ОБЪЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ

Б 1-11-84-380

ДЕПОНИРОВАННАЯ ПУБЛИКАЦИЯ

Дубна 1984.

JOINT INSTITUTE FOR NUCLEAR RESEARCH

Laboratory of Nuclear Problem

FARAGO Istvan

RISK DATA PROCESSING UTILITIES

Dubna 1984

This publication discusses the capabilities of the RISK experiment data processing utility programs and the control statements used with each program. These programs are used by programmers responsible for organizing and maintaining the experiment's data.

The information contained in this publication is subject to significant change. Any such changes will be published in new editions or RISK internal reports. Before using this publication, consult the latest internal report of RISK experiment data processing, to learn the editions that are applicable and current.

The author does not accept any responsibility for loss or damage arising from the use of information contained in any of his reports or in any communication about his tests or investigations.

Faragó István

May, 1984

~~30.05.84~~
22.06.84

51-11-84-380

Prerequisite publication

OS Sort/Merge Program, for a reference to sorting techniques, control statement formats, program operations, the inclusion of the user-written routines, efficient use of the program and the description of the program generated messages.

Recommended publications

OS Messages and Codes, which contains a complete listing and explanation of the messages and codes issued by the IBM OS operating system components.

OS JCL Reference, which contains a complete explanation of the job control statements available for the OS operating system.

OS Utilities, which contains a full description of the use of the IBM OS operating system utility programs.

This publication assumes that the reader is familiar with the IBM S/36o OS operating system terms and concepts.

Notational Conventions

A uniform system of notation is used to describe the syntax of utility control statements. This notation provides a basis for describing the structure of utility control statements. That is, it describes which parameters are required and which are optional, the options available in expressing values, and the required punctuation.

Bold Type: in the notation, bold type /LIST, Q, etc./ is used to indicate specific values that can be entered.

Italic Type: /nn, user-information, etc./ is used where a number, character string, or keyword is to be inserted by the user.

Punctuation: the period/./, comma/,/, equal sign/=/, and apostrophe /'/ are used for punctuation and must be coded as shown. These punctuation marks serve to separate the parameters of a utility control statement.

Brackets: /[ ]/ indicate that the elements and punctuation they enclose are optional. The brackets themselves are for descriptive purposes only, and are not to be coded.

Braces: /{}/ indicate a required choice. The braces themselves are for descriptive purposes only, and are not to be coded.

Underscoring: indicates a value that is assumed by the program if no value is entered for that element.

Ellipsis: /.../ is used to indicate that one or more additional parameters or sets of parameters, each of the same format, optionally can be added to the operand.

Restriction

Unless otherwise indicated in the description, a temporary data set can be processed only if the user specifies the complete name generated for the data set by the system /for example, DSNAME=SYS84oo5.Toooool.RPool.JOBTEMP.EXMAPLE/.

Multiprogramming Considerations

In an MVT environment, a region size should be specified for each application of a utility program. The region size is determined by the number of bytes in the utility program and by the block sizes of the data sets used in the job step. A region size can be specified as a parameter in the EXEC job control statement specifying the utility program name.

A job that modifies a system data set /identified by SYSn./ must be run in a single job environment; however, a job that uses a system data set, but does not modify it, can be run in a multiprogramming environment. The operator should be informed of all jobs that modify system data sets.

The DD job control statements should ensure that the volumes on which the data sets reside cannot be shared when update activity is being performed.

Control

The utility programs are controlled by job control statements and utility control statements. The job control statements and utility control statements necessary to use utility programs are provided in the major discussion of each utility program.

Job Control Statements

A utility program can be introduced to the operating system in different ways:

- Job control statements can be included in the input stream.
- Job control statements, placed in a procedure library or defined as an in-stream procedure, can be included via the EXEC job control statement.
- A utility program can be invoked by a calling program.

If job control statements are placed in a procedure library, they should satisfy the requirements for most applications of the program; a procedure, of course, can be modified or supplemented for applications that require additional parameters, data sets, or devices. The IBM S/360 OS data-set utility program IEBUPDTE can be used to enter a procedure into a procedure library.

Utility Control Statements

Utility control statements are used to identify a particular function to be performed by a utility program.

The dynamic data structure of HYDRA.

The basic unit of the data structure is a thing called a "bank". It is an ordered collection of numbers belonging to or describing an object, physical or ideal. You may have a bank carrying the parameters of a particular track, or a bank describing the properties of a particular module of your detector, or also a bank carrying all the parameters and counters of a histogram. All banks are held in one FORTRAN common array Q called the "dynamic store". If L is the address of a particular bank its first number is addressed as Q(L+1).

The simplest structure to be built from these units is the "linear structure", a collection of several banks of the same kind, for example, all tracks of an event, all the histograms in a particular run. To materialize this grouping of like banks in the computer, each such bank has a reserved word containing the pointer to the next bank, with the last bank having zero in this word, indicating that there is no next bank. The pointer to a bank is simply the address of that bank in the dynamic store.

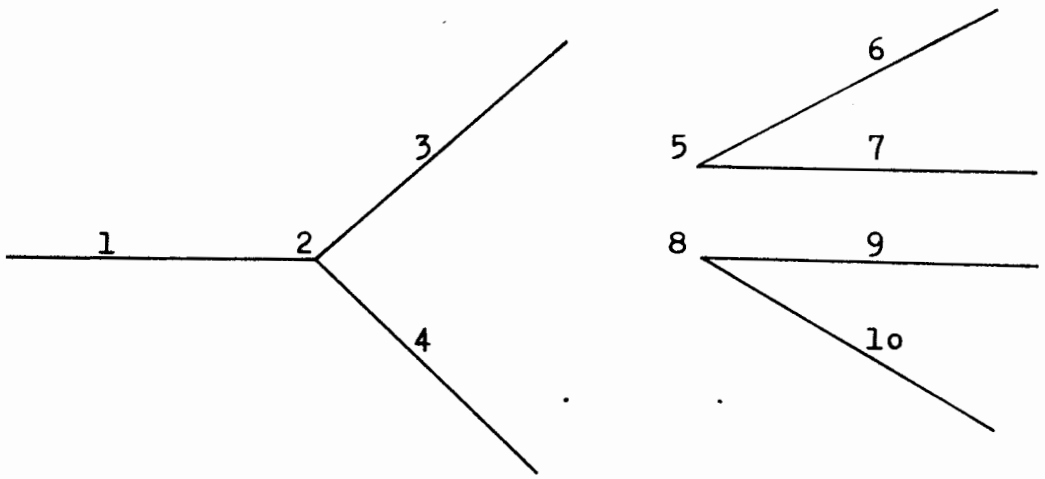
For many types of information this simplest structure is all that is needed, obviously so for the example of the histograms. But for many other cases one needs a more sophisticated structure, allowing relations between banks of different kinds. The fundamental relation "dependence of existence" is defined in the data structure and supported throughout the HYDRA system routines. For example, if a bank EV represents an event and several banks VX represent the vertices of this event, one will build a

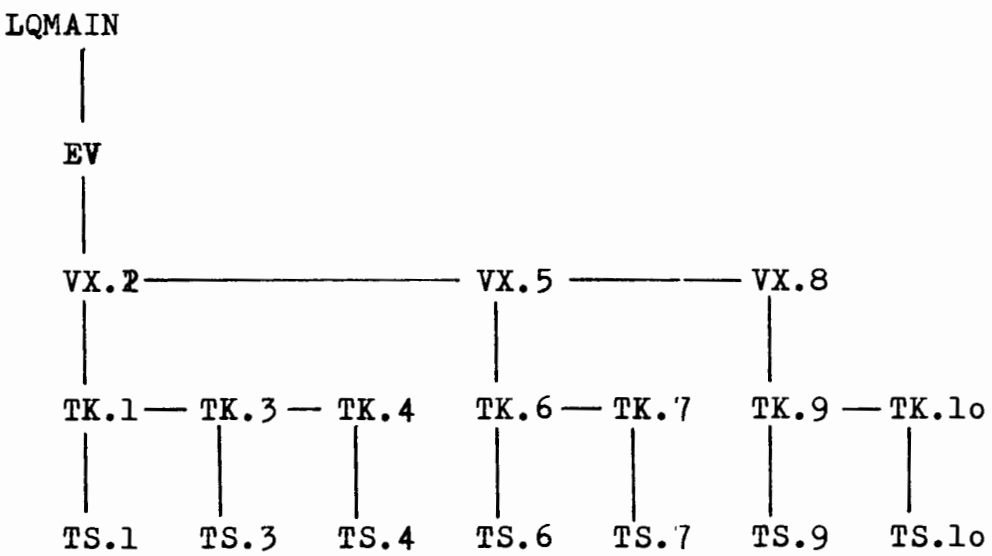
data-structure in which the VX banks depend from the EV bank in their existence, the EV bank having a reserved word containing the pointer to the first bank of the linear structure of VX banks. A request then to the HYDRA system to write the event to tape, for example, propagates through this pointer from the EV bank to all VX banks of the event.

The HYDRA system allows for pointers from any bank to any other bank to be handled by the user as he needs to. The system handles these links in a straight-forward way. In the jargon these pointers are called "reference links", in opposition to the two kinds of "structural links" used to interconnect the banks into a data structure.

