Normal return. IRET gives the number of banks deleted.
      IRET < 0 : Warning.  This call is neglected.
            -1 ; No such record ID is found.
```

4.3 Put data to a Bank element

To put data to a bank element, several routines are available. They are are TBPUT, TBPUTH, TBAPND, TBAPDH, TBMOD and TBMODH. The routines TBPUT, TBAPND and TBMOD are used for putting I*4 word data to the bank element, while the routines TBPUTH, TBANDH and TBMODH are for I*2 word data.

For routines TBPUT and TBPUTH, the calling sequences are as follows:

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBPUT( IDREC,'Bank_name', NE, NWORDS, IDATA, IRET )
  ------------------------------------------------------------
 (Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
     NE          = element number
     NWORDS      = Number of I*4 words in the input data IDATA
     IDATA       = Input data
 (Output)
     IRET > 0 : Normal return
            1 ; Bank element was created.
            2 ; Element data was replaced.
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; No such bank is found.

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBPUTH( IDREC,'Bank_name', NE, NWORDS, IDATA, IRET )
  ------------------------------------------------------------
 (Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
     NE          = element number
     NWORDS      = Number of I*2 words in the input data IDATA
     IDATA       = Input data
 (Output)
     IRET > 0 : Normal return
            1 ; Bank element was created.
            2 ; Element data was replaced.
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; No such bank is found.
```

If these put routines are called more than two times, the data of the last call is kept, because new data overwrite the old data.

The routines TBAPND and TBAPDH are used to append data to a bank element. Even if an element has no data, these routines are also available to put the first data to an element instead of TBPUT and TBPUTH.

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBAPND( IDREC,'Bank_name', NE, NWORDS, IDATA, IRET )
  -------------------------------------------------------------
(Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
     NE          = element number
     NWORDS      = Number of I*4 words in the input data IDATA
     IDATA       = Input data
(Output)
     IRET > 0 : Normal return
            1 ; Bank element was created.
            2 ; Element data was appended.
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; No such bank is found.

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBAPDH( IDREC,'Bank_name', NE, NWORDS, IDATA, IRET )
  -------------------------------------------------------------
(Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
     NE          = element number
     NWORDS      = Number of I*2 words in the input data IDATA
     IDATA       = Input data
(Output)
     IRET > 0 : Normal return
            1 ; Bank element was created.
            2 ; Element data was appended.
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; No such bank is found.
```

It is possible to replace part of the element data by calling TBMOD or TBMODH.

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBMOD( IDREC,'Bank_name', NE, IADD, NWORDS, IDATA, IRET )
  ------------------------------------------------------------------
(Input)
     IDREC       = Record ID
    'Bank_name'  = 64 characters at maximum
     NE          = element number
     IADD        = relative address (I*4 unit) in the element data
     NWORDS      = Number of I*4 words in the input data IDATA
     IDATA       = Input data
```

```
(Output)
     IRET = 1 : Normal return
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; No such bank is found.
           -3 ; No such bank element is found.
           -4 ; The relative start address is out of range.
           -5 ; The relative end address is out of range.
```

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBMODH( IDREC,'Bank_name', NE, IADD, NWORDS, IDATA, IRET )
  -------------------------------------------------------------------
(Input)
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
     NE         = element number
     IADD       = relative address (I*2 unit) in the element data
     NWORDS     = Number of I*2 words in the input data IDATA
     IDATA      = Input data
(Output)
     IRET = 1 : Normal return
     IRET < 0 : Warning.  This call is neglected.
           -1 ; No such record ID is found.
           -2 ; No such bank is found.
           -3 ; No such bank element is found.
           -4 ; The relative start address is out of range.
           -5 ; The relative end address is out of range.
```

4.4 Get Data from a Bank element

For getting data from a bank element, the routines TBGET, TBGETH, TBPICK and TBPIKH are prepared. The routines TBGET and TBPICK are for I*4 word data, while the routines TBGETH and TBPIKH are for I*2 word data. To get the full element data, TBGET or TBGETH is useful. But, if the data size is large and only a part of the data is needed, TBPICK or TBPIKH is recommended.

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBGET( IDREC,'Bank_name', NE, NWORDS, IDATA, IRET )
  ------------------------------------------------------------
(Input)
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
     NE         = element number
(Output)
     NWORDS     = Number of I*4 words of the output data IDATA
     IDATA      = Output data
```

```
    IRET        = 1 : Normal return
    IRET        < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
                 -2 ; No such bank is found.
                 -3 ; No such bank element is found.
```

