GEM TN-93-293

TOR
Off-line Event Reconstruction for TTR
(User Guide)

P. Murat
Institute for Theoretical and Experimental Physics-ITEP

February 17 1993

Abstract:

This note contains detailed description of TOR - the TTR Off-line Reconstruction code. TOR provides capabilities to process the TTR data, reconstruct tracks of cosmic particles detected by TTR and to study the resolution of high precision drift and cathode strip chambers. Program was intensively used during the TTR tests of GEM muon chambers and more that 10^6 events were reconstructed with TOR to the moment. TOR is written in FORTRAN 77 and FORTRAN 77 extensions were widely used in coding. The program runs on HP Cluster of SSCL PDSF computing facility under UNIX operational system.

Contents

T i
1 Getting started with TOR 1
1.1 Data analysis with TOR 1
1.2 I/O files used by TOR 2
1.3 tor.init file format 3
1.4 TOR runfile format 3
1.5 TOR event file format 5
2 Data types used by TOR 8
2.1 Constants and data types of general use 8
2.2 Constants and data types describing TTR event 10
2.3 CAMAC description in TOR 11
2.4 Constants and data types used by TOR fitting procedures 13
2.5 Constants and data types used for histogramming 14
2.6 Constants and data types for TTR scintillation counters 15
2.7 Constants and data types used for TTR Iarocci tubes description 18
2.8 Constants and types used for drift chambers description 19
2.9 Constants and data types used for CSC description 23
3 TOR global variables 27
3.1 Common-block /TRACKS/ 27
3.2 Common-block /TTR_EVENT/ 27
3.3 Common-block /GLOBAL_VARS/ 29
3.4 Common-block /TTRIO/ 30
3.5 Common-block /TTR_SCINT/ 31
3.6 Common-block /TTR_IAROCCI/ 31
3.7 Common-block /PDTR_VARS/ 32
3.8 Common-block /CSCR_VARS/ 32
3.9 Common-block /PDTU_VARS/ 33
3.10 Common-block /TZERO/ 34
3.11 Common-block /WS/ 34
3.12 Common-block /TTR_BITS/ 34
3.13 Common-block /PAWC/ 35
3.14 Common-block /HREQUEST/ 35
4 Event parameters calculated by TOR 36
5 TTR scintillation counters 37
5.1 TTR scintillation counters data format 37
5.2 TTR scintillation counters constant file format 37
5.3 Examples 39
5.4 Histogramming for scintillation counters 40
5.4.1 Reserved histogram ID's 40
5.4.2 Histogram sets booked for scintillation counters 40
5.4.3 Histogramming parameters calculated for TTR scintillation counters 41
6 TTR Iarocci tubes 42
6.1 TTR Iarocci tubes data format 42
6.2 The format of TTR Iarocci constant file 42
6.3 Examples 43
6.4 Histogramming for Iarocci's 43
6.4.1 Reserved histogram ID's 43
6.4.2 Histogram sets booked for Iarocci tubes 43
6.4.3 Parameters calculated for TTR Iarocci tubes 43

7 TOR processing of pressurised drift tubes 44
7.1 PDTR data format 44
7.2 PDTR constant file format 44
7.3 Histogramming for PDTR 46
7.3.1 Histogram ID's reserved for PDTR 46
7.3.2 Histogram sets booked for PDTR 47
7.3.3 Histogramming parameters calculated for PDTR 47
8 Processing of Michigan pressurized drift tube chamber 48

have the same limits and the same binning. This allows to combine all the histograms of the same type into one **histogram set** and to operate with **histogram set** as a unit of histogramming - for example, - to request filling of the whole **histogram set.**

3. At the final stage of data analysis PAW [2] package, developed in CERN, is supposed to be used. For each technology processed TOR calculates some number of event parameters and provides tools to put these parameters into PAW ntuples. Each of these parameters has symbolic name read by TOR from runfile. The following symbolic names are used.

'c001' to 'c999' : parameters of Dubna CSC

'e001' to 'e999' : general parameters of event

'i001' to 'i999' : parameters of TTR Iarocci's

'p001' to 'p999' : parameters of Dubna PDT chambers

's001' to 's999' : parameters of TTR scintillators

'u001' to 'u999' : parameters of Michigan PDT chamber

These histogramming parameters will be discussed in more detail in the chapters, devoted to the description of different technologies. Moreover, TOR allows user to define additional event parameters to be put into ntuples and to create up to 10 ntuples simultaneously.

1.2 I/O files used by TOR

There are several files used by TOR.

tor.init file : when started TOR looks in the current directory for a text **tor.init** file and reads from it initial data.

runfile : TOR reads steering data cards and some corrections to the program constants from **runfile.** TOR runfile name is defined in **tor.init** file. TOR **runfile** is written in ASCII format as well.

event file : TOR reads experimental data to be processed from **event file.** TTR data acqusition system stores the data corresponding to each experimental run to separate **event files.** Only one experimental run can be processed during each TOR job. TOR **event file** is written in binary format described in sect.1.5.

constant files : TOR reads inital data describing TTR geometry and chambers being tested from so-called **constant files.** Constant files for all technologies are written in text (ASCII) format.

rt-files : the data on space-time functions for different drift chambers is read from so-called

1.3 tor.init file format

When started TOR looks in the current directory for the text file **tor.init** and reads initial data from it. This file itself defines only the names of constant files used for initialization of TTR geometry and name of so-called runfile which contains steering cards, defining data file with run to be processed, TTR subsystems and tested chambers to be processed, histograms and ntuples to be filled and so on. In **tor.init** file only one user keyword is recognised :

*FILE technology filename 0

Parameters:

technology : FFREAD text constant (enclosed in apostrophes) : technology type. Possible values :

'scin'	: TTR scintillation counters
'iaro'	: TTR Iarocci tubes
'pdtr'	: Dubna drift chambers
'pdtu'	: Michigan drift chamber
'cscr'	: Dubna strip chamber
'run'	: runfile

filename : FFREAD text constant - name of corresponding data file

0 : end of data card marker

Here is the example of **tor.init** file:

```
list
c                 read scintillators data
*file 'scin'   '/usr/local/users/murat/constants/TTR.scint.dat.txt'    0
c                              read iarocci data
*file 'iaro'   '/usr/local/users/murat/constants/TTR.iarocci.dat.txt' 0
c                              read PDTR data
*file 'pdtr'   '/usr/local/users/murat/constants/TTR.pdtr.dat.txt'    0
c                              read PDTU data
*file 'pdtu'   '/usr/local/users/murat/constants/TTR.pdtu.dat.txt'    0
c                              read CSCR data
*file 'cscr'   '/usr/local/users/murat/constants/TTR.cscr.dat.txt'    0
c                              read run steering cards
*file 'run'    '/usr/local/users/murat/ttr/run1127.init'              0
end
```

1.4 TOR runfile format

TOR runfile is text file defining TTR event file to be processed by TOR, technologies to be processed, histograms and ntuples to be filled and so on. At the same time TOR runfile contains corrections to some program constants which are to be changed for procession of given event file. The conventional names for TOR runfiles are **runxxxx.init**, where xxxx stands for TTR run number. For example, runfile for TTR run 1127 has name **run1127.init**.

The following data cards and data blocks are recognized here.

1. **event file name** is defined by card

   *INPUT filename

   where **filename** is event file name (text constant, enclosed in apostrophes)

2. By default TOR processes all the events in event file. Nevertheless, a possibility exists to choose range of events to be processed in given event file. The range may be set by data card

   MINEVENT minevent maxevent

   where **minevent** and **maxevent** are minimal and maximal event numbers to be processed by TOR.

3. Data card

   DEBUG minevent maxevent

   requires debug output to be printed for events in the range from **minevent** to **maxevent.**

4. Different levels of debug output are switched ON by data block defined by keyword *BIT. The format of this data block is the following.

   *BIT bit1 bit2 ... bitn
   'END'

   where integer constants **bit1, ..., bitn** define levels of debug output requested, 'END' is end of input marker.

5. Technologies to be processed are defined by data block

   *PROCESS tech1 tech2 ... techn
   'END'

   where **tech1 ... techn** - names of technologies. The following technology names are recognized:
   'scin', 'iaro', 'pdtr', 'pdtu', 'cscr'. Any subset of technologies can be chosen for processing.

6. Histogram filling is requested by similar data block, defined by *HISTOGRAM keyword:

   *HISTOGRAM tech1 .... techn
   'END'

   where **tech1 ... techn** - names of technologies for which histograms are to be filled. The histogramming parameters for given technology are calculated regardless whether or not histogram filling was requested for this technology.