In the full sense, a data structure is the collection of data plus all their logical inter-relations.

We can visualize the building the data-structures. Take as an example an event of particle interaction and decay: along the track 1 comes the primary particle colliding with a stationary particle at vertex 2, giving rise to the two charged particles 3 and 4 and to two neutral particles, which decay at vertices 5 and 8. The information about the vertices we keep in VX banks, about the tracks in TK banks with measured points in the track segment TS banks, in a structure like this:





The event-bank EV "supports" the whole structure, its address kept in LQMAIN, a permanent link at the beginning of LQ which is equivalenced with Q dynamic store FORTRAN common array. One link of this EV bank supports the VX banks /other links may support other sub-structures not shown above, e.g. fiducial bank/. This link points to the first VX bank, holding the information about the vertex 2. This VX bank points with one link to the next VX bank, and this in turn to the next one.

Each VX bank supports with another link the banks TK for its associated tracks. We use here the notation VX.8 and TK.4, for example, to indicate merely on the paper the VX bank for vertex 8, and the TK bank for the track 4. In the computer only the bank ID's VX and TK are real. Any designators or labels for individual banks, if at all needed, are defined and handled by the user as he pleases.

Although this example shows the problem which historically helped trigger the creation of HYDRA, it is given here strictly as an example. Looking at this example we notice the following properties: The "linear structure" appears at each level except the highest. The vertices VX form a linear structure, and the TK form three linear structures, one for each vertex. Normally, the order of the banks in a linear structure is of no importance. By convention link 1 is reserved for connenting linear structures; it is used to step through the banks from one to next of the same kind, link 1 of the last bank contains zero because for it there is no "next".

Of a nature very different from the "horizontal" connection by link 1 is the "vertical" connection is mediated by link 2 of the VX bank. This link 2 materializes the relation "dependence of existence", i.e., if a given VX bank is to be written out to the output tape, very likely the TK's have to go with it.

Transfer data structures

FQXOUT transfers a data structure from the dynamic store to a sequential file, FQXIN does the inverse transfer. These transfers are done with standard unformatted FORTRAN statements READ and WRITE. In high-energy physics applications most FQX data sets contain data structures representing events.

Associated with each data structure is the "d/s header" containing user-defined identifying information, to allow rapid selection of data structures.

On the external data set, a data structure is represented by a straight copy of the relevant sections of the dynamic store, broken down into logical records of at most 5lo words for ease handling in auxiliary programs. Each data structure is accompanied by its memory-occupation table to allow relocation of all links on input.

FQXIN and FQXOUT are capable of handling several files in parallel, thus allowing merged input to be processed, or several output streams to driven.

To allow transfer of FQX files between computers of different make, the FQT routines are provided to read and write FQX files in the representation of the IBM 36o/37o computers, i.e. files as written by FQXOUT on the IBM.

PUOS measurement specification

Measuring information obtained with PUOS devices are first recorded to the drum of an on-line BESM-4 computer, then copied onto magnetic tape by means of a computer CDC-1604-A. The latest one has word length of 48 bits, therefore in the following by a "word" we will mean a group of 48 bits. The bit positions are enumerated from right to left.

The PUOS tape consists of blocks. The length of a block is 256 words at most. One stereopair measurement may consist of one or more blocks. If the first word of the first block is zero, it means that the stereopair is continued in the next block too. The measurement information starts from the second word of the block.

The information on the PUOS tape is coded in two different forms. The coordinate values of a point are written in normal binary format into some bits of a word. Another way of coding is used on the PUOS tape to write one hexadecimal digit on four bits - used for each administrative information.

PUOS data-structure specification

```
/   Pair Header                 :fix, length in pseudowords
/   Passport                    :fix, four administration words
/   Topology                    :optional, administration words

//      Oddview Header          :fix, contains the view-number
////        Track Header          :fix, contains the track-number
//////          Coordinate-pair :fix, contains the  x,y values
////        Track Trailer         :fix, two zerowords
//      Oddview Trailer         :fix, one sentinelword

//      Evenview Header         :fix, contains the view-number
////        Track Header          :fix, contains the track-number
//////          Coordinate-pair :fix, contains the  x,y  values
////        Track Trailer         :fix, two zerowords
//      Evenview Trailer        :fix, one sentinelword

/   Pair trailer                :fix, three zerowords
```

PUOS data representation

48-bits pseudowords as 6 bytes logical records, imagined as 12 four-bits hexadecimal digits.

word/bits. 45. 41. 37. 33. 29. 25. 21. 17. 13. 9. 5. 1.

imagine. 1. 2. 3. 4. 5. 6. 7. 8. 9. 1o. 11. 12.

1/ o1-o4:"F" nonzero in the last block of the stereopair

imagine. . . . . F .

2/ 13-24:"L" the length of the meas. in 48-bits words

imagine. . . . . L . L . . . .

3/ o1-12:"R" the rollnumber

13-28:"F" the framenumber

29-32:"C" the remeasurement cycle number

33-36:"O" the odd-view number

imagine. . . . O . C . F . F . F . F . R . R . R .

4/ empty:

5/ 21-28:"O" operator number

29-36:"T" measurement table number

imagine. . . . O . O . T . T . . . . .

6/ o1-o4:"V" number of measured vertices

o5-12:"T" number of measured tracks

13-2o:"A" number of measured apex tracks /primary tracks/

21-28:"H" number of measured hanging tracks

29-32:"S" serial number of this block in the stereopair - 1.

imagine. . . . . S . H . H . A . A . T . T . V .

7/ o1-o4:"L" label

o5-12:"T" type

13-2o:"N" target number

21-28:"S" number of special tracks

29-32:"F" number of blocks in the stereopair measurement

imagine. . . . . F . S . S . N . N . T . T . L .

word/bits. 45. 41. 37. 33. 29. 25. 21. 17. 13. 9. 5. 1.

imagine. 1. 2. 3. 4. 5. 6. 7. 8. 9. 1o. 11. 12.

a/ 33-36:"V" recorded viewnumber - sentinel for start of view

imagine. . . . V . . . . . .

b/ 17-2o:"3" sentinel for end of view

imagine. . . . . 3 . . . . .

c/ o1-o8:"N" recorded tracknumber - sentinel for start of track

imagine. . . . . N . N .

d/ o1-16:"X" binary x-coordinate measured value

21-36:"Y" binary y-coordinate measured value

imagine: . . . Y . Y . Y . . X . X . X . X .

e/ two empty words - sentinel for end of track

f/ three empty words - sentinel for end of stereopair

Special considerations for the topology label words

"L" =o: no beam measured

=1: beam measured

=2: special track

=3: charged secondary

=4: neutral secondary

"T" for label o and 1: beam type =1: for π^-, =2: for K^-,

=3: for $\bar{p}$, =4: for D^-.

for label 2: track type =1,2,3,4 as the beam,

=5 for "black" track,

=6 for electron or positron.

for label 3 and 4: vertex type =1: for V^o, =2: for γ,

=3: for V^I, =4: for K_s.

"N" the target number o-:-1o; 2o for gas.

"S" for label o and 1 not used;
for label 2: number of special tracks
for label 3 and 4: number of secondary tracks

"F" number of blocks for the total event; this is the only information concerning not only one stereopair but the full event.

There is one such a word for the apex /main vertex/ with label zero or one, for every type of special tracks with label two, and for every secondary interaction with label three or four.

Deviations of PUOS format

Respecifications

6.th word, bits 29-32:"S" imagined.
new meaning: event number

7.th word, bits 21-28:"S" imagined.
new meaning: experiment number

Extension

coordinate word, bits 37-44: point label/binary/ value.
especially for the measured fiducial mark's identification.