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    CALL TBGETH( IDREC,'Bank_name', NE, NWORDS, IDATA, IRET )
 ------------------------------------------------------------
(Input)
    IDREC       = Record ID
   'Bank_name'  = 64 characters at maximum
    NE          = element number
(Output)
    NWORDS      = Number of I*2 words of the output data IDATA
    IDATA       = Output data
    IRET        = 1 : Normal return
    IRET        < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
                 -2 ; No such bank is found.
                 -3 ; No such bank element is found.
```

To pick a part of the element data, the routines TBPICK and TBPIKH are available.

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    CALL TBPICK( IDREC,'Bank_name', NE, IADD, NWORDS, IDATA, IRET )
 -------------------------------------------------------------------
(Input)
    IDREC       = Record ID
   'Bank_name'  = 64 characters at maximum
    NE          = element number
    IADD        = relative address (I*4) of the part to be picked
                  up in the element data
    NWORDS      = Word length (I*4) of the part
(Output)
    IDATA       = Output data
    IRET        = 1 : Normal return
    IRET        < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
                 -2 ; No such bank is found.
                 -3 ; No such bank element is found.
                 -4 ; The relative start address is out of range.
                 -5 ; The relative end address is out of range.
```

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    CALL TBPIKH( IDREC,'Bank_name', NE, IADD, NWORDS, IDATA, IRET )
```

```
  -------------------------------------------------------------------
(Input)
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
     NE         = element number
     IADD       = relative address (I*2) of the part to be picked
                  up in the element data
     NWORDS     = Word length (I*2) of the part
(Output)
     IDATA      = Output data
     IRET       = 1 : Normal return
     IRET       < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
                 -2 ; No such bank is found.
                 -3 ; No such bank element is found.
                 -4 ; The relative start address is out of range.
                 -5 ; The relative end address is out of range.
```

4.5 Get Location of a Bank element in the Data buffer

To get/put data from/to a bank element, TBGET and TBPUT etc. are available. But, in each of these routines, data copy from/to the input/output data to/from the data buffer is made. It is, of course, time comsumming. The reason why we transfer the input/output data through a subroutine argument is just for data protection. However, if a user does not directly change the data in the data buffer but refers to them directly, the data will be well protected and fewer data copies will be made. For this purpose, data of an element are designed to occupy a continuous region in the data buffer. There are two ways to get the location of an element in the data buffer, one is to get the location in a local variable and the other is in a global variable. The former is done by the following routine TBLOC:

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBLOC( IDREC,'Bank_name', NE, ATT, LOC, NWORDS, IRET )

  ---------------------------------------------------------------
(Input)
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
     NE         = element number
     ATT        = Attribute of 'I*2' or 'I*4'
(Output)
     LOC        = Absolute address of the first word of the element
                  data in units of I*2 or I*4 depending on ATT.
     NWORDS     = Word length of the element data in units of I*2
                  or I*4 depending on ATT.
     IRET       = 1 : Normal return
     IRET       < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
```

```
                  -2 ; No such bank is found.
                  -3 ; No such bank element is found.
                  -4 ; Illegal attribute
```

The location of the bank element may be changed by the gabage collectin, which occurs when an enough continuous space in the DSB cannot be obtained for an element to which data is put or appended. Thus, before every use of the data buffer, the above routine should be called.

When the location and data size of an element are used in many places, calling TBLOC each time is not economical. For this case, we can define a global location and word-length variables by the following way:

```
      COMMON /POINTR/ LOC,NWORDS
       ..........
      IDREC  = 1
      NE     = 1
       .........
      CALL TBLINK( IDREC,'Test_Bank', NE, 'I*2', LOC, NWORDS, IRET )
       .........
```

After calling TBLINK, the values of LOC and NWORDS always give the location and word length of the element NE of the "Test_Bank" in the record IDREC as long as the record is in the DSB. When, however, a new record is read into the same DSB, the old data is destroyed and the value -1 is put into the variable NWORDS. Therefore, before using these global location variables, the user should check the word length. If the word length is a negative value, the global location variable is not available for the moment. The calling sequence of TBLINK is the following:

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      CALL TBLINK( IDREC,'Bank_name', NE, ATT, LOC, NWORDS, IRET )

   ----------------------------------------------------------------
 (Comment)
      The variables LOC and NWORDS are recommended to be labelled
      common variables.  There is no restriction about the name of
      labelled common.
 (Input)
      IDREC       = Record ID
     'Bank_name'  = 64 characters at maximum
      NE          = element number
      ATT         = Attribute of 'I*2' or 'I*4'
      LOC         = memory location of the variable LOC.
      NWORDS      = memory location of the variable NWORDS.
 (Output)
      LOC         = Absolute address of the first word of the element
                    data in the unit of I*2 or I*4 depending on ATT.
      NWORDS      = Word length of the element data in the unit of
                    I*2 or I*4 depending on ATT.
```

```
     IRET       = 1 : Normal return
     IRET       < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
                 -2 ; No such bank is found.
                 -3 ; No such bank element is found.
                 -4 ; Illegal attribute
                 -5 ; This bank element is already linked.
```

4.6 Other Routines for the Bank Element

To delete a bank element, call TBDELE. The calling sequence is as follows:

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBDELE( IDREC,'Bank_name', NE, IRET )
  ----------------------------------------------
 (Input)
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
     NE         = element number
 (Output)
     IRET       = 1 : Normal return
     IRET       < 0 : Warning.  This call is neglected.
                 -1 ; No such record ID is found.
                 -2 ; No such bank is found.
                 -3 ; No such bank element is found.
```

The routine TBNOEL gives the number of elements and their element numbers for a bank. This is used in the following way:

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBNOEL( IDREC,'Bank_name', NOELM, NEARAY )
  ---------------------------------------------------
 (Input)
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
 (Output)
     NOELM >=0  : Number of bank elements in the bank
           < 0  : There is no such a bank.
     NEARAY     = Array containing the element number of the each
                  element.
```

To obtain the data size of a bank element, the routine TBQWEH is useful. Its usage is

```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     CALL TBQEWH( IDREC,'Bank_name', NE, NWORDS )
  ------------------------------------------------
 (Input)
```

```
     IDREC      = Record ID
    'Bank_name' = 64 characters at maximum
     NE         = element number
(Output)
     NWORDS     = Size of the bank element in I*2 word units.
                  If there is no such a bank element, NWORDS =< 0
```

5 Write Bank Data

To output data of the bank system to a mass storage device, the routine TBWRIT is prepared, whose usage is as follows:

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      CALL TBWRIT( LUNIT, IDREC, IRET )
   --------------------------------------
 (Input)
      LUNIT      = Logical unit number of the output device
      IDREC      = Record ID
 (Output)
      IRET      >= 0 : Number of output words in I*4 unit
      IRET       < 0 : Warning.  This call is neglected.
                  -1 ; No such record ID is found.
                  -2 ; Write error
               -1000 ; End of file
```

After writing a record by TBWRIT, the record data on a DSB is not altered. The user can still get element data from the record. When the return code after TBREAD is a positive number and coincides with a valid record ID, the record data on the relevant DSB is replaced by the new data.

Even if a bank has no element, its bank information is written on the output file by TBWRIT. If such bank information is not required on the output file, the routine TBSQEZ in section 4.2 should be called before calling TBWRIT. After calling TBSQEZ, those banks, which have no bank element, are deleted.

6 Other Utility Routines

When we debug a user program, it is sometime useful to dump the contents of the bank data. For this purpose, the listing routines in this section are prepared.

6.1 Listing Routines

To list the status of the data buffer and all record names allocated to or sharing DSBs, the routine TBLSUB is available.

```
~~~~~~~~~~~~~~~~~

   CALL TBLSUB

-----------------
(Output)
     (1) Print status of the data buffer
           a. Number of DSBs allocated
           b. Allocated size of each DSB and allocated numbers of
              banks and bank elements in each DSB
           c. Used size of each DSB and used numbers of banks and
              bank elements in each DSB
     (2) Print a table of record names allocated to or sharing DSBs
         and the record name, currently on DSB, is marked by 'ON'.
```

By calling TBLBNM, the bank name table, its sort table and bank information table are printed for all records currently on DSBs. Its usage is as follows:

```
~~~~~~~~~~~~~~~~~

   CALL TBLBNM

-----------------
(Output)
         Print contents of ISORT, KEYBNK, number of elements and
        bank name for each bank in each record currently on DSBs.
        To know the meaning of ISORT etc., see appendix A.
```

To list the link, element key and address tables, call the routine TBLLNK.

```
~~~~~~~~~~~~~~~~~

   CALL TBLLNK

-----------------
(Output)
         Print contents of LINK, KEYELM and IADTBL for each bank
        element in each record currently on DSBs.
```

To dump element data, there are two different routines. One is TBLEL0, which dumps the data in five different attributes- I*4, I*2, R*4, hexadecimal and character. Another is TBLEL1, which dumps the data according to a given attribute.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   CALL TBLEL0( IDREC, 'Bank_name', NE, MXWORD)

------------------------------------------------
(Input)
     IDREC      = Record ID number
    'Bank_name' = Bank name
     NE         = Element number of the bank element
     MXWORD     = Number of words to be printed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
    CALL TBLEL1( IDREC, 'Bank_name', NE, MXWORD, ATT )
 ------------------------------------------------------
(Input)
    IDREC       = Record ID number
   'Bank_name'  = Bank name
    NE          = Element number of the bank element
    MXWORD      = Number of words to be listed.
                  When MXWORD = 0, list all words.
    ATT         = Attribute type: 'I*4','I*2',R*4','A*4' or 'HEXA'
```

6.2 Other Routines

The errors in the bank system are categorized into two levels, one is warning and the other fatal error. When a fatal error occurs, the system prints the error message and stops the program. The logical unit number of the printing device for error messages can be defined by the following routine TBLUNI.

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~
    CALL TBLUNI( LUNIT )
 --------------------------
(Input)
    LUNIT       = Logical unit number of the printing device
```

The user can suppress printing of warning messages by routine TBWARN.

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~
    CALL TBWARN( LEVEL )
 --------------------------
(Input)
        LEVEL =< 0 : suppress printing messages except fatal one.
    0 < LEVEL < 10 : suppress printing warning messages
        LEVEL >=10 : print all messages
```