7. All the histograms of the histogram set are assumed to have the same limits and number channels. To the moment histogram sets conntain only 1-dimensional histograms. Histogram set is requested by data card

*HSET tech name nchannels xmin xmax vmx

where

tech : technology name

name : histogram set name. These are described in the sections corresponding to the technologies

nchannels (integer) : number of channels in the historam

xmin, xmax, vmx (real) : histogram limits and maximal channel contents (see HBOOK [5] description for details).

8. Ntuple is requested by data block defined by keyword *NTUPLE. The data block format is the following.

```
*NTUPLE ID p1name p2name ...
          pkname ...
          pmname ...
          0
```

where

ID : HBOOK ID of ntuple

p1name... : character constants enclosed in apostrophes - symbolic names of parameters to be put into ntuple.

0 : end of parameter list marker.

9. Corrections to the constants for any given technology should start with data card

   TECHNOLOGY tech_code

   where **tech_code** - integer constant, defining technology code

10. For drift technologies the data cards (data blocks) corresponding to following keywords are recognized in **runfile**: RT, WDY, WYSL, TZERO, DTZERO. The format of corresponding input data is described in section 8.2.

11. The following CSC keywords are recognized in runfile: PEDESTALS, PWIDTHS, SLOPES. The format of corresponding data is described in section 10.2.

1.5 TOR event file format

TOR reads input event data from files created by TTR data acqusition system. Each event read by TOR consists of event header, one or several data groups and end-of-event marker. Along with normal data events TTR files contain some additional information which is written there in the form of events. Such special events differ from normal data events by special flag set in the event header.

TTR event has the following structure:

```
STRUCTURE/TTR_EVENT_/
  integer*2          size
  integer*2          daq_mask
  integer*2          trigger_mask
  integer*2          run_number
  integer*4          number
  integer*4          date_time
  record/GROUP_/     group
END STRUCTURE
```

size : event size in 2-byte words, including header size and end-of-event marker

daq_mask : provides information about type of event, data acquisition system used, zero suppression and so on. This information is encoded in the following way:

```
bits 0-3 : 0 : normal data event
           1 : comment event
           2 : lower threshold event
           3 : upper threshold event
           4 : special event
           5 : initial comment event
           6 : voltage comment event
           7 : calibration comment event
           6 : Monte Carlo event
           9-15 : reserved for future use
bit 4 :    0 : software zero suppression OFF
           1 : software zero suppression ON
bit 5 :    0 : Macintosh DAQ system
           1 : UNIX DAQ system
bits 6-15 : reserved for future use
```

trigger_mask :

run_number : TTR run number (1..N)

number event number within the run (1..N)

date_time : date and time (4-byte character constant)

group : placeholder for the first data group. Each group contains data from one or several CAMAC modules of the same type placed in one CAMAC crate. Group starts from group header followed by the data itself. The structure of the group is as follows.

```
STRUCTURE/GROUP_/
  integer*2          size
  byte               group_mask
  byte               tech_type
  byte               crate
  byte               first
  byte               nmodules
  byte               module_type
```

```
    integer*2            data(2000)
  END STRUCTURE
```

size : group size = data size + 4(group header size), in 2-byte words

group_mask :

tech_type : technology type. The following technology types are defined.

- **0** : no technology
- **1** : LSDT
- **2** : Brookhaven CSC
- **3** : Dubna RDT
- **4** : Michigan RDT
- **5** : RPC
- **6** : honeycomb chambers
- **7** : Houston CSC
- **8** : Dubna CSC
- **11** : TTR magnet
- **12** : TTR scintillators
- **13** : TTR Iarocci tubes

crate : crate number

first : first CAMAC station number

nmodules : number of modules in the group

module_type : type of the module. Module types are defined as follows.

- **0** : empty slot
- **1** : LeCroy ADC 2249A
- **2** : Philips TDC 7186H
- **3** : NIM trigger module TU7800
- **4** : LeCroy scaler 4434
- **5** : Kinetic ADC 3512
- **6** : LeCroy PLM 2365
- **7** : Lecroy PLU 4508
- **8** : STAS (Iarocci readout interface)
- **9** : LeCroy TDC 2277
- **12** : GPIB crate controller 3988

data : group data, array length 2000 in the definition was chosen to cover maximal event length

end-of-event marker : 0xFFFFFFFF (4 bytes)

2 Data types used by TOR

Most of the data used by TOR are organized in structures. Here the description of all the structures and constants used by TOR is presented.

2.1 Constants and data types of general use

The following constants are general purpose constants used in different parts of TOR code.

```
real*8          PI, PI2, PI4
parameter       (PI = 3.141592654, PI2 = PI/2., PI4 = PI/4.)
real*8          LARGE_VALUE
parameter       (LARGE_VALUE = 1.e10)
real*8          SPEED_OF_LIGHT
parameter       (SPEED_OF_LIGHT = 33.3564)
integer         MAX_HITS
parameter       (MAX_HITS = 20)
integer         MAX_BITS
parameter       (MAX_BITS = 100)
integer         UNDEFINED
parameter       (UNDEFINED = 0)
integer*4       EOE_MARKER
parameter       (EOE_MARKER = -1)
```

Some of the constants above have self-explaining names, others require additional explanation.

MAX_HITS : maximal number of hits for any track processed by TOR

MAX_BITS : number of TOR steering bits

EOE_MARKER : end-of-event marker (0xFFFFFFFF).

The structures described below do not belong to any certain technology.

```
STRUCTURE/LINE_/
   real*8          slope
   real*8          inter
   real*8          tx, ty
   real*8          nx, ny
   real*8          x0, y0
   real*8          chi2
   real*8          rho2
   real*8          errs
   real*8          erri
   real*8          cov(2,2)
END STRUCTURE
```

slope : straight line slope

inter : straight line intercept

tx,ty : X- and Y- components of unit vector tangent to the straight line

nx,ny : X- and Y- components of unit vector normal to the straight line

x0,y0 : X- and Y-coordinates of some point on the straight line (base point)

chi2** : chi^2 of straight line fit

rho2** : rho^2 (mean residual squared)

errs : error in straight line slope

erri : error in straight line intercept

cov : error matrix of straight line LSQ fit

```
STRUCTURE/FILE_/
   integer            file
   character*84       name
END STRUCTURE
```

file : C pointer to file descriptor

name : file name

```
STRUCTURE /PARTICLE_/
   real*8             mass
   real*8             charge
   real*8             r(3)
   real*8             p(3)
   real*8             energy
   real*8             pabs
   real*8             beta
   real*8             gamma
END STRUCTURE
```

mass : particle mass in GeV

charge : particle charge (in units of electron charge)

r : current particle position

p : vector of particle momentum in GeV

energy : particle energy in GeV

pabs : $\sqrt{p(1)^2 + p(2)^2 + p(3)^2}$

beta : pabs/energy

gamma : energy/mass

```
STRUCTURE/TRACK_/
   record/PARTICLE_/   p0
   record/PARTICLE_/   p
   record/LINE_ /      l0
   record/LINE_ /      l
```

```
    record/TUBE_ /        hit(MAX_HITS)
    integer               nhits
    integer               nhits_per_layer(MAX_HITS)
    integer               niter
    logical               found
  END STRUCTURE
```

p0 : Monte Carlo particle parameters (not used by TOR)

p : reconstructed particle parameters (not used by TOR)

l0 : track parameters starting at the iteration

l : track parameters calculated after the iteration

hit : array of track hits

nhits_per_layer(i) : number of track hits in i-th layer

niter : number of iterations required for fitting procedure to converge

found : .TRUE. if track was found, otherwise .FALSE.