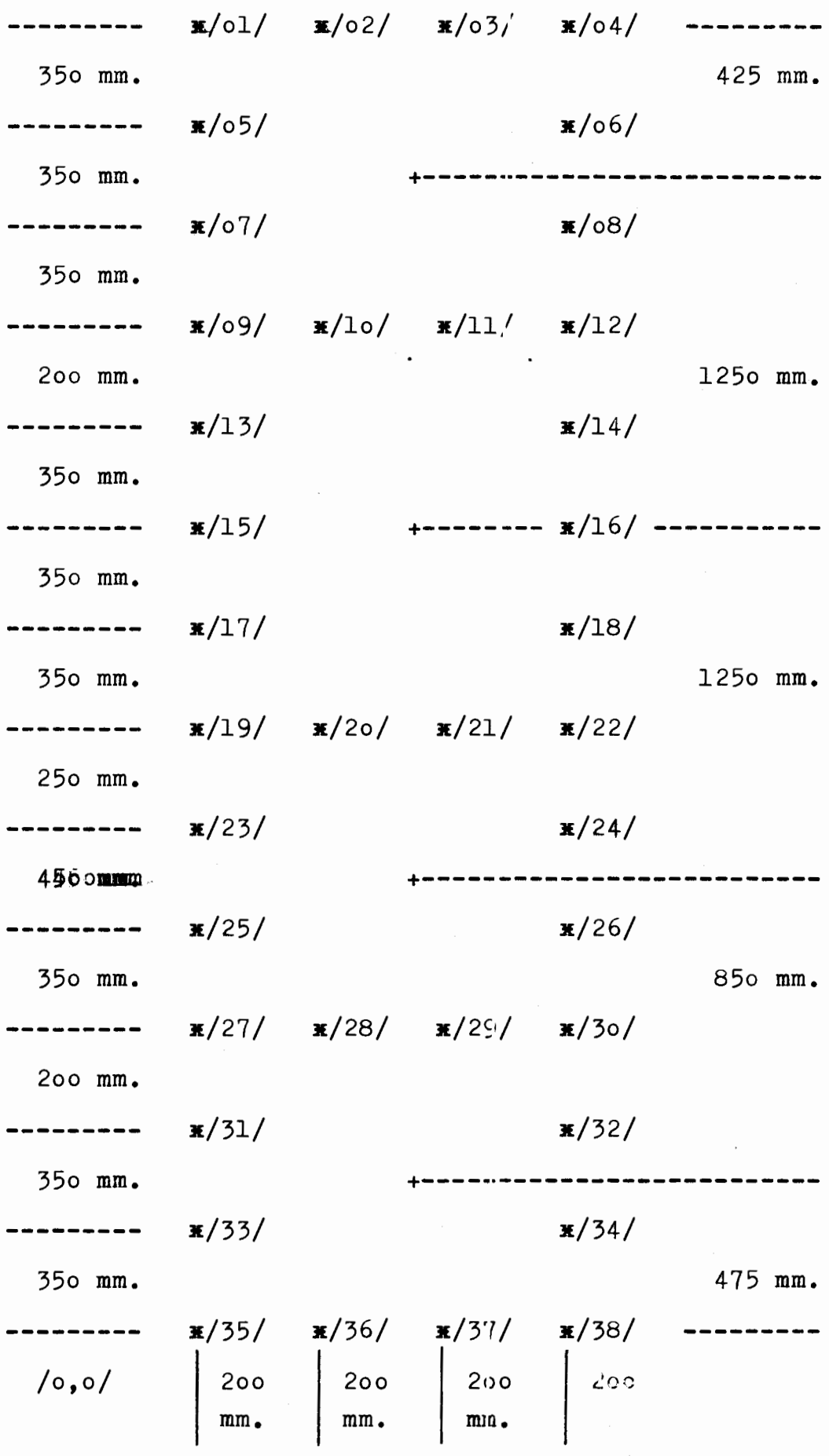
The independently sortable sequential data set of stereo-pairs: DCB= BLKSIZE=7200,LRECL=32760,RECFM=VBS

+4 bytes: the Record Descriptor Word

+6 bytes: the generated Flagword

---

offset	length	contents	
byte.bit	byte.bit	form	description
1.4	0.4	hex.	one digit oddview-number
2.0	0.4	hex.	one digit cycle-number
2.4	2.0	hex.	four digits frame-number
4.4	1.4	hex.	three digits roll-number
13.4	1.0	hex.	two digits operator-number
14.4	1.0	hex.	two digits device-number
20.0	0.4	hex.	one digit event-number
20.4	1.0	hex.	two digits number of hanging tracks
21.4	1.0	hex.	two digits number of primary tracks
22.4	1.0	hex.	two digits number of measured tracks
23.4	0.4	hex.	one digit number of measured vertices
26.4	1.0	hex.	two digit experiment-number


----
*/01/
*/02/
*/03/
*/04/
----
350 mm.
425 mm.
----
*/05/
*/06/
350 mm.
----
*/07/
*/08/
350 mm.
----
*/09/
*/10/
*/11/
*/12/
200 mm.
1250 mm.
----
*/13/
*/14/
350 mm.
----
*/15/
*/16/
350 mm.
----
*/17/
*/18/
350 mm.
1250 mm.
----
*/19/
*/20/
*/21/
*/22/
250 mm.
----
*/23/
*/24/
450 mm.
----
*/25/
*/26/
350 mm.
850 mm.
----
*/27/
*/28/
*/29/
*/30/
200 mm.
----
*/31/
*/32/
350 mm.
----
*/33/
*/34/
350 mm.
475 mm.
----
*/35/
*/36/
*/37/
*/38/

/0,0/
200 mm.
200 mm.
200 mm.
200

Experiment Specification

The "experiment number" specified for PUOS measurements, has two decimal digits in the meaning of format-specification and type-specification of the measurement, in the format "FT" as integer number, xxxxxx described below:

"F"=o: old format - two times measured fiducial mark's track.
=1: new format - only as o.th track measured fiducial marks.

"T"=o: original type - measured whole events
=1: decomposition type - the measured event will be decomposed as a serie of unique, track-by-track artifically generated subevents - especially for the "high P_t experiment".
=2: decomposition type - as above, especially for the "high multiplicity" experiment.
=3: original type - especially for the "psi experiment".
=4: original type - especially for the "vzero experiment".
=5: decomposition type - especially for the "psi experiment".
=6: decomposition type - especially for the "psi experiment".

Input and Output

//CDCPUOS DD - sequential data set of PUOS measured form stereopairs; see "puos d/s."; max. block 256 words of 48 bits - CDC16o4A machine.
DCB=(BLKSIZE=1536,RECFM=U)
2o4.8 i.e. 2o5 words of 6o bits CDC65oo machine.
DCB=(BLKSIZE=154o,RECFM=U)

//OWNPUOS DD - independently sortable sequential data set of stereopairs; complete assembled measurement's blocks in one logical record, which, of course, might be longer than one original physical block.
DCB=(BLKSIZE=72oo,RECFM=VBS,LRECL=3276o)

//IBMPUOS DD - processable common form of sequential data set of stereopairs; without any blocking-structuring, the unit of information one 48-bit word, i.e. a 6-byte pseudorecord.
DCB=(BLKSIZE=72oo,LRECL=6,RECFM=FB)

//PUOSFQX DD - sequential data set of complete events, for the geometrical reconstruction program; HYDRA-FQX format, packed, internally organized complete data structures.
DCB=(BLKSIZE=$\left\{\begin{matrix}3520\\3600\end{matrix}\right\}$,RECFM=VBS)

//GEOMFQX DD - sequential data set of geometrically reconstructed complete events, output of the geometry; HYDRA-FQX fromat, packed, internally organized complete data structures.
DCB=(BLKSIZE={3520|3600},RECFM=VBS)

//GEOMDST DD - sequential data set as Data Summary Tape, for the further processing; standard FORTRAN read /unformatted/ accessable;
DCB=(BLKSIZE={256|7200},RECFM=VBS)

The utility programs are:

== RISKDPUo, which is used to inject the fiducial mark labels into the PUOS measured stereo-pair measurements./common/.

== RISKDPU1, which is used to incorporate changes to the processable common form sequential data set of stereo-pairs.

== RISKDPU2, which is used to convert a sequential data set of PUOS measured stereo-pairs into an independently sortable sequential data set of stereo-pairs.

== RISKDPU3, which is used to convert an independently sortable sequential data set of stereo-pairs into the processable common form sequential data set of stereo-pairs.

== RISKDPU4, which is used to convert a processable common form sequential data set of stereo-pairs into the original PUOS measured form sequential data set of stereo-pairs.

== RISKDPU5, which is used to convert a sequential data set of PUOS measured form stereo-pairs into the processable common form sequential data set of stereo-pairs.

== RISKDPU6, which is used to convert a sequential data set of PUOS measured form stereo-pairs into a sorted processable common form sequential data set of stereo-pairs.

== RISKDPU7, which is used to list a processable common form sequential data set of stereo-pairs, or to produce special output of the measured fiducial marks of the stereo-pairs.

== RISKDPU8, which is used to construct a sequential data set of complete events for the geometrical reconsturction.

== RISKDPU9, which is used to construct a sequential data set of the geometrically reconstructed complete events.

RISKDPUo program

RISKDPUo is a data processing utility program used to inject the fiducial mark labels into the processable common form stereo-pair measurements .

Input and Output

RISKDPUo uses as input a sequential data set of processable common form stereo-pair measurements, uses as auxiliary input a sequential data set of fiducial mark recognition constants, produces as output a sequential data set of processable common form stereo-pair measurements, produces three independent sequential data sets as message data sets, and produces an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPUo provides a return code to indicate the results of program execution. The return codes and their meanings are:

= oooo, which indicates successful completion,

= ooo4, which indicates that a run control parameter is coded incorrectly, or used erroneously,

= ool6, which indicates an unrecoverable error,

= o24o, which indicates an unrecoverable error by the PASCAL Run Time Support System /possible FORTRAN error/.

Control

RISKDPUo is controlled by job control statements, utility control statements are not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPUo.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defined a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the PARM parameter of the EXEC statement; see the "PARM information on the EXEC statement" below.
SYSPRINT DD	defines a sequential message data set. The data set can be written to a system output device.
IBMPUOS DD	defines an input sequential data set of processable common form of stereo-pairs.
OUTPUOS DD	defines an output sequential data set of processable common form of stereo-pairs.
PØDUMP DD	defines a sequential message data set for the PASCAL Run Time Support System /diagnostics/.
FTo3Fool DD	defines a sequential message data set for the fiducial recognition debugging. The data set must written to a system output device.