Appendix

A Contents of common TBSBNK

```
 common /TBSBNK/
 PARAMETER       (MAXSUB= 20, MXSHR= 10, MAXBNK= 200, MAXELM= 4000)
 INTEGER*2       IDSUB, ISORT, KEYBNK, LINK
 COMMON /TBSBNK/ MAXSIZ, NWFREE, NOSUB, NOSHR
.,               NWSUB(3,MAXSUB),IDSUB (12,MAXSUB)
.,               ISORT (MAXBNK), NWBNK(2,MAXBNK), KEYBNK(2,MAXBNK),
.,               LINK(3,MAXELM), NWELM(2,MAXELM), IADTBL(3,MAXELM)
 CHARACTER*64    SUBNAM, BNKNAM, SHRNAM
 COMMON /TBSNAM/ SUBNAM(MAXSUB),BNKNAM(MAXBNK),SHRNAM(MAXSUB,MXSHR)
    Required Buffer size ( in Bytes)
      740*MAXSUB + 74*MAXBNK + 26*MAXELM + 16
    For     (20)       (200)      (4000)        = 133616 Bytes
   a. MAXSIZ            = Size of the Data_Buffer (I*4)
   b. NWFREE            = Word_Count of the free space (I*4)
   c. NOSUB             = Number of the Data_Sub_Buffers (I*4)
   d. SUBNAM(MAXSUB)  : DSB_Name_Table in Character*64 format
   e. NOSHR             = Number of the Shared DBS's     (I*4)
   f. SHRNAM(MAXSUB,MXSHR) : Record names in the shared DBS
   g. NWSUB(3,MAXSUB) : DSB_Word_Count_Table
        NWSUB(1,j)      = Offset of the j-th Data_Sub_Buffer (DSB)
        NWSUB(2,j)      = Size of the j-th DSB
        NWSUB(3,j)      = Number of wrods in the j-th DSB
   h. IDSUB(12,MAXSUB) : DSB_Parameter_Table
        IDSUB(1,j)      = ID of the j-th DSB
        IDSUB(2,j)      = Offset in Bank_Name_Table: BNKNAM,KEYBNK
                          of the j-th DSB.
        IDSUB(3,j)      = Maximum number of Banks in the j-th DSB
        IDSUB(4,j)      = Number of Banks in the j-th DSB
        IDSUB(5,j)      = Offset in Element_Number_List: LISTEL
                          of j-th Data_Sub_Buffer
        IDSUB(6,j)      = Maximum number of Bank_elements in the
                          j-th DSB
        IDSUB(7,j)      = Number of Bank_elements in the j-th DSB
        IDSUB(8,j)      = Flag for Gabage_Collection:(0=Not Needed)
        IDSUB(9,j)      = When this is a shared DBS, the number of
                          sharing record names is given.
        IDSUB(10,j)     = When this is a shared DSB, the sequential
                          number of the shared DSB is given.
        IDSUB(11,j)     = DSB status
            + < 0       : Nothing on the DSB
```

```
        ! = 0        : The record defined by TBALOC is on DSB.
        + > 0        : One of shared records is on the DSB
                       When this is a shared DSB, the pointer to
                       record name table is given, where the
                       current record name on this DSB is given.
     IDSUB(12,j)     = Unit of extension size
                         If this number is zero, size of extension
                         is determined by the just one required.
i. ISORT(MAXBNK)   : Sort_Table
                       The pointers to the Bank_Name_Table are
                       stored in the sorted order.
j. BNKNAM(MAXBNK)  : Bank_Name_Table in Character*64 format
k. NWBNK(2,MAXBNK) : Number of word counts for the Bank
     NWBNK(1,j)      = Allocated # of words for the j-th Bank
     NWBNK(2,j)      = # of words for extension
l. KEYBNK(2,MAXBNK): Bank_Key_Table
     KEYBNK(1,j)     = Address of the first Element of the Bank
                       in the Link_Table: LINK, Eelement_Table:
                       NWELM and Address Table: IADTBL
     KEYBNK(2,j)     = Number of elements in the Bank BNKNAM(j)
m. LINK(2,MAXELM)  : Link_Table
     LINK(1,j)       = (-1 / anyother ) = ( Unused/ Element # )
     LINK(2,j)       = Give the address of the next Element.
     LINK(3,j)       = Attribute of the element def'd by TBLINK
n. NWELM(2,MAXELM) : Words count for the Bank element
     NWELM(1,j)      = Allocated # of words in the Bank_Element
     NWELM(2,j)      = Number of words in the Bank_Element
o. IADTBL(3,MAXELM): Address_Table
     IADTBL(1,j)     = address of the Bank_Element in the
                       Data_Buffer.
     IADTBL(2,j)     = Address of the pointer variable
     IADTBL(3,j)     = Address of the word variable
                       If users define a pointer and word
                       variables by calling TBLINK(...),
                       addresses of the pointer and word
                       variables are saved in IADTBL(2/3,j).
                       These variables are always updated
                       after these values are changed.
```

B An example

```
PARAMETER ( NSIZE = 100000)
COMMON /TBSBUF/ IBUFF( NSIZE )
DATA LUCAL/ 10/, LURAW/ 20/, LUOUT/ 30/
```

```
      CALL TBSIZE( NSIZE )
      CALL TBALOC( 'Calibration constants', 1,  10,  20,  2000, IRET )
      CALL TBALOC( 'Begin Run',             2,  10,  20,  2000, IRET )
      CALL TBALOC( 'Event',                 4, 100, 200, 50000, IRET )
      CALL TBSHAR( 2, 'End Run', 3, IRET )

      CALL TBREAD( LUCAL, IDRET )
      IF( IDRET .EQ. -1000 ) THEN
        STOP  'NO CALIBRATION CONSTANTS'
      ENDIF

 1000 CALL TBREAD( LURAW, IDRET )
      IF( IDRET .EQ. -1000 ) THEN
        STOP  'END OF DATA'
      ELSEIF( IDRET .EQ. 2 ) THEN
         :::::::::::::::::::::::::::::  Handle the begin run record
      ELSEIF( IDRET .EQ. 3 ) THEN
         :::::::::::::::::::::::::::::  Handle the end run record
      ELSEIF( IDRET .EQ. 4 ) THEN
         :::::::::::::::::::::::::::::  Handle the event record
      ENDIF

      CALL TBWRIT( LUOUT, IDRET )

      GO TO 1000

      END
```