2.2 Constants and data types describing TTR event

Constants below are different **event.daq_mask** codes.

```
integer*4     NORMAL_EVENT_CODE
parameter     (NORMAL_EVENT_CODE = 0)
integer*4     COMMENT_EVENT_CODE
parameter     (COMMENT_EVENT_CODE = 1)
integer*4     LTHRESH_EVENT_CODE
parameter     (LTHRESH_EVENT_CODE = 2)
integer*4     UTHRESH_EVENT_CODE
parameter     (UTHRESH_EVENT_CODE = 3)
integer*4     SPECIAL_EVENT_CODE
parameter     (SPECIAL_EVENT_CODE = 4)
integer*4     INITIAL_COMMENT_EVENT_CODE
parameter     (INITIAL_COMMENT_EVENT_CODE = 5)
integer*4     VOLTAGE_COMMENT_EVENT_CODE
parameter     (VOLTAGE_COMMENT_EVENT_CODE = 6)
integer*4     CALIBRATION_COMMENT_EVENT_CODE
parameter     (CALIBRATION_COMMENT_EVENT_CODE = 7)
integer*4     MONTE_CARLO_EVENT_CODE
parameter     (MONTE_CARLO_EVENT_CODE = 8)
```

Strustures /GROUP_/ and /TTR_EVENT_/ are described in section 1.5.

```
STRUCTURE/GROUP_/
  integer*2        size
  byte             group_mask
  byte             tech_type
  byte             crate
```

```
    byte            first
    byte            nmodules
    byte            module_type
    integer*2       data(2000)
  END STRUCTURE


  STRUCTURE/TTR_EVENT_/
    integer*2       size
    integer*2       daq_mask
    integer*2       trigger_mask
    integer*2       run_number
    integer*4       number
    integer*4       date_time
    record/GROUP_/  group
  END STRUCTURE
```

2.3 CAMAC description in TOR

```
  STRUCTURE/TDC_CHANNEL_/
    real*8          count
    real*8          time
    real*8          dist
    real*8          slope
    real*8          intercept
    real*8          zero
    real*8          overflow
    integer*4       plane
    integer*4       tube
  END STRUCTURE
```

count : raw TDC data (TDC counts+0.5)

time : time measured (time = count*slope+intercept)

dist : drift distance reconstructed

slope : TDC channel slope

intercept : TDC channel intercept

zero : time measured, corresponding to drift time equal to zero

overflow : maximal readings in readout channel

plane : plane number corresponding to readout channel

tube : tube number corresponding to readout channel

```
  STRUCTURE/MODULE_/
    integer         module
    integer         channel
  END STRUCTURE
```

module : sequential number of CAMAC module

channel : channel subaddress within the module

```
STRUCTURE/TDC_PH7186H_/
   record/TDC_CHANNEL_/     channel(16)
   integer*4                nchannels
   integer*4                end_of_record
END STRUCTURE
```

channel(i) : description of i-th channel in TDC module

nchannels : number of channels in TDC module

end_of_record : place holder

```
STRUCTURE/TDC_LRS2277_/
   record/TDC_CHANNEL_/     channel(32)
   integer*4                nchannels
   integer*4                end_of_record
END STRUCTURE
```

channel(i) : description of i-th channel in TDC module

nchannels : number of channels in TDC module

end_of_record : place holder

```
STRUCTURE/ADC_CHANNEL_/
   real*8                   count
   real*8                   pedestal
   real*8                   slope
   real*8                   overflow
   real*8                   threshold
   real*8                   amplitude
END STRUCTURE
```

count : ADC readings in readout channel

pedestal : pedestal value in given channel

slope : slope in given channel

overflow : maximal possible (2^{nbits}-1) reading in the channel

threshold : minimal amplitude allowed (not used to the moment)

amplitude : charge read out from this channel : slope*count+pedestal

```
STRUCTURE/ADC_LRS2249A_/
   record/ADC_CHANNEL_/     channel(12)
   integer*4                nchannels
   integer*4                end_of_record
END STRUCTURE
```

channel(i) : description of i-th channel in ADC module

nchannels : number of channels in ADC module

end_of_record : place holder

2.4 Constants and data types used by TOR fitting procedures

```
integer      MAX_FIT_POINTS
parameter    (MAX_FIT_POINTS = 100)
```

```
STRUCTURE/LSQ_POINT_/
   real*8                x
   real*8                y
   real*8                ey
   logical*4             rejected
   integer*4             end_of_record
END STRUCTURE
```

x,y : point X- and Y-coordinates (LSQ fit is assumed to be performed in 2-dimensional space)

ey : point error

rejected : .TRUE. if point was considered as "bad" by fitting procedure, otherwise .FALSE.

end_of_record : place holder

```
STRUCTURE/LSQ_DATA_/
   record/LSQ_POINT_/    pt(MAX_FIT_POINTS)
   integer*4             npoints
   integer*4             end_of_record
END STRUCTURE
```

pt : array of points to be fitted

npoints : number of points to be fitted

end_of_record : place holder

```
STRUCTURE/LSQ_SUMS_/
   real*8                n, w, x, y
   real*8                x2, xy, y2
   real*8                xmean, ymean, x2mean, y2mean
   real*8                sigxx, sigxy, sigyy
END STRUCTURE
```

Let **s** to be variable of type /LSQ_SUMS_/.

n : number of points fitted

w : $\sum w_i$ - sum of weights

x, y : $\sum x_i w_i$, $\sum y_i w_i$

x2, xy, y2 : $\sum x_i^2 w_i, \sum x_i y_i w_i, \sum y_i^2 w_i$

xmean = s.x/s.w

ymean = s.y/s.w

x2mean = s.x2/s.w

y2mean = s.y2/s.w

sigxx = $s.x2mean - s.xmean^2$

sigxy = s.xy/s.n-s.xmean*s.ymean

sigyy = $s.y2mean - s.ymean^2$

2.5 Constants and data types used for histogramming

The constants below define dimensions of different arrays.

```
integer     MAX_HIST_SET
parameter   (MAX_HIST_SET = 10)
integer     MAX_NTUPLE
parameter   (MAX_NTUPLE = 10)
integer     MAX_HIST_PAR
parameter   (MAX_HIST_PAR = 2000)
integer     NTUPLE_SIZE
parameter   (NTUPLE_SIZE = 100)
```

MAX_HIST_SET : maximal number of histogram sets filled simultaneously by TOR

MAX_NTUPLE : maximal number of ntuplas filled simultaneously by TOR

MAX_HIST_PAR : maximal number of histogramming parameters for one technology

NTUPLE_SIZE : maximal number of parameters in one ntuple

```
STRUCTURE/TTR_HIST_PAR_/
   real*4                    evt (MAX_HIST_PAR)
   real*4                    sc (MAX_HIST_PAR)
   real*4                    iar (MAX_HIST_PAR)
   real*4                    pdtr(MAX_HIST_PAR)
   real*4                    pdtu(MAX_HIST_PAR)
   real*4                    cscr(MAX_HIST_PAR)
END STRUCTURE
```

evt : array of general event histogramming parameters

sc : array of histogramming parameters for TTR scintillation counters

iar : array of histogramming parameters for TTR Iarocci tubes

pdtr : array of histogramming parameters for PDTR

pdtu : array of histogramming parameters for PDTU

cscr : array of histogramming parameters for CSCR

```
STRUCTURE/HIST_SET_/
   character*4          technology
   character*4          name
   real                 xmin
   real                 xmax
   integer              nx
   real*4               vmx
END STRUCTURE
```

technology : name of the technology for which histogram set is booked

name : histogram set name

xmin, xmax, nx, vmx : lower and upper limits, number of channels and packing parameter (assumed to be the same for all the histograms in the histogram set)

```
STRUCTURE/NTUPLE_/
   integer              id
   integer              nvars
   character*4          name(NTUPLE_SIZE)
   integer              p_ (NTUPLE_SIZE)
END STRUCTURE
```

id : ID of ntuple (to be used by HBOOK)

nvars : number of variables in ntuple

name(i) : symbolic name of i-th parameter

p_(i) : pointer to i-th ntuple parameter

2.6 Constants and data types for TTR scintillation counters

```
integer     MAX_SC_COUNTERS
parameter   (MAX_SC_COUNTERS = 24)
integer     MAX_SC_PLANES
parameter   (MAX_SC_PLANES = 4)
integer     MAX_SC_HITS
parameter   (MAX_SC_HITS = 10)
integer     MAX_SC_TDC
parameter   (MAX_SC_TDC = 64)
integer     MAX_SC_ADC
parameter   (MAX_SC_ADC = 48)
```