Statement	Use / c o n t i n u a t i o n /
FTo6Fool DD	defines a sequential message data set. The data set must be written to a system output device.
FTloFool DD	defines an auxiliary input sequential data set of the fiducial mark recognition constants.

You can identify a private library as step library in a similar manner to that described above for a job library. A //STEPLIB DD statement applies to a job step only and overrides any //JOBLIB DD statement for the duration of the jobstep. You can also specify a step library - but not a job library - in a cataloged procedure.

Restriction

= the FTo6Fool DD statement is required for use of RISKDPUo.
= the FTloFool DD statement is required for use of RISKDPUo.
= the FTo3Fool DD statement, if needed, must be written to a system output device with DCB= BLKSIZE=121,RECFM=FA parameter.
= the IBMPUOS DD statement defined input sequential data set has a logical record length of 6 bytes, and consits of unblocked records or fix blocked RECFM=FB records with any desirable blocksize, but not longer than 72oo bytes in case of disk storage using, or 3276o bytes in case magnetic tapesusing.

PARM information on the EXEC statement

The RISKDPUo program needs one program parameter to specify the experiment number for the processing. The PARM parameter is a keyword parameter: code PARM= followed with the parameter value. For example:

```
// EXEC PGM=RISKDPo,PARM=EXPNUM1o
```

If you are using a cataloged procedure, you must include the PARM parameter in the EXEC statement that invokes the procedure and qualify the keyword with the name of the procedure step that invokes the program RISKIPUo, for example:

```
// EXEC RISKDP67,PARM.DPUo=EXPNUM12
```

The only RISKDPUo program parameter is "EXPNUMft" form, that permit you assign a value for the processing experiment number. The meaning of the "ft" fields see the "Experiment Specification" in "conventions".

Special Consideration

The actual program version must be used as:

```
//STEPLIB DD DISP=SHR,DSN=RISK.DPU.LOADLIB.{MoT192|MoT33o}
```

with

```
//FT1oFool DD DISP=SHR,DSN=RISK.DPU.{R192FIDU|R33oFIDU}
```

RISKDPU1 program

RISKDPU1 is a data processing utility program used to incorporate changes to the processable common form stereo-pair measurements, and/or generalised selection operations.

Input and Output

RISKDPU1 uses as input a sequential data set of processable common form stereo-pair measurements and a card-image sequential data set of the utility control statements, produces an output sequential data set of processable common form stereo-pair measurements, a sequential message data set, and an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPU1 provides a return code to indicate the result of program execution. The return codes and their meanings are:

= oooo, which indicates successful completion,

= ooo4, which indicates that a run control parameter is coded incorrectly, or used erroneously,

= oo16, which indicates an unrecoverable error,

= oo24, which indicates an unrecoverable error by the PASCAL Run Time Support System.

Control

RISKDPU1 program is controlled by job control statements and utility control statements.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU1.

Statement		Use
JOB		initiates the job.
JOBLIB	DD	defines a partitioned data set as job library.
EXEC		specifies the program name, of, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the PARM parameter of the EXEC statement; see the "PARM information on the EXEC statement" below.
SYSPRINT	DD	defines a sequential message data set. The data set can be written to a system output device.
SYSIN	DD	defines an input sequential data set of utility control statements; must be card image, i.e. the logical record length 8o byte, with any desirable blocklength, or unblocked, but fixed.
IBMPUOS	DD	defines an input sequential data set of processable common form stereo-pair measurements.
OUTPUOS	DD	defines an output sequential data set of processable common form stereo-pair measurements.

Statement	Use	/ c o n t i n u a t i o n /
P@DUMP	DD	defines a sequential message data set for the PASCAL Run Time Support System /diagnostics/.

You can identify a private library as step library in a similar manner to that described above for a job library. A //STEPLIB DD statement applies to a job step only and overrides any //JOBLIB DD statement for the duration of the jobstep. You can also specify a job step library - but not a job library - in a cataloged procedure.

Restriction

= the SYSIN DD statement is required for use of RISKDP1.
= the IBMPUOS DD and OUTPUOS DD statement defined input and output sequential data sets have a logical record length of 6 bytes, and consist of unblocked records or fix blocked records with any desirable blocksize, but not longer than 7200 bytes in case of disk storage using, or 32760 bytes in case magnetic tape.

PARM information on the EXEC statement

The RISKDPU1 program needs only one program parameter via tha PARM-field of the EXEC statement that invokes it. The PARM parameter is a keyword parameter: code PARM=' followed by the current date in the form mm/dd/yy where "mm" denotes the month of year, "dd" denotes the day of month, and "yy" denotes the last two digits of year; and enclosing with a single quotation mark.

For example:

```
// EXEC PGM=RISKDP1,PARM='04/29/84'
```

If you are using a cataloged procedure, you must include the PARM parameter in the EXEC statement that invokes the procedure and qualify the keyword with the name of the procedure step that invokes the program RISKDPU1, for example:

```
// EXEC RISKDP16,PARM.DPU1='04/29/84'
```

Utility Control Statements

- maintenance

:REPORT: first-recordnumber last-recordnumber

:MODIFY: first-recordnumber last-recordnumber

hexadecimal content - max. 36 bytes per cards, continuously

:VERIFY: first-recordnumber last-recordnumber

hexadecimal content - max. 36 bytes per cards, continuously

:CHANGE: first-recordnumber last-recordnumber

hexadecimal content - max. 36 bytes per cards, continuously

- selection

:OUTPUT: first-recordnumber last-recordnumber

:OUTRUN: lowest-runnumber highest-runnumber

:OUTFRM: lowest-framenumber highest-framenumber

:OUTEVT: lowest-eventnumber highest-eventnumber

:OUTVWS: lowest-viewnumber highest-viewnumber

:OUTEXP: lowest-experimentnumber highest-experimentnumber

RISKDPU2 program

RISKDPU2 is a data processing utility program used to convert a sequential data set of PUOS measured stereo-pairs into an independently sortable form sequential data set of stereo-pair measurements.

Input and Output

RISKDPU2 uses as input a sequential data set of PUOS measured stereo-pairs, produces as output an independently sortable sequential data set of stereo-pairs, a sequential data set as message data set, and produces an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPU2 provides a return code to indicate the result of program execution. The return codes and their meanings are:

= oooo, which indicates successful completion,

= ool6, which indicates an unrecoverable error,

= oo24, which indicates an unrecoverable error by the PASCAL Run Time Support System.

Control

RISKDPU2 is controlled by job control statements, utility control statements are not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU2.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defined a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM parameter of EXEC statement.
SYSPRINT DD	defines a sequential message data set. The data set can be written to a system output device.
CDCPUOS DD	defines an input sequential data set of PUOS measured stereo-pairs.
OWNPUOS DD	defines an output sequential data set of independently sortable stereo-pairs.
P@DUMP DD	defines a sequential message data set for the PASCAL Run Time Support System /diagnostics/.

You can identify a private library as step library in a similar manner to that described above for the job library. A //STEPLIB DD statement applied to a job step only and overrides any //JOBLIB DD statement for the duration of the jobstep. You can also specify a step library - but not a job library - in a cataloged procedure.

RISKDPU3 program

RISKDPU3 is a data processing utility program used to convert an independently sortable sequential data set of stereo-pairs into the processable common form sequential data set of stereo-pairs.

Input and Output

RISKDPU3 uses as input a sequential data set of independently sortable form of stereo-pairs, produces as output a sequential data set of the processable common form of the stereo-pairs, a sequential data set as message data set, and produces an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPU3 provides a return code to indicate the result of program execution. The return codes and their meanings are:
= oooo, which indicates successful completion,
= ool6, which indicates an unrecoverable error,
= oo24, which indicates an unrecoverable error by the PASCAL Run Time Support System.

Control

RISKDPU3 is controlled by job control statements, utility control statements are not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU3.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defined a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM parameter of EXEC statement.
SYSPRINT DD	defines a sequential message data set. The data set can be written to a system output device.
OWNPUOS DD	defines an input sequential data set of independently sorted form stereo-pairs.
IBMPUOS DD	defines an output sequential data set of processable common form stereo-pairs.
PASDUMP DD	defines a sequential message data set for the PASCAL Run Time Support System /diagnostics/.

You can identify a private library as step library in a similar manner to that described above for the job library. A //STEPLIB DD statement applied to a job step only and overrides any //JOBLIB DD statement for the duration of the jobstep. You can also specify a step library - but not a job library - in a cataloged procedure.

RISKDPU4 program

RISKDPU4 is a data processing utility program used to convert a processable common form sequential data set of stereo-pairs into the original PUOS-measured form sequential data set of stereo-pairs.

Input and Output

RISKDPU4 uses as input a sequential data set of processable common form of stereo-pairs, produces as output a sequential data set of PUOS-measured form of stereo-pairs, a sequential data set as message data set, and produces an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPU4 provides a return code to indicate the result of program execution. The return codes and their meanings are:

= oooo, which indicates successful completion,

= ool6, which indicates an unrecoverable error,

= oo24, which indicates an unrecoverable error by the PASCAL Run Time Support System.

Control

RISKDPU4 is controlled by job control statements, utility control statements are not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU4.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM parameter of EXEC statement.
SYSPRINT DD	defines a sequential message data set. The data set can be written to a system output device.
IBMPUOS DD	defines an input sequential data set of processable common form of stereo-pairs.
CDCPUOS DD	defines an output sequential data set of PUOS-measured form stereo-pairs.
PSDUMP DD	defines a sequential message data set for the PASCAL Run Time Support System /diagnostics/.

You can identify a private library as step library in a similar manner to that described above for the job library. A //STEPLIB DD statement applied to a job step only and overrides any //JOBLIB DD statement for the duration of the job step. You can also specify a step library - but not a job library - in a cataloged procedure.

RISKDPU5 program

RISKDPU5 is a data processing utility program used to convert a sequential data set of PUOS-measured form stereo-pairs into the processable common form sequential data set of stereo-pairs.

Input and Output

RISKDPU5 uses as input a sequential data set of PUOS-measured form of stereo-pairs, produces as output a sequential data set of processable common form stereo pairs and a sequential data set as message data set.

RISKDPU5 provides a return code to indicate the result of program execution. The return codes and their meanings are:
= oooo, which indicates successful completion,
= ooo4, which indicates an unrecoverable error.

Control

RISKDPU5 is controlled by job control statements, utility control statements not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU5.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specified the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM parameter of EXEC statement.
SYSPRINT DD	defines a sequential data set as message data set. The message data set can be written to a system output device.
CDCPUOS DD	defines an input sequential data set of PUOS-measured form stereo-pairs.
IBMPUOS DD	defines an output sequential data set of processable common form stereo-pairs.

You can identify a private library as step library in a similar manner to that described above for the job library. A //STEPLIB DD statement applied to a job step only and overrides any //JOBLIB DD statement for the duration of the jobstep. You can also specify a step library - but not a job library - in a cataloged procedure.

RISKDPU6 program

RISKDPU6 is a data processing utility program used to convert a sequential data set of PUOS-measured form stereo-pairs into a s o r t e d sequential data set of the processable common form of stereo-pairs.

Input and Output

RISKDPU6 uses as input a sequential data set of PUOS-measured form of stereo-pairs, uses some /min. three, max. seven/ auxiliary work-sequential data sets, produces as output a sorted sequential data set of processable common form stereo-pairs, and two independent message data sets.

RISKDPU6 provides a return code to indicate the result of program execution. The return codes and their meanings are:
= oooo, which indicates successful completion,
= ool6, which indicates abnormal Sort-program completion.

Control

RISKDPU6 is controlled by job control statements, utility control statements not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU6.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM parameter of EXEC statement.
SORTLIB DD	defines a partitioned data set as Sort library.
SYSOUT DD	defines a sequential data set as message data set for Sort program. The message data set can be written to a system output device.
SYSPRINT DD	defines a sequential data set as message data set. The message data set can be written to a system output device.
PUOSWKnn DD	where nn might be o1,o2,...31 - defines auxiliary storage sequential work-data sets for the Sort program. /min. 3, max. 31 different data sets/
CDCPUOS DD	defines an input sequential data set of PUOS-measured form stereo-pairs.
IBMPUOS DD	defines an output sequential data set of sorted processable common form stereo-pairs.

You can identify a private library as step library in a similar manner to that described abcve for the job library. A //STEPLIB DD statement applied to a job step only and overrides any //JOBLIB DD statement for the duration of the jobstep. You can also specify a step library - but not a job library - in a cataloged procedure.

Special Consideration

The generated sort field of PUCS data structure passport fields are to be sorted in ascending order.

experiment number	/two hex. digits/
roll number	/three hex. digits/
frame number	/four hex. digits/
event number	/one hex. digit/
oddview number	/one hex. digit/
cycle number	/one hex. digit/

Formula to limit calculate the approximate number of tracks: 1.5(number of records in the input data set)/(7ooo/recordlength).

If the data set length is known - i.e. you know how much space it occupies - you assign intermediate storage space that is at least 25% longer than the space occupied by the input.

Default assignment: 7 work areas, each of loo tracks spaces.

RISKDPU7 program

RISKDPU7 is a data processing utility program used to list a sequential data set of processable common form stereo-pairs or to produce an output sequential data set of the fiducial marks of the stereo-pairs.

Input and Output

RISKDPU7 uses as input a sequential data set of processable common form stereo-pair measurements, produces a sequential data set as message data set, or an output sequential data set of the fiducial marks of the stereo-pair measurements, and produces an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPU7 provides a return code to indicate the results of program execution. The return codes and their meanings are:

= oooo, which indicates successful completion,

= ooo4, which indicates that a run control parameter is coded incorrectly, or used erroneously,

= oo16, which indicates an unrecoverable error,

= oo24, which indicates an unrecoverable error by the PASCAL Run Time Support System.

Control

RISKDPU7 is controlled by job control statements, utility control statements are not used.

can also specify a step library - but not a job library - in a cataloged procedure.

Restriction

The PUOSFDS DD statement defined output sequential data set has a logical record length of 196 bytes with blocksize of 7o6o bytes of VBS record format. In that meaning unformatted FORTRAN readable records of the fiducial marks - for further processing. The contents of the integer words:
roll, frame, event, view, track numbers, number of fiducial marks, and for each fiducials: label, x, y; Maximum 14 measured fiducial marks per view.
For example: assume declaration of variables:

```
INTEGER*4 ROLL,FRAME,EVNT,VIEW,TKNUM,NFIDUS,MARKS 3,14
```

the input:

```
READ(lun-dsrn) ROLL,FRAME,EVNT,VIEW,TKNUM,NFIDUS,MARKS
```

or, with a direct loop for the variable-number fiducials:

```
...,((MARK(J,I),J=1,3),I=1,NFIDUS)
```

The correspondent DCB specification is:

```
...,DCB= BLKSIZE=7o6o,LRECL=196,RECFM=VBS ,...
```

PARM information on the EXEC statement

The RISKDPU7 program needs one program parameter to specify the request of the generation of fiducial mark records onto the //PUOSFDS DD statement defined output sequential data set. The PARM parameter is a keyword parameter: code PARM= followed with the parameter value. For example:

```
// EXEC PGM=RISKDPU7,PARM=FIDUCIAL
```

If you are using a cataloged procedure, you must include the PARM parameter in the EXEC statement that invokes the procedure and qualify the keyword with the name of the procedure step that invokes the program RISKDPU7. For example:

```
// EXEC RISKDP67,PARM.DP7=FIDUCIAL
```

The only RISKDPU7 program parameter is "FIDUCIAL" .

RISKDPU8 program

RISKDPU8 is a data processing utility program used to construct a sequential data set of complete events for the geometrical reconstruction data processing utility RISKDPU9 program.

Input and Output

RISKDPU8 uses as input a sequential data set of the sorted processable common form of measured stereo-pairs, uses as auxiliary input a sequential data set of the fiducial mark label's tables, produces as output a sequential data set of complete events for the geometrical reconstruction utility RISKDPU9 program, produces two indenpendent sequential data sets as message data sets, and produces an optional /suppressable/ auxiliary message data set for the PASCAL Run Time Support System's diagnostic messages.

RISKDPU8 provides a return code to indicate the results of program execution. The return codes and their meanings are:

= oooo, which indicates successful completion.

= ooo4, which indicates that a run control parameter is coded incorrectly, or used erroneously.

= oo16, which indicates an unrecoverable error.

= o24o, which indicates an unrecoverable error by the PASCAL run-time support system /possible FORTRAN run time error/.

Control

RISKDPU8 is controlled by job control statements. Utility control statements are not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU8.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the PARM parameter of the EXEC statement; see the "PARM information on the EXEC statement" below.
FIDUCIAL DD	defines a sequential data set of the fiducial mark's label tables.
IBMPUOS DD	defines a sequential data set of the sorted processable common form of measured stereo-pairs.
SYSPRINT DD	defines a sequential message data set. The data set can be written to a system output device, [illegible]
FTo6Fool DD	defines a sequential message data set. The data set must be written to a system output device.

Statement	Use / c o n t i n u a t i o n /
FTo8Fool DD	defines a sequential output data set of complete events for the geometrical reconstruction utility RISKDPU9 program.
P@DUMP DD	defines a sequential message data set for the PASCAL runtime support system./diagnostic messages/.

You can identify a private library as step library in a similar manner to that described above for a job library. A //STEPLIB DD statement applies to a job step only and overrides any //JOBLIB DD statement for the duration of the job step. You can also specify a step library - but not a job library - in a cataloged procedure.

Restriction

- The FTo6Fool DD statement is required for use of RISKDPU8.
- The FIDUCIAL DD statement is required for use of RISKDPU8, when specified a concrete fiducial table name by the PARM parameter of the EXEC statement.
- The IBMPUOS DD statement defined input sequential data set has a logical record length of 6 bytes, and consist of fixed blocked RECFM=FB or unblocked records.
- The FTo8Fool DD statement defined output sequential data set has a logical record length of 32,756 bytes, and consist of variable blocked spanned RECFM=VBS records, with any desirable blocksize, but in case transporting must be BLKSIZE=3600.