MAX_SC_COUNTERS : number of TTR scintillation counters

MAX_SC_PLANES : number of scintillation counter planes

MAX_SC_HITS : maximal allowed number of **sc_track** hits

MAX_SC_TDC : number of time readout channels from scintillation counters

MAX_SC_ADC : number of analog readout channels from scintillation counters

```
STRUCTURE/SC_TABLE_/
   record/MODULE_/        tdc(MAX_SC_TDC)
   record/MODULE_/        adc(MAX_SC_ADC)
END STRUCTURE
```

tdc(i) : reference to TDC channel, corresponding to i-th time readout channel from scintillation counters

adc(i) : reference to ADC channel, corresponding to i-th analog readout channel from scintillation counters

```
STRUCTURE/SCOUNTER_/
   real*8          xc, yc, zc
   real*8          xsize, ysize, zsize
   real*8          v0
   real*8          x0
   integer*4       iaa
   integer*4       ita
   real*8          delay_a
   real*8          corr_a(3)
   integer*4       iab
   integer*4       itb
   real*8          delay_b
   real*8          corr_b(3)
   real*8          xhit
   real*8          tr_minus_t1
   real*8          tr_minus_t25
   integer         plane
   logical         hit
END STRUCTURE
```

xc,yc,zc : X-, Y-, Z-coordinates of the scintillation counter center

xsize, ysize, zsize : X-, Y- and Z-dimensions of the scintillation counter

v0, x0 : constants used for determination of hit X-coordinate : $x_{hit} = v0(T_B - T_A) + x0$, where T_A and T_B are times, measured from two ends of scintillation counter

iaa : number of ADC readout channel for PMT_A

ita : number of TDC readout channel for PMT_A

delay_a : time delay in PMT_A time readout channel

corr_a : walk correction coefficients in PMT_A channel (not used for the moment)

iab, itb, delay_b, corr_b : the same for PMT_B time and analog readout channels

xhit : X-coordinate of hit

tr_minus_t1 : $T_{trigger} - T_1$, where $T_{trigger}$ - trigger time, T_1 - time, when muon hit scintillation counter

tr_minus_t25 : $T_{trigger} - T_{25}$, where T_{25} - time when muon crossed plane Z = 250.

plane : scintillation plane number for this counter

hit : .TRUE. if counter was hit, otherwise .FALSE.

```
STRUCTURE/SC_PLANE_/
   integer*4          first
   integer*4          ncounters
END STRUCTURE
```

first : number of the first scintillation counter in given plane; counters in one plane have sequentially increasing numbers : first, first+1, first+2, ...

ncounters : number of scintillation counters in scintillation plane

```
STRUCTURE/SC_HIT_/
real*8   x, y, z
real*8   ex, ey
integer  counter
logical  rejected
END STRUCTURE
```

x, y, z : space coordinates of scintillation counter hit in TTR coordinate system

ex, ey : errors in X and Y used for track fit

counter : number of scintillation counter (backward reference)

rejected : .TRUE. if hit was rejected by track fitting procedure, otherwise .FALSE.

```
STRUCTURE/SC_TRACK_/
   real*8               costh
   record/LINE_ /       lx
   record/LINE_ /       ly
   record/SC_HIT_/      hit(MAX_SC_HITS)
   integer*4            nhits
   integer*4            nhits_per_layer(MAX_SC_PLANES)
   logical*4            found
END STRUCTURE
```

costh : track cosine theta

lx : straight line fitted to the track in XZ plane

ly : straight line fitted to the track in XY plane

hit : array of track hits

nhits : number of hits found on the track

nhits_per_layer(i) : number of track hits in i-th scintillation plane

found : .TRUE. if **sc_track** was found, otherwise .FALSE.

2.7 Constants and data types used for TTR Iarocci tubes description

```
integer     MAX_IAR_CLUSTERS
parameter   (MAX_IAR_CLUSTERS = 50)
integer     N_IAR_PLANES
parameter   (N_IAR_PLANES = 4)
```

MAX_IAR_CLUSTERS : maximal number of clusters processed in TTR Iarocci tubes

N_IAR_PLANES : number of TTR Iarocci planes

```
STRUCTURE/IAROCCI_CLUSTER_/
  integer*4                    first
  integer*4                    size
  real*8                       x, y, z
  real*8                       dx, dy
END STRUCTURE
```

first : number of the first cluster strip

size : cluster size (number of strips in the cluster)

x, y, z : cluster coordinates in TTR system

dx, dy : cluster residuals in X and in Y from **sc_track**

```
STRUCTURE/IAROCCI_PLANE_/
  real*8                       xc, yc, zc
  real*8                       xsize, ysize
  real*8                       x_sc_track
  real*8                       y_sc_track
  real*8                       strip_size
  integer*4                    nstrips
  integer*4                    nxstrips
  record/IAROCCI_CLUSTER_/     cl(MAX_IAR_CLUSTERS)
  integer*4                    chain
  integer*4                    nxclust
  integer*4                    nyclust
  integer*4                    nclusters
  integer*4                    xclosest
  integer*4                    yclosest
END STRUCTURE
```

xc, yc, zc : space coordinates of the Iarocci plane center

xsize, ysize : X- and Y-dimensions of the Iarocci plane

x_sc_track : X-coordinate of **sc_track** in the given Iarocci plane

y_sc_track : Y-coordinate of **sc_track** in the given Iarocci plane

strip_size : readout strip pitch

nstrips : total number of readout strips in the given Iarocci plane

cl : array of clusters found in the given plane

chain : readout chain for this plane

nxclust : number of X-clusters (clusters on X-strips)

nyclust : number of Y-clusters (clusters on Y-strips)

nclusters : total number of clusters found in this plane (nxclust+nyclust)

xclosest : X-cluster, closest to **sc_track**

xclosest : X-cluster, closest to **sc_track**

2.8 Constants and types used for drift chambers description

```
integer      MAX_RDT_CHAMBERS
parameter    (MAX_RDT_CHAMBERS = 2)
integer      MAX_RDT_LAYERS
parameter    (MAX_RDT_LAYERS   = 8)
integer      MAX_RDT_TUBES
parameter    (MAX_RDT_TUBES    = 32)
integer*4    MAX_RDT_CHANNELS
parameter    (MAX_RDT_CHANNELS = 128)
```

MAX_RDT_CHAMBERS : maximal number of RDT chambers of the same technology

MAX_RDT_LAYERS : maximal number of layers in one RDT chamber

MAX_RDT_TUBES : maximal number of tubes in one layer of RDT chamber

MAX_RDT_CHANNELS : maximal number of readout channels for one RDT technology

```
STRUCTURE/TUBE_/
  real*8          wx, wy, wz
  real*8          cx, cy, cz
  real*8          dwy, wyslope
  real*8          dwz, wzslope
  record/POINT_/  h
  real*8          rho
  real*8          sig
  real*8          time
  real*8          dr
  real*8          dy
  real*8          xhit
  real*8          tr_minus_t1
  integer         tdc_channel
  integer         counts
  integer         iw
  integer         layer
  integer         ich
  logical         hit
```

```
    logical          rejected
    logical          used
    integer          nhits
    integer          end_of_record
  END STRUCTURE
```

wx,wy,wz : X-,Y-,Z-coordinates of one end of the wire in TTR coordinate system

cx,cy,cy : direction cosines of the wire in TTR coordinate system

dwy : Y-displacement of the wire end from its nominal position (used only for input)

wyslope : wire slope in XY-plane (used only for input)

dwz : Z-displacement of the wire end from its nominal position (used only for input)

wzslope : wire slope in XZ-plane (used only for input)

h : track coordinates in its point closest to the wire

rho : distance from the track to the wire

sig : drift tube resolution corresponding to the hit drift distance

time : drift time measured in the drift tube

dr : hit residual from the track in R

dy : hit residual from the track in Y

xhit : hit X-coordinate

tr_minus_t1 : used when /TUBE_/ structure is used for description of hits in scintillators (for **sc_track**) and contains the difference between trigger time and time when muon hit scintillation counter

tdc_channel : TDC channel number for this tube

counts : raw TDC data read out from this tube

iw : tube number within the layer

layer : layer number within the tube

ich : drift chamber number

hit : .TRUE. if tube was hit (reasonable time was read out from the tube), otherwise .FALSE.

rejected : .TRUE. if hit corresponding to this tube was rejected by track fitting procedure, otherwise - .FALSE.

used : .TRUE. if tube is to be used in the analysis, otherwize - .FALSE.