PARM information on the EXEC statement

The RISKDPU8 program offers a number of optional facilities that you can select by including the appropriate keywords in the PARM parameter of the EXEC statement that invokes it. The PARM parameter is a keyword parameter: code PARM='followed by the list of options, separating the options with commas and enclosing the list within single quotation marks /N.B. if only one appears, the quotation marks can be omitted/, for example:

```
// EXEC PGM=RISKDPU8,PARM='EXPNUM1o,FIDUCIAL'
```

The length of the option list must not exceed loo characters, including the separating commas - the program RISKDPU8 has 11 different 8-characters length option keywords. You may specify the options in any order.

If you are using a cataloged procedure, you must include the PARM parameter in the EXEC statement that invokes the procedure and qualify the keyword with the name of the procedure step that invokes the program RISKDPU8, for example:

```
// EXEC RISKDP89,PARM.DPU8='EXPNUM15,PSIAUG83,LISTNONE'
```

The RISKDPU8 program options are of two types:

= simple keywords: a positive form /e.g. LISTFIDU or COMPOUND/, that requests a facility, and one alternate negative form of any listing facility /LISTNONE/ that rejects each listing.

= keywords that permit you assign a value to a function /e.g. EXPNUM15, or MULDEC83 i.e. fiducial mark label's table-name/

This table shown the RISKDPU8 program options

Program Options		
Listing options	LISTFULL -	enables each listing options, below
	LISTFIDU-	requests fiducial mark table listing
	LISTPASS -	requests meas.passport listing
	LISTMEAS -	requests meas.data listing
	LISTONLY -	requests generated event listing
	LISTNONE -	disables each listing options, above
Control options	EXPNUMft -	specifies experiment number
	COMPOUND -	requests complete event generation
	RESTRICT -	disables secondary interactions
Fiducial Mark handling	FIDUCIAL -	automatic fiducial mark labeling
	tblename -	table directed fiducial labeling

Listing Options

LISTFULL enables each listing options. This facility should only be used in the case of a program failure, for debugging purposes. LISTFIDU requests fiducial mark label table listing. The actually used fiducial mark label table/s/ readed in by the RISKDPU8 program from the //FIDUCIAL DD statement defined sequential data set. The description of a complete fiducial mark label table see in "Fiducial Mark Label Table" paragraph. The listing of fiducial mark label tables always ends with an informative message which contains the specified table identifier "tblename", if any.

LISTPASS requests passport listing, i.e. the administration head part of the measurement. Description of the contents of the passport see "Puos Data Structure" paragraph.
LISTMEAS requests data structure listing, i.e. the information part of the measurement. Description of the data structure of the measurement see "Puos Data Structure" parapgraph.
LISTONLY requests output event generation listing, i.e. the resultate data structure which will be produced for the geometrical reconstruction. Description of the output event see "HYDRA Data Structure" paragraph.
LISTNONE disables each listing options. This facility should only be used in the case of a special program run, e.g. repetitions, when not needed a well known printout to repeate.

Control Options

EXPNUMft specifies experiment number of the measurements - for each on the actual input data set. Symbol "ft" meaning two decimal digits, first of them the format-specification, the second one the measurement type specification. The possible experiment numbers see the "Experiment Number Specification" paragraph.
COMPOUND requests complete event generation. This facility should only be used in the case of special experients, when automatically, - by the experiment number - must be track-by-track subevent generation.
RESTRICT disables secondary interaction generation, i.e. the measured secondary interaction will be generated - only administratively - as an independent primary interaction. N.B. secondary interaction has always measured vertex - primary never.

Fiducial Mark Label Table

The program RISKDPU8 requires the sequential data set defined //FIDUCIAL DD statement cnly when you use the predefined tables for the table-directed fiducial mark handling process. In the case of table-directed fiducial mark handling, the program fetches the fiducial mark label tables from the //FIDUCIAL DD statement defined sequential data set, which must be ha card image format, i.e. the DCB specification of the DD statement which LRECL=8o with any desirable block size but maximum 72oo bytes long.

The structure of a complete fiducial mark label table:

1st card specifies a symbolic identifier of that table, must be unique for each tables, and must be as the first eight position of the input card; after this symbol, in free form, two integer numbers stay, which defines the lower and upper filmnumbers applicable with that tabel; the last parameter a date as an integer number, in "mmddyy" form, to check the sequnce of remaining cards of that table /i.e. each cards must be have this date as last signal parameter of that cards/.

2nd card specifies the eight views contained number of labels; the form of the card "nv" "nf" integer pair sequence, where "nv" means the viewnumber, "nf" means the number of fiducial marks on the view; the last integer /the 17th number/ on this card the date, in the same form and value as it was on the first card.

Fiducial Mark Label Table /con't/

3rd-10th cards specify the possible eight views contained labels; the sequence is strictly view-1 to view-8, on each cards the possible maximum 14 fiducial mark labels in free form as integer numbers; the unmeasured or restricted labels symbolised as explicit zeroes, the last integer /the 15th number/ the date, in the same form and value as it was on the first card.

The sequential card image input data set defined //FIDUCIAL DD statement can contain one or more complete fiducial mark label tables.

This table shows the fiducial mark label tables which are applicable on program RISKDPU8 /up to may 1984/.

table name	lower film	upper film	1	2	3	4	5	6	7	8
			number of fiducial marks							
PSIMAY83	4oo	415	8	8	8	8	1o	1o	8	8
PSIAUG83	431	451	8	8	8	8	9	9	7	7
PSINOV83	431	451	8	8	8	8	9	9	7	7
HPTAPR83	33o	368	6	6	6	6	6	6	o	o
HPTMAY83	33o	368	8	8	8	8	8	8	o	o
HPTOCT83	33o	368	11	11	11	11	12	12	o	o
MULDEC83	179	2o5	8	8	8	8	8	8	8	8
PUOSAELT	33o	368	8	8	8	8	o	o	o	o

RISKDPU9 program

RISKDPU9 is a data processing utility program used to construct a sequential data set of the geometrically reconstructed complete events.

Input and Output

RISKDPU9 uses as input a sequential data set of the prepared complete events for geometrical reconstruction, uses as auxiliary input a sequential data set of th optical title bank constants, produces as output a sequential data set of the geometrically reconstructed complete events, and produces three independent sequential data sets as message data sets.

RISKDPU9 provides a return code to indicate the results of program execution. The return codes and their meanings are:
= oooo, which indicates successful completion.
= o24o, which indicates an unrecoverable error.

Control

RISKDPU9 is controlled by job control statements and utility control statement for the tape handling initialisation.

Job Control Statements

This table shows the job control statements necessary for using RISKDPU9.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM field of the EXEC statement.
FTo2Fool DD	defines a message data set, which must be written to a system output device /debugging/.
FTo5Fool DD	defines an input sequential data set of the actual optical title bank constants.
FTo6Fool DD	defines a message data set, which must be written to a system output device /printed output/.
FTo7Fool DD	DUMMY - not yet implemented facility /INDEX-cards/
FTo9Fool DD	defines an input sequential data set of the prepared complete events for the geometrical reconstruction.
FTloFool DD	defines an output sequential data set of the geometrically reconstructed complete events.
FTllFool DD	DUMMY - not yet implemented facility /save-file/.

Utility Control Statement

The second concatenated card image sequential data set of the FTo5F001 DD statement can contain the tape handling initialization parameters for the "XQIN" title bank.

The form and contents:

```
//   DD *  - as concatenation for the //FTo5Fool DD statement.
*_____XQIN_____8(FORMAT-statement specification)
... the 8 parameter-values
*FINISH
```

The default assigment resides in the optical titles resident library RISK.GEOMETRY.TEXTLIB.TITLES partitioned data set as member GEOMTITL.

The optical title banks - in this library - as members are:

group number	lower film	upper film	member name
165	lol	175	RUN165
192	179	2o5	RUN192
272	248	329	RUN272
33o	33o	368	RUN33o
37o	37o	399	RUN37o
412	4oo	415	RUN412
433	431	451	RUN433

The "XQIN" Tape Handling Initialisation Title Bank

Used by XQINIT routine from patch TAPEHB.

Number of Links=o

Number of Structural Links=o

Number of Data Words=8

The data word contents

1	NSKI	number of events skipped on input
2	NEVI	last event skipped on input
3	NSKO	number of events skipped on output
4	NEVO	last event skipped on output
5	NSKS	number of events skipped on save
6	NEVS	last event skipped on save
7	NEVP	total number of events to be processed after skip
8	NLGIN	input unit - data set reference number, if nonzero

Considerations

NEVI,NEVO,NEVS .eq. o means skip without check

NEVI,NEVO,NEVS .ne. o skip and check number of last skipped event

If "eof" reached on output /or save/ while skipping, the tape is positioned before the "eof" without check.

NEVO,NEVS .eq. o forces NEVO,NEVS .eq. NEVI

NSKO,NSKS .lt. o means no output, no save.

NEVP .eq. o there is no restriction on the number of events to be processed.

Restrictions

= FTo6Fool message data set contains the FORTRAN Run Time Support System error messages - if suppressed, you have not any information about the trouble/s/.

= FTo9Fool and FTloFool sequential data sets must be in the HYDRA system FQT format sequential data sets, i.e. blocksize 3600 /or 3520 if disk storage used/, logical record length 3276o bytes, and record format VBS.

RISKDPUa program

RISKDPUa program is a data processing utility program used to construct a sequential data set of Data Summary Tape form complete geometrically reconstructed events.

Input and Output

RISKDPUa uses as input a sequential data set of HYDRA-FQT form complete geometrically reconstructed events, uses as base input a sequential data set of Data Summary Tape form complete geometrically reconstructed events, uses as an auxiliary input a sequential data set of magnetic field constants; produces as output a sequential data set of /updated/ Data Summary Tape, and a sequential data set as message data set.

RISKDPUa provides a return code to indicate the results of the program execution. The return codes and their meanings are:
= oooo, which indicates successful completion,
= o24o, which indicates an unrecoverable error.

Control

RISKDPUa is controlled by job control statements and utility control statements.

Job Control Statements

This table shows the job control statements necessary for using RISKDPUa.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM field of the EXEC statement.
FTo1Fool DD	defines a sequential data set of the auxiliary input constants of the magnetic field data.
FTo5Fool DD	defines a sequential data set of the input utility control statements.
FTo6Fool DD	defines a message data set, which must be written to a system output device /printed output/.
FTloFool DD	defines an input sequential data set of HYDRA-FQT form complete geometrically reconstructed events.
FTllFool DD	defines an input sequential data set of DST/SLICE/ form complete geometrically reconstructed events.
FTl2Fool DD	defines an output sequential data set of DST/SLICE/ form complete geometrically reconstructed events.

Utility Control Statements

FORTRAN format-controlled read statement readed control cards have the FORMAT(A1o,I1o) specification. Exception: the vertex coordinate-declaration cards, which readed by FORMAT(3F1o.o).

This table shows the utility control statements necessary for using RISKDPUa.

Statement		Use
FULLOG	{1|2|3}	logging-level declaration /standard = 2/
VERTEX	n	number of successive vertex-declarations
... n-times coordinate-declarations		
MAXIM	n	number of processable input events
DEBUG	n	number of debugged processable input events
SKIP	n	number of skipped input events on FT1oFool DD.
CREATE		"don't use old DST" command
NEWDST		"produce new DST" command
LUNREAD	n	overwrite dsrn. of //FTo5Fool DD statement
LUNPRINT	n	overwrite dsrn. of //FTo6Fool DD statement
LUNFQT	n	overwrite dsrn. of //FT1oFool DD statement
LUNOLD	n	overwrite dsrn. of //FT11Fool DD statement
LUNNEW	n	overwrite dsrn. of //FT12Fool DD statement

N.B. - the first four characters of the keywords are significant.

RISKDPUb program

RISKDPUb is a data processing auxiliary program used to list a sequential data set of HYDRA-FQT form data structure's headervector elements, or, by program-modification, special debugging list production.

Input and Output

RISKDPUb uses as input a sequential data set of HYDRA-FQT form complete event's data structures, produces three independent message data sets as output.

RISKDPUb provides a return code to indicate the results of program execution. The return codes and their meanings are:
= oooo, which indicates successful completion,
= o24o, which indicates an unrecoverable error.

Control

RISKDPUb is controlled by job control statements, utility control statements are not used.

Job Control Statements

This table shows the job control statements necessary for using RISKDPUb.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM field of the EXEC statement.
FTo6Fool DD	specifies an output message data set for the FORTRAN Run Time Support System diagnostic messages, and, if by any program-modification is requested, the resultate listing of the HYDRA-fashioned debugging. /see Hydra-Manual, DQSNAP-description/.
FTo7Fool DD	specifies an output message data set for the last element's header vector listing. The data set must be written to a system output device, with explicit DCB specification.
FTo8Fool DD	specifies an output message data set for the each element's header vector listing. The data set must be written to a system output device, with explicit DCB specification.
FTo9Fool DD	xpxxixixxx defines an input sequential data set of HYDRA-FQT form data structures of complete events. Might be RISKDPU9/HYGEOM/ input or output data set.

RISKDPUc program

RISKDPUc is a data processing auxiliary program used to draw a prepared complete event for geometrical reconstruction to the line printer.

Input and Output

RISKDPUc uses as input a sequential data set of HYDRA-FQT form prepared complete events for geometrical reconstruction, a sequential data set of control statements, and produces an output message data set of the draw of the specified event and another message data set for the FORTRAN Run Time Support System diagnostics and/or HYDRA-system logging messages.

RISKDPUc provides a return code to indicate the results of program execution. The return codes and their meanings are:
= oooo, which indicates a successful completion,
= o24o, which indicates an unrecoverable error.

Control

RISKDPUc is controlled by job control statements and utility control statements.

Job Control Statements

This table shows the job control statements necessary for using RISKDPUc.

Statement	Use
JOB	initiates the job.
JOBLIB DD	defines a partitioned data set as job library.
EXEC	specifies the program name, or, if the job control statements reside in a procedure library, the procedure name. Additional information not needed to specify in the PARM field of EXEC statement.
FTo5Fool DD	defines an input sequential data set of utility control statements.The data set must be card image.
FTo6Fool DD	defines an output message data set for the FORTRAN Run Time Support System diagnostics. The data set must be written to a system output device.
FTo7Fool DD	defines an output message data set of drawing. The data set must be written to a system output device with explicit DCB specification.
FTo8Fool DD	defines an input sequential data set of prepared cpmplete events for geometrical reconstruction in HYDRA-FQT form.

You can identify a private library as step library in a similar manner to that described above for a job library. A //STEPLIB DD statement applies to a job step only and overrides

any //JOBLIB DD statement for the duration of the jobstep. You can also specify a step library - but not a job library - in a cataloged procedure.

Utility Control Statements

The FORTRAN format-controlled read statement readed control cards have the format specification (I1o,1oA4). The first value means the sequence number of the specified event in the input sequential data set, the next 4o character as header-text will be applicated on the drawing.

The RISK DP program project

program number	input DD	process	output DD
o	IBMPUOS	fiducial label insertion	IBMPUOS
1	IBMPUOS	direct data manipulation	IBMPUOS
2	CDCPUOS	data structure recreation	OWNPUOS
3	OWNPUOS	data structure recreation	IBMPUOS
4	IBMPUOS	data structure recreation	CDCPUOS
5	CDCPUOS	data structure recreation	IBMPUOS
6	CDCPUOS	d/s recreation and sorting	IBMPUOS
7	IBMPUOS	data structure listing	SYSPRINT
		fiducial mark extraction	PUOSFDS
8	IBMPUOS	data structure recreation	PUOSFQX
9	PUOSFQX	geomatrical reconstruction	GEOMFQX
A	GEOMFQX	geomatrical reconstruction	GEOMDST
B	PUOSFQX GEOMFQX	auxiliary - up to user	SYSPRINT
C	PUOSFQX	auxiliary /drawing/	SYSPRINT

The Cataloged JCL Procedures

A cataloged JCL procedure is a set of Job Control Language statements stored in a system library, the procedure library. It comprises one or more EXEC statements, each of which may be followed by one or more DD statements.

You can retrieve the statements by naming the cataloged procedure in the PROC parameter of an EXEC statement in the input job stream. When the Job Scheduler encounters such an EXEC statement, it replaces it in the input stream with the statements of the cataloged procedure.

The use of cataloged procedures saves time and obviates errors in coding frequently used sets of job control statements. Even if the statements in a cataloged procedure do not match your requirements exactly, you can easily modify them or add new statements for the duration of a job.

The RISK DP Cataloged Procedures

#oRISK - designed to use the program RISKDPUo /IBMPUOSo/

#1RISK - designed to use the program RISKDPU1 /IBMPUOS1/

#2RISK - designed to use the program RISKDPU2 /IBMPUOS2/

#3RISK - designed to use the program RISKDPU3 /IBMPUOS3/

#4RISK - designed to use the program RISKDPU4 /IBMPUOS4/

#5RISK - designed to use the program RISKDPU5 /IBMPUOS5/

#6RISK - designed to use the program RISKDPU6 /IBMPUOS6/

#7RISK - designed to use the program RISKDPU7 /IBMPUOS7/

#8RISK - designed to use the program RISKDPU8 /IBMPUOS8/

#9RISK - designed to use the program RISKDPU9 /GEOMETRY/

#ARISK - designed to use the DST generator /SHAFARIK/

#BRISK - designed to use the FQX auxiliary /FQREADER/

#CRISK - designed to use the FQX auxiliary /FQDRAWER/

Appendix

When you create or retrieve a data set, the system requires certain information. This information is supplied on the DD statement the defines the data set.

This appendix can be used as a checklist: as you code your DD statements, find the function you are performing in the left-hand column of the table. Across from the function are two separate lists of parameters. These parameters describe the information that you must supply to the system and the information that you may have to supply. You can compare your DD statement with what is listed to make sure all the required information is available to the system.

Following the table are examples of the DD statements that might be used when performing functions described in the table. Each example is keyed by number to a particular block within the table.