nhits : number of hits in this channel (for multihit TDC's)

end_of_record : place holder

```
STRUCTURE/PDT_CHAMBER_/
   real*8                    x0, y0, z0
   real*8                    theta
   real*8                    phi
   real*8                    psi
   real*8                    dy(MAX_RDT_LAYERS)
   real*8                    dz(MAX_RDT_LAYERS)
   real*8                    step
   real*8                    lwire
   real*8                    rtube
   real*8                    rwire
   real*8                    wall
   record /TUBE_/            tube(MAX_RDT_TUBES,MAX_RDT_LAYERS)
   integer                   nlayers
   integer                   ntubes(MAX_RDT_LAYERS)
   integer                   nhits
   integer                   nhits_per_layer(MAX_RDT_LAYERS)
   logical                   used
   integer*4                 end_of_record
END STRUCTURE
```

x0,y0,z0 : coordinates of chamber reference point

theta : chamber rotation angle in XY plane

phi, psi : 2 other Eiler rotation angles of the chamber (not used to the moment)

dy(i) : Y-offset of the 1-st wire in i-th layer with respect to chamber reference point

dz(i) : Z-offset of the 1-st wire in i-th layer with respect to chamber reference point

step : step between wires

lwire : wire length

rtube : tube inner radius

rwire : wire radius (not used in reconstruction)

wall : tube wall thickness (not used in reconstruction)

tube(i,j) : *record, describing i-th tube in j-th layer*

nlayers : number of drift tube layers in the chamber

ntubes(i) : number of tubes in i-th layer

nhits : total number of hits in drift chamber (for given event)

nhits_per_layer(i) : number of hits in i-th layer of the chamber

used : .TRUE. if chamber is to be processed by TOR, otherwise .FALSE.

end_of_record : place holder

Described below are the structures used for parametrization of space-time function R(T) for drift chambers

The following constants define dimensions of different arrays.

```
integer*4      INT_TABLE_SIZE
parameter      (INT_TABLE_SIZE = 400)
integer*4      MAX_RT_CORRECTIONS
parameter      (MAX_RT_CORRECTIONS = 10)
integer*4      MAX_CORR_DEGREE
parameter      (MAX_CORR_DEGREE = 10)

STRUCTURE/INTERPOLATION_TABLE_/
   real*8                                y(INT_TABLE_SIZE)
   real*8                                x(INT_TABLE_SIZE)
   real*8                                xmin
   real*8                                step
   integer                               nx
   integer                               end_of_record
END STRUCTURE
```

y : array of function values $y_i = F(x_i)$

x : array of X-coordinates

xmin : X-coordinate of the 1-st point

step : step in X between points

nx : number of points where function is tabulated

end_of_record : place holder

```
STRUCTURE/PDT_RT_RECORD_/
   record/INTERPOLATION_TABLE_int_table
   record/FILE_/                 file
   real*8                        tmax
   real*8                        rmax
   logical*4                     defined
   integer*4                     id
   integer*4                     end_of_record
END STRUCTURE
```

int_table : interpolation table used to tabulate R(T) dependence

file : description of RZ file containing histogram with R(T) dependence

tmax : maximal time allowed, if T > tmax then R(T) = rmax

rmax : inner tube radius

defined : .TRUE. if R(T) dependence if defined, .FALSE. otherwise

id : identifier of the histogram containing R(T) dependence

end_of_record : place holder

2.9 Constants and data types used for CSC description

Constants defined below are used in different DIMENSION operators

```
integer*4      MAX_CSC_STRIPS
parameter      (MAX_CSC_STRIPS = 192)
integer*4      MAX_CSC_PLANES
parameter      (MAX_CSC_PLANES = 4)
integer*4      MAX_CSC_CLUSTERS
parameter      (MAX_CSC_CLUSTERS = 20)
integer*4      MAX_CSC_CHANNELS
parameter      (MAX_CSC_CHANNELS = 192)
integer*4      MAX_CSC_CHAMBERS
parameter      (MAX_CSC_CHAMBERS = 2)
```

MAX_CSC_STRIPS : maximal total number of CSCR strips (sum over all planes)

MAX_CSC_PLANES : maximal number of planes in CSCR chamber

MAX_CSC_CLUSTERS : maximal number of clusters per plane allowed

MAX_CSC_CHANNELS : maximal number of CSCR channels read per event

MAX_CSC_CHAMBERS : maximal number of CSCR chambers to be processed

Structure /CSC_STRIP_/ is used for strip description.

```
STRUCTURE/CSC_STRIP_/
   real*8          count
   real*8          cprime
   real*8          q
   real*8          qnorm
   real*8          ped
   real*8          pwid
   real*8          c0, c1, c2, c3
   real*8          y
   real*8          z
   integer*4       chamber
   integer*4       plane
   integer*4       istrip
   logical*4       hit
END STRUCTURE
```

Let s to be variable of /CSC_STRIP_/ type : record/CSC_STRIP_/ s
Then :

s.count : raw ADC data read in this channel

s.cprime : ADC data corrected for correlated noise

s.q : charge induced on this strip. Scale here is arbitrary but should be the same for all strips

s.ped : pedestal position in the ADC channel corresponding to this strip (in units of ADC counts)

s.pwid : pedestal width in the corresponding ADC channel (in units of ADC counts)

s.qnorm : normalized charge in units of pedestal width :
s.qnorm = (s.count-s.ped)/s.pwid
it is used when dealing with correlated noise

s.c0,s.c1,s.c2,s.c3 : calibration constants for the channel corresponding to the given strip :

$$s.q = s.c1*(s.cprime-s.ped)+s.c2*(s.cprime^2-s.ped^2) +s.c3*(s.cprime^3-s.ped^3)$$
(we use qubic parabola to fit ADC calibration curve)

s.y : strip Y-coordinate in chamber local system

s.z : strip Z-coordinate in TTR system

s.chamber : chamber number for given strip

s.plane : plane number within the chamber

s.istrip : strip number within the plane

s.hit : .TRUE. if strip was hit, .FALSE. otherwise

Structure /CSC_CLUSTER_/ defines cluster record.

```
STRUCTURE/CSC_CLUSTER_/
   real*8          x
   real*8          y
   real*8          z
   real*8          q
   real*8          qmax
   real*8          r1
   real*8          r2
   real*8          delta
   real*8          dy
   real*8          dz
   integer*4       chamber
   integer*4       plane
   integer*4       size
   integer*4       max
END STRUCTURE
```

Let cl to be variable of /CSC_CLUSTER_/ type : record/CSC_CLUSTER_/ cl
Then

cl.x, cl.y, cl.z : cluster coordinates in TTR coordinate system

cl.q : cluster charge (sum of strip charges)

cl.qmax : charge of the central strip (strip with maximal charge)

cl.max : number (within the strip plane) of the central strip

cl.r1 = strip(max-1).q/cl.qmax

cl.r2 = strip(max+1).q/cl.qmax

cl.delta = cl.y-Y(central strip)

cl.dy : Y-residual from the global track (now from RDT track)

cl.dz : Z-residual from the track (not defined now)

cl.chamber : CSC chamber number (back reference)

cl.plane : strip plane number within the chamber (back reference)

cl.size : cluster size (number of strips in cluster)

Structure /CSC_CHAMBER_/ gives the description of cathode strip chamber.

```
STRUCTURE/CSC_CHAMBER_/
   real*8                    x0, y0, z0
   real*8                    phi
   real*8                    psi
   real*8                    pitch
   real*8                    wstrip
   real*8                    lstrip
   record/CSC_PLANE_/        plane(MAX_CSC_PLANES)
   integer*4                 nplanes
   integer*4                 nhits
   integer*4                 nclusters
   logical*4                 used END STRUCTURE
END STRUCTURE
```

Let csc to be variable of /CSC_CHAMBER_/ type : record/CSC_CHAMBER_/ csc

csc.x0, csc.y0, csc.z0 : coordinates of the chamber reference point in TTR system

csc.phi, csc.psi : Eyler rotation angles (not used to the moment)

csc.pitch : strip pitch (assumed to be the same for all planes)

csc.wstrip : real strip width (not used to the moment)

csc.lstrip : strip length (not used to the moment)

csc.plane(i) : description of i-th strip plane

csc.nplanes : number of strip planes in the given CSC chamber

csc.nhits : total number of strips hit in CSC chamber

csc.nclusters : total number of clusters found in CSC chamber

csc.used : .TRUE. if CSC chamber is to be processed, .FALSE. otherwise.