FUNCTION:	INFORMATION: that is			
Creating a Data Set	Always Required		May Be Required	
Temporary Data Sets				
Creating a Data Set on a Unit Record Device	UNIT	1	DCB UCS	2
Creating a Data Set on a Tape Volume	UNIT	3	DCB VOLUME LABEL	4
Creating a Data Set in the Output Stream	SYSOUT	7	DCB UNIT SPACE	8
Creating a Data Set on a Direct Access Volume	UNIT SPACE	9	DCB VOLUME LABEL	1o
Nontemporary Data Sets				
Creating a Data Set on a Tape Volume	UNIT DSNAME DISP	11	LABEL DCB VOLUME	12
Creation a Generation Data Set on a Tape Volume	UNIT DSNAME DISP	13	DCB LABEL VOLUME	14
Creating a Sequential Data Set on a Direct Access Volume /BSAM or QSAM/	UNIT DSNAME DISP SPACE	15	LABEL DCB VOLUME	16

Creating a Data Set /con't/	Always Required		May Be Required	
Creating a Data Set With Direct Organization on a Direct Access Volume /BDAM/	UNIT DSNAME DISP SPACE	17	LABEL VOLUME	18
Creating a Partitioned Data Set on a Direct Access Volume /BPAM/	UNIT DSNAME DISP SPACE	19	LABEL VOLUME DCB	2o
Creating a New Member for a Partitioned Data Set	DISP DSNAME	21	UNIT VOLUME	22
Creating a Data Set With Indexed Sequential Organization o on a Direct Access Volume /QISAM/	UNIT DSNAME DISP DCB SPACE	23	VOLUME LABEL	24
Creation a Generation Data Set on a Direct Access Volume	SPACE DISP UNIT DSNAME	25	DCB LABEL VOLUME	26
Retriveing a Data Set				
Retrieving a Cataloged Data Set	DSNAME DISP	27	DCB LABEL UNIT	28

Retrieving a Data Set /con't/	Always Required		May Be Required	
Retrieving a Noncataloged Data Set on a Tape Volume	DSNAME UNIT VOLUME DISP	29	LABEL DCB	30
Retrieving a Noncataloged Sequential Data Set on a Direct Access Volume /BSAM or QSAM/	UNIT VOLUME DSNAME DISP	31	LABEL	32
Retrieving a Noncataloged Data Set with Direct Organization on a Direct Access Volume	UNIT VOLUME DSNAME DISP	33	LABEL	34
Retrieving a Member of a Partitioned Data Set	DISP DSNAME	35	UNIT VOLUME	36
Retrieving a Data Set with Indexed Sequential Organization on a Direct Access Volume /QISAM or BISAM/	DSNAME UNIT VOLUME DCB DISP	37		
Retriveng a Passed Data Set	DSNAME DISP	38	LABEL DCB UNIT VOLUME	39

Examples

```
 1   //DDo1 DD UNIT=14o3
 2   //DDo2 DD UNIT=14o3,UCS=PCAN,DCB=PRTSP=2
 3   //DDo3 DD UNIT=24oo
 4   //DDo4 DD UNIT=24oo-1,DCB=DEN=1,VOL=SER=141312,LABEL=2
 5   //DDo5 DD SYSOUT=L
 6   //DDo6 DD SYSOUT=G,DCB=PRTSP=2
 7   //DDo7 DD SYSOUT=(M,,7956)
 8   //DDo8 DD SYSOUT=B,UNIT=2314,SPACE=(8o,3oo),
     //         DCB=BLKSIZE=64o
 9   //DDo9 DD UNIT=SYSDA,SPACE=(TRK,(2o,5))
1o   //DDlo DD UNIT=2314,SPACE=(CYL,(2,1)),LABEL=(,SUL),
     //         VOLUME=SER=131412,DCB=(RECFM=S,LRECL=X)
11   //DD11 DD UNIT=24oo,DSN=OUT,DISP=(NEW,KEEP)
12   //DD12 DD UNIT=24oo-2,DSN=WLK18,DISP=(,KEEP),LABEL=(,NL),
               DCB=TRTCH=C,VOL=SER=L21413
13   //DD13 DD DISP=(,CATLG),UNIT=24oo,DSN=R43o.PSI(+1)
14   //DD14 DD DISP=(,CATLG),UNIT=24oo,DSN=R33o.HPT(+1),
     //         LABEL=(,SUL),DCB=A.B.C,VOL=SER=XXXXXX
15   //DD15 DD UNIT=2314,DSN=LNG,DISP=(,KEEP),SPACE=(TRK,(12,2))
16   //DD16 DD UNIT=2314,DSN=CLB,DISP=(,CATLG),
     //         SPACE=(1o24,(1oo,2o)),LABEL=(,SUL,,EXPDT=841o1),
     //         VOL=SER=413121,DCB=(BLKSIZE=1o24,RECFM=F)
17   //DD17 DD UNIT=2314,DSN=CJB,DISP=(NEW,KEEP),
     //         SPACE=(CYL,(9,1)),DCB=DSORG=DA
18   //DD18 DD UNIT=2314,DSN=QAY,DISP=(,PASS),
     //         SPACE=(1o24,(2oo,2o)),DCB=(DSORG=DA,BLKSIZE=1o2o,
     //         KEYLEN=4,RECFM=F),LABEL=(,SUL),VOL=SER=%L2141
```

Examples /con't/

```
19   //DD19 DD UNIT=2314,DSN=PDS99,DISP=(NEW,KEEP),
     //         SUBALLOC=(CYL,(2o,1,3),STEP1.DDo)
2o   //DD2o DD UNIT=2314,DSN=AHTRY,DISP=(,CATLG),
     //         SPACE=(CYL,(8,2,2)),LABEL=(,,PASSWORD),
     //         VOL=SER=CCCCCC,DCB=(RECFM=F,LRECL=8o)
21   //DD21 DD DSNAME=AHTRY(SET4),DISP=OLD
22   //DD22 DD UNIT=2314,VOL=SER=AAAAAA,DISP=OLD,
     //         DSNAME=SHTR(MEMB2)
23   //DD23 DD UNIT=2314,DSN=DAT(PRIME),DISP=(NEW,KEEP),
     //         DCB=DSORG=IS,SPACE=(TRK,(1o,,1))
24   //DD24 DD UNIT=2314,DSN=ISQ(PRIME),DISP=(,KEEP),
     //         DCB=(DSORG=IS,BLKSIZE=24o,CYLOFL=1,OPTCD=MYLR,
     //         RECFM=FB,LRECL=6o,RKP=18,KEYLEN=1o),SPACE=(CYL,2),
     //         VOL=SER=CCCCCC,LABEL=EXPDT=84365
     //      DD UNIT=2314,DSN=ISQ(OVLOW),DISP=(,KEEP),DCB=*.DD24,
     //         SPACE=(CYL,1),VOL=SER=DDDDDD,LABEL=EXPDT=84365
25   //DD25 DD DSN=PAY.WEEK(+1),DISP=(,CATLG),UNIT=2314,
     //         SPACE=(TRK,(3,2))
26   //DD26 DD DSN=INV.FORM8(+2),DISP=(,CATLG),UNIT=2314,
     //         VOL=SER=BBBBBB,LABEL=(,SUL),SPACE=(CYL,(2,1)),
     //         DCB=(MODEL2,RECFM=F,LRECL=8o)
27   //DD27 DD DSN=A.B.C,DISP=OLD
28   //DD28 DD DSN=KELL23,DISP=OLD,LABEL=(,NSL),UNIT=(,P),
     //         DCB=(BUFNO=4,HIARCHY=1)
29   //DD29 DD DSN=FILE9,UNIT=24oc,VOL=SER=27ooo1,DISP=OLD
```

Examples /con't/

```
30   //DD30 DD DSN=MILS,UNIT=2400-2,DISP=(OLD,PASS),VOL=SER=21,
     //         LABEL=(,NSL),DCB=(BLKSIZE=1600,LRECL=80)
31   //DD31 DD DSNAME=GLOSS,DISP=OLD,UNIT=2314,VOL=SER=EEEEEE
32   //DD32 DD DSN=LAB84,UNIT=2314,VOL=SER=CCCCCC,LABEL=(,,,IN)
33   //DD33 DD DSN=SERNOS,DISP=OLD,UNIT=2314,VOL=SER=BBBBBB
34   //DD34 DD DSN=BOLS,DISP=OLD,VOL=SER=EEEEEE,
     //         UNIT=2314,LABEL=(,SUL)
35   //DD35 DD DSN=PGM(A82),DISP=SHR
36   //DD36 DD DSN=LIBS(PROJ34),UNIT=2314,VOL=SER=CCCCCC,DISP=OLD
37   //DD37 DD DSN=IND32,UNIT=(2314,2),DISP=OLD,
     //         DCB=DSORG=IS,VOL=SER=(CCCCCC,DDDDDD)
38   //DD38 DD DSN=CHAN,DISP=(OLD,KEEP)
39   //DD39 DD DSN=*.STEP2.CREATE,DISP=(OLD,DELETE),LABEL=(,NL),
     //         UNIT=(,2),VOL=(PRIVATE,,4),DCB=*.STEP2.CREATE
```