Structure /CSC_PLANE_/ describes one plane of cathode strip chamber.

```
STRUCTURE/CSC_PLANE_/
   real*8                    theta
   real*8                    dy
   real*8                    dz
   real*8                    pshift
   real*8                    xtrack
   real*8                    ytrack
   record/CSC_STRIP_/        strip (MAX_CSC_STRIPS)
   record/CSC_CLUSTER_/      cluster(MAX_CSC_CLUSTERS)
   integer*4                 closest
   integer*4                 nstrips
   integer*4                 nhits
   integer*4                 nclusters
   logical*4                 used
END STRUCTURE
```

Let **pln** to be variable of /CSC_PLANE_/ type : record/CSC_PLANE_/ pln Then

pln.dy : Y-coordinate of the 1st strip in given plane is defined as
y = csc.y0+pln.dy

pln.dz : Z-coordinate of the strip plane ipl is defined as
z = csc.z0+pln.dz

pln.strip(i) : description of i-th strip in given plane

pln.cl(i) : description of i-th cluster found in given plane

pln.pshift : coherent pedestal shift in given plane (dimensionless units)

pln.ytrack : Y-coordinate of the external track in plane

pln.xtrack : X-coordinate of the external track in plane

pln.nstrips : number of strips in given plane

pln.nhits : number of strips hit in given plane

pln.nclusters : number of clusters found in given plane

pln.used : .TRUE. if plane is to be processed, .FALSE. otherwise

Structure /CSC_TABLE_/ contains scales setting the correspondence between different types of numeration

```
STRUCTURE/CSC_TABLE_/
   integer                   channel(MAX_CSC_STRIPS,MAX_CSC_PLANES,
+                                   MAX_CSC_CHAMBERS)
   integer                   strip (MAX_CSC_CHANNELS)
   integer                   plane (MAX_CSC_CHANNELS)
   integer                   chamber(MAX_CSC_CHANNELS)
END STRUCTURE
```

Let **t** to be variable of /CSC_TABLE_/ type : record/CSC_TABLE_/ t
Then

t.channel(ist,ipl,ich) : number of readout channel (word in CSCR data group) corresponding to strip ist in plane ipl in chamber ich

t.strip(i) : strip number (within plane) corresponding to readout channel i

t.plane(i) : plane number (within chamber) corresponding to readout channel i

t.chamber(i) : CSC chamber number corresponding to readout channel i

3 TOR global variables

This section contains description of global variables used by TOR

3.1 Common-block /TRACKS/

This common-block contains information about tracks found by TOR.

```
 common/TRACKS/     global_track, pdtr_seg(MAX_CHAMBERS),
+                   pdtu_seg(MAX_CHAMBERS), pdtr_track,
+                   pdtu_track,
+                   sc_track, iar_track, ttr_track,
+                   track_found
 record/TRACK_/     global_track, pdtr_seg, pdtu_seg, pdtr_track,
+                   pdtu_track
 record/SC_TRACK_/  sc_track
 record/SC_TRACK_/  iar_track
 record/SC_TRACK_/  ttr_track
 logical            track_found
```

global_track : track, reconstructed using hits from all chambers processed

pdtr_seg(i) : track segment reconstructed using hits found in i-th Dubna RDT chamber

pdtu_seg(i) : track segment reconstructed using hits found in i-th Michigan RDT chamber (really there is only 1 Michigan RDT chamber)

pdtr_track : track reconstructed usind hits only from Dubna RDT chambers

pdtu_track : track reconstructed usind hits only from Michigan RDT chambers

sc_track : track reconstructed usind only hits in TTR scintillation counters

iar_track : track reconstructed using hits in TTR Iarocci tubes (to the moment it is reconstructed using 2 top Iarocci planes and only in XZ-plane)

ttr_track : track reconstructed using only information given by TTR scintillation counters and Iarocci's (to the moment ttr_track = sc_track)

3.2 Common-block /TTR_EVENT/

Variables in this common-block define TTR event read by TOR.

```
      common/TTR_EVENT/ event, event_code, n_events_read,
     +                  sc_tdc_group_present, sc_adc_group_present,
     +                  iarocci_group_present, pdtr_group_present,
     +                  pdtu_group_present, cscr_group_present,
     +                  sc_tdc_group_, sc_adc_group_, iarocci_group_,
     +                  pdtr_group_, pdtu_group_, cscr_group_
      record/TTR_EVENT_/ event
      integer*4         event_code, n_events_read
      logical           sc_tdc_group_present, sc_adc_group_present,
     +                  iarocci_group_present, pdtr_group_present,
     +                  pdtu_group_present, cscr_group_present
      pointer           (sc_tdc_group_ , sc_tdc_group ),
     +                  (sc_adc_group_ , sc_adc_group ),
     +                  (iarocci_group_, iarocci_group),
     +                  (pdtr_group_ , pdtr_group ),
     +                  (pdtu_group_ , pdtu_group ),
     +                  (cscr_group_ , cscr_group )
      record/GROUP_/    sc_tdc_group, sc_adc_group, iarocci_group,
     +                  pdtr_group , pdtu_group , cscr_group
```

event : event read by TOR

event_code : TTR event code. There are several types of TTR events :

normal data event	(NORMAL_EVENT_CODE	= 0)
comment event	(COMMENT_EVENT_CODE	= 1)
lower thresholds event	(LTHRESH_EVENT_CODE	= 2)
upper thresholds event	(UTHRESH_EVENT_CODE	= 3)
special event	(SPECIAL_EVENT_CODE	= 4)
initial comment event	(INITIAL_COMMENT_EVENT_CODE	= 5)
voltage comment event	(VOLTAGE_COMMENT_EVENT_CODE	= 6)
calibration comment event	(CALIBRATION_COMMENT_EVENT_CODE	= 7)
Monte Carlo event	(MONTE_CARLO_EVENT_CODE	= 8)

n_events_read : number of events read by TOR to the moment

sc_tdc_group_present = .TRUE. if scintillation counters TDC group is present in the event read and = .FALSE. otherwise

sc_adc_group_present = .TRUE. if scintillation counters ADC group is present in the event read and = .FALSE. otherwise

iarocci_group_present = .TRUE. if Iarocci tubes group is present in the event read = .FALSE. otherwise

pdtr_group_present = .TRUE. if Dubna RDT group is present in the event read = and .FALSE. otherwise

pdtu_group_present = .TRUE. if Michigan RDT group is present in the event read and = .FALSE. otherwise

cscr_group_present = .TRUE. if Dubna CSC group is present in the event read and = .FALSE. otherwise

sc_tdc_group_ : pointer to scintillation counters TDC group

sc_adc_group_ : pointer to scintillation counters ADC group

iarocci_group_ : pointer to Iarocci tubes group

pdtr_group_ : pointer to Dubna RDT group

pdtu_group_ : pointer to Michigan RDT group

cscr_group_ : pointer to Dubna CSC group

3.3 Common-block /GLOBAL_VARS/

Common-block /GLOBAL_VARS/ contains TOR global variables of general use.

```
      common/GLOBAL_VARS/ end_of_run, end_of_event, error,
     +                    sc_code, iar_code, pdtr_code, pdtu_code,
     +                    cscr_code,
     +                    run_init_file,
     +                    run_number, min_event , max_event,
     +                    min_evt_debugged, max_evt_debugged,
     +                    last_event_debugged, debug_step,
     +                    input_data_type,
     +                    process_scounters, process_iarocci,
     +                    process_pdtr, process_cscr, process_pdtu,
     +                    technology_code, n_format_errors,
     +                    info_step
      logical             end_of_run, end_of_event
      integer*4           run_number, min_event, max_event
      integer*4           n_format_errors, info_step
      real*8              sc_code, iar_code, pdtr_code, pdtu_code,
     +                    cscr_code
      record/FILE_/       run_init_file
      real*8              error
      integer             min_evt_debugged, max_evt_debugged
      integer             last_event_debugged, debug_step
      integer             input_data_type
      logical             process_scounters, process_iarocci
      logical             process_pdtu
      logical             process_pdtr, process_cscr
      integer*4           technology_code
```

end_of_run : not used

end_of_event : not used

error : event reconstruction code :

-100 : no track reconstructed from scintillation counters
-10 : no track in PDTR
¿ 0 : good event, number of track hits in PDTR

sc_code : TTR scintillators reconstruction code

0 : sc counters o'k
100 : sc_track not found
-1 : too many (> 10) hits in scintillators

iar_code : TTR Iarocci's reconstruction code

= number of points on iar_track (2 if iar_track was found, 0 - if not)
if total number of clusters in all the Iarocci planes exceeds MAX_IAR_CLUSTERS, then 10*(nclusters-MAX_IAR_CLUSTERS) is added to **iar_code**

pdtr_code : Dubna RDT reconstruction code

pdtu_code : Michigan RDT reconstruction code

cscr_code : Dubna CSC reconstruction code

run_init_file : record defining .init run name

run_number : current run number

min_event : minimal number of event to be processed

max_event : maximal number of event to be processed

min_evt_debugged : minimal number of event for which debug print output is required

max_evt_debugged : maximal number of event for which debug print output is required

last_evt_debugged : number of last event for which event number was printed

process_scounters = .TRUE. if processing of scintillation counters was requested, otherwise = .FALSE.

process_iarocci : the same, but for iarocci tubes

process_pdtr : the same, but for Dubna RDT chambers

process_pdtu : the same, but for Michigan RDT chambers

process_cscr : the same, but for Dubna CSC chamber

technology_code : code of technology for which input data are currently read

n_format_errors : number of event format errors encountered when reading input data

info_step : not used now, but will be used!

3.4 Common-block /TTRIO/

Common-block /TTRIO/ contains global I/O (input/output) variables

```
COMMON/TTRIO/ input, hbfile
record/FILE_/      input, hbfile
```

input : record, describing input data file

hbfile : record, describing output file used to store HBOOK histograms and ntuples

3.5 Common-block /TTR_SCINT/

Common-block /TTR_SCINT/ contains information about TTR scintillation counters.

```
      common/TTR_SCINT/       scounter(MAX_SC_COUNTERS),
     +                        sc_tdc(MAX_SC_TDC), sc_adc(MAX_SC_ADC),
     +                        sc_table,
     +                        sc_init_file,
     +                        n_sc_hit, n_sc_planes_hit,
     +                        n_hits_per_plane(MAX_SC_PLANES),
     +                        sc_plane(MAX_SC_PLANES),
     +                        n_sc_planes, n_sc_counters, n_sc_per_plane
      record/SCOUNTER_/       scounter
      record/TDC_PH7186H_/    sc_tdc
      record/ADC_LRS2249A_/   sc_adc
      record/SC_TABLE_/       sc_table
      record/FILE_/           sc_init_file
      record/SC_PLANE_/       sc_plane
      integer                 n_sc_hit
      integer                 n_sc_planes_hit
      integer                 n_hits_per_plane
      integer                 n_sc_planes
      integer                 n_sc_counters
      integer                 n_sc_per_plane
```

scounter(i) : description of i-th scintillation counter

sc_tdc(i) : description of i-th TDC module used for time measurements in scintillators

sc_adc(i) : description of i-th ADC module used for amplitude measurements in scintillators

sc_table : description of different scintillators scales

sc_init_file : description of TTR scintillators constant file

n_sc_hit : number of scintillation paddles hit (for the given event)

n_sc_planes_hit : number of scintillation planes hit (for the given event)

n_hits_per_plane(i) : number of scintillation paddles hit in i-th plane

sc_plane(i) : description of i-th scintillation counter plane

n_sc_planes : total number of scintillation counter planes

n_sc_counters : total number of scintillation counters

n_sc_per_plane : number of scintillation sounters per plane (supposed to be the same for all planes)

3.6 Common-block /TTR_IAROCCI/

Common-block /TTR_IAROCCI/ contains description of TTR iarocci tubes

```
common/TTR_IAROCCI/      iarocci_plane(4), iarocci_init_file
record/IAROCCI_PLANE_/   iarocci_plane
record/FILE_/            iarocci_init_file
```

iarocci_plane(i) : description of i-th Iarocci tubes plane

iarocci_init_file : description of Iarocci constant file

3.7 Common-block /PDTR_VARS/

Common-block /PDTR_VARS/ contains variables describing Dubna RDT chambers.

```
  common/PDTR_VARS/          pdtr_chamber(MAX_CHAMBERS),
+                            pdtr_rt (MAX_CHAMBERS),
+                            pdtr_tdc (MAX_PDTR_TDC),
+                            pdtr_table,
+                            pdtr_init_file,
+                            n_pdtr_chambers, n_hits_pdtr,
+                            pdtr_ch_number(MAX_CHAMBERS)
  record/PDT_CHAMBER_/       pdtr_chamber
  record/PDT_RT_RECORD_/     pdtr_rt
  record/TDC_PH7186H_/       pdtr_tdc
  record/PDT_TABLE_ /        pdtr_table
  record/FILE_/              pdtr_init_file
  integer                    n_pdtr_chambers, n_hits_pdtr
  integer                    pdtr_ch_number
```

pdtr_chamber(i) : record describing i-th Dubna RDT chamber (to the moment there are 2 such chambers)

pdtr_rt(i) : record containing R(T) dependence for i-th Dubna chamber

pdtr_tdc(i) : description of i-th TDC module used for Dubna chambers

pdtr_table : record, containing different Dubna chamber scales

pdtr_init_file : text file with initialization constants for Dubna chambers

n_pdtr_chambers : number of chambers to be processed

n_hits_pdtr : total number of hits in all the chambers for the event processed

pdtr_ch_number(i) : number of Dubna chamber to be processed i-th in the procession order

3.8 Common-block /CSCR_VARS/

Common-block /CSCR_VARS/contains variables describing CSC chamber built in Dubna (CSCR stands for Cathode Strip Chamber Russian).

```
  common/CSCR_VARS/          cscr_chamber(MAX_CSC_CHAMBERS),
+                            cscr_table,
+                            cscr_init_file,
+                            n_cscr_channels,
```

```
+                          n_cscr_chambers, n_hits_cscr,
+                          cscr_ch_number(MAX_CHAMBERS)
record/CSC_CHAMBER_/       cscr_chamber
record/CSC_TABLE_ /        cscr_table
record/FILE_/              cscr_init_file
integer                    n_cscr_channels
integer                    n_cscr_chambers, n_hits_cscr
integer                    cscr_ch_number
```

cscr_chamber(i) : record, containing description of i-th Dubna CSC

cscr_table : record, containing different scales describing Dubna CSC

n_cscr_chambers : number of Dubna CSC chambers to be processed

n_cscr_channels : total number of channels (strips) for Dubna CSC

n_hits_cscr : total number of strips hit in all CSCR for the event processed

cscr_ch_number(i) : number of Dubna CSC to be processed i-th in the procession order

3.9 Common-block /PDTU_VARS/

Common-block /PDTU_VARS/contains variables used for description of RDT chamber built in Michigan State University..

```
  common/PDTU_CHAMBERS/ pdtu_chamber(MAX_CHAMBERS),
+                       pdtu_rt (MAX_CHAMBERS),
+                       pdtu_tdc (MAX_PDTU_TDC),
+                       pdtu_table,
+                       pdtu_init_file,
+                       n_pdtu_chambers, n_hits_pdtu,
+                       pdtu_ch_number(MAX_CHAMBERS)
  record/PDT_CHAMBER_/ pdtu_chamber
  record/PDT_RT_RECORD_/ pdtu_rt
  record/TDC_LRS2277_/ pdtu_tdc
  record/PDT_TABLE_ /  pdtu_table
  record/FILE_/        pdtu_init_file
  integer              n_pdtu_chambers
  integer              n_hits_pdtu
  integer              pdtu_ch_number
```

pdtu_chamber(i) : record describing i-th Michigan RDT chamber (for the moment there is only 1 such chamber)

pdtu_rt(i) : record containing R(T) dependence for i-th Michigan chamber

pdtu_tdc(i) : description of i-th TDC module used for Michigan chambers (TDC LRS2277)

pdtu_table : record, containing different Michigan chamber scales

pdtu_init_file : text file with initialization constants for Michigan chambers

n_pdtu_chambers : number of Michigan chambers to be processed

n_hits_pdtu : total number of hits in all the chambers for the event processed

pdtu_ch_number(i) : number of Michigan chamber to be processed i-th in the procession order

3.10 Common-block /TZERO/

Common-block /TZERO/ contains variables used for T_0 determination for drift chambers (Dubna and Michigan ones).

```
common/TZERO/ trigger_sc, tr_minus_t1, z_top_hit
real*8        trigger_sc
real*8        tr_minus_t1
real*8        z_top_hit
```

trigger_cs : scintillation counter, defined trigger time for the given event

tr_minus_t1 : time difference between time when trigger occured and time when muon crossed the top scintillation plane

z_top_hit : Z-coordinate of the scintillator which defined trigger time

3.11 Common-block /WS/

Common-block /WS/ defines working spase used ONLY for temporary goals inside different subroutines. It is supposed that all information in /WS/ is lost after exiting the subroutine.

```
COMMON/WS / iws(1000)
integer*4   iws
real*4      ws(1000)
character*4 cws(1000)
equivalence (iws(1),ws(1)), (cws(1),ws(1))
```

3.12 Common-block /TTR_BITS/

Common-block /TTR_BITS/ contains logical switches for different debug levels

```
common/TTR_BITS/ bit(100)
logical bit
```

bit(i) = .TRUE. if i-th level of debug output is requested

bit number	routine, using this bit	Action
1	next_event_read	.TRUE. : prints information about each event format error encountered .FALSE. : prints only brief summary about the number of format errors encountered
2	next_event_read	debug of event in binary format
10		MC debug (very primitive)
41	cscr_unpack	CSCR strips debug output
42	cscr_track	CSCR clusters debug output
51		

52		
53		
54		
55		
56		PDT track debug output
61	tor_term	if .TRUE. then HINDEX output is printed in the end of run if .FALSE. no HINDEX output

3.13 Common-block /PAWC/

Common-block /PAWC/ is standard HBOOK common-block

```
common/PAWC / ipawc(-17:NWPAWC)
integer       ipawc
```

3.14 Common-block /HREQUEST/

Common-block /HREQUEST/ defines histogram sets and ntuples requested by the user as well, as event parameters, used for histogramming.

```
  common/HREQUEST/      hist_set(MAX_HIST_SET),
+                       ntuple(MAX_NTUPLE),
+                       n_hist_sets, n_ntuples,
+                       hp, fill_pdtr_hist, fill_sc_hist, fill_iar_hist,
+                       fill_cscr_hist, fill_pdtu_hist
  integer               n_hist_sets, n_ntuples
  record/HIST_SET_/     hist_set
  record/NTUPLE_ /      ntuple
  record/TTR_HIST_PAR_/hp
  logical               fill_pdtr_hist, fill_sc_hist, fill_iar_hist,
+                       fill_cscr_hist, fill_pdtu_hist
```

hist_set(i) : description of i-th histogram set requested by the user

ntuple(i) : description of i-th ntuple requested by the user

n_hist_sets : total number of histogram sets requested

n_ntuples : number of ntuples requested

hp : description of event parameters used for histogramming

fill_pdtr_hist = .TRUE. if filling of PDTR (Dubna RDT) histograms is required, otherwise = .FALSE.

fill_sc_hist : the same but for TTR scintillation counters

fill_iar_hist : the same but for TTR iarocci tubes

fill_cscr_hist : the same but for Dubna CSC chamber

fill_pdtu_hist : the same but for Michigan RDT chamber

4 Event parameters calculated by TOR

When processing the event TOR calculates a large number of event parameters to be analysed. They can be divided into so-called general parameters of event (characterizing the whole event) and parameters specific for any particular technology. In this section the general parameters of event are described, parameters calculated for different technologies are described in sections, corresponding to the technologies.

General parameters of event calculated by TOR have 4-character symbolic names **'exxx'**, where xxx stands for parameter index in **hp.evt** array.

hp.evt(1) : event number
hp.evt(2) : run number
hp.evt(3) : event size
hp.evt(4) : error (event reconstruction code)
hp.evt(5) : sc_code (sc. counters reconstruction code)
hp.evt(6) : iar_code (iarocci's reconstruction code)
hp.evt(7) : pdtr_code (Dubna RDT reconstruction code)
hp.evt(8) : pdtu_code (Michigan RDT reconstruction code)
hp.evt(9) : cscr_code (Dubna CSC reconstruction code)

The following are parameters of global track reconstructed by TOR using all available hits in the chambers processed.

hp.evt(431) : global track slope
hp.evt(432) : global track intercept
hp.evt(433) : error in global track slope
hp.evt(434) : error in global track intercept
hp.evt(435) : chi^2 of global track fit
hp.evt(436) : rho^2 of global track fit
hp.evt(437) : number of global track hits
hp.evt(438) : number of global track fit iterations

After the global track was found the following parameters of its hits are defined :

hp.evt(460+i) : drift time for i-th track hit
hp.evt(490+i) : drift distance corresponding to i-th track hit
hp.evt(520+i) = 5. if Y(wire) > Y(track)
= -5. if Y(wire) < Y(track)
hp.evt(550+i) : delta(R) = $R_{hit} - R_{track}$, where R_{track} is the distance from the track to the wire
hp.evt(580+i) : delta(Y) = $Y_{hit} - Y_{track}$ at $X = X_{hit}$
hp.evt(610+i) : plane number for i-th hit
hp.evt(640+i) : tube (or strip) number for i-th hit
hp.evt(670+i) : distance from the track to the wire for i-th hit
hp.evt(700+i) : X-coordinate of i-th hit
hp.evt(730+i) : Y-coordinate of i-th hit
hp.evt(760+i) : Z-coordinate of i-th hit
hp.evt(790+i) : delta(Z) = $Z_{hit} - Z_{track}$ at $X = X_{hit}$

5 TTR scintillation counters

TTR scintillation counters are used to provide trigger and T_0 measurements for drift technologies. For T_0 measurements each counter is viewed by 2 photomultipliers and times corresponding to signals, coming from both ends are measured. Average of times T_A and T_B measured from both ends defines T_0, their difference defines hit X-coordinate (coordinate along the scintillation counter).

5.1 TTR scintillation counters data format

Both times and amplitudes of the scintillation counter signals are read by TTR data aquisition system, so there are 2 different data groups in TTR event corresponding to scintillation counters.

TDC group : data part of TDC group consists of 64 16-bit words (1 word per channel). Each word contains time readings (bits 0-11) and CAMAC subaddress (bits 12-15). 12-bit TDC's (LRS 7186H) operate in the range 100ns (25ps/count) in COMMON STOP mode.

ADC group : for analog readout of scintillation counters data 12-bit ADC LRS 2249A are used. Data part of ADC group consists of 48 16-bit data words (1 word per channel). Each word in bits 0-11 contains ADC readings.

5.2 TTR scintillation counters constant file format

Scintillation counters constant file is a text file, read by FFREAD routine [4] during the initialization stage. The name of this file is defined in **tor.init** file. The following data cards and records are recognized.

1. Technology type 12 (TTR scintillators) should be defined in the beginning of the constant file by data card

TECHNOLOGY 12

2. Total number of scintillation counters processed by TOR is defined by card

NTOT n

where n (integer) is the number of counters.

3. Number of processed scintillation planes is defined by data card

NPLANES n

where n(integer) is the number of planes

4. Number of scintillation counters per plane (assumed to be the same for all planes) is defined by data card

NCOUNTERS n

where n(integer) is the number of counters per plane

5. CAMAC modules (ADC's and TDC's) used in readout are described by similar data blocks

starting with keyword *MODULE.

*MODULE name module_number nchannels

where name(text constant of $\leq$ 8 characters) stands for the type of CAMAC module defined. Possible values are:

'LRS2249A' : LeCroy ADC 2249A

'PH7186H' : Philips TDC 7186H

'LRS2277' : LeCroy TDC 2277

The *MODULE card should be followed by **nchannels** cards describing channel parameters. Each card should have the following format.

TDC case : subaddress slope intercept zero max_counts

subaddress (integer) : CAMAC subaddress of the given channel

slope (real) : TDC channel slope in ns/(TDC count)

intercept (real) : TDC channel intercept in ns

zero (real) : not used, but should be present (as a placeholder)

max_counts (integer) : maximal readings in this channel

ADC case : channel_number slope intercept max_counts

subaddress (integer) : CAMAC subaddress of the given channel;

slope (real) : TDC channel slope in ns/(TDC count);

intercept (real) : TDC channel intercept in ns;

max_counts (integer) : maximal readings in this channel

6. Scintillation counter is described by data block starting with keyword *SCIN. The data block has the following format.

```
*SCIN number plane first_run last_run x0 v0
      adcA tdcA delayA w1A w2A w3A
      adcB tdcB delayB w1B w2B w3B
      xsize ysize zsize X0 Y0 Z0
```

number (integer) : scintillation counter number

plane (integer) : number of scintillation plane corresponding to this counter

first_run (integer) : not used

last_run (integer) : not used

x0, v0 (real) - constants used for calculation of hit X-coordinate :
$X = x0+v0(T_A-T_B)$, T_A and T_B - times, measured from the both ends of the scintillation counter

adcA : A ADC channel (sequential number of word in ADC data group) for given scintillation counter

tdcA : A TDC channel for given scintillation counter

delayA : delay (in ns) in A time channel

w1A, w2A, w3A : walk correction constants for A channel (not used to the moment)

adcB, tdcB, delayB, w1B, w2B, w3B : the same for B channel of the scintillation counter

xsize : scintillator length

ysize : scintillator width

zsize : scintillator thickness

X0, Y0, Z0 : X, Y and Z coordinates of scintillator center

5.3 Examples

1. **TDC description**

```
*MODULE
c          TDC type number nchannels
           'PH7186H' 1      16
c          subaddr slope      intercept     zero     max_count
           1       .25037E-01 .63131E+01    .0       4095.0
           2       .25019E-01 .59157E+01    .0       4095.0
           3       .25016E-01 .55220E+01    .0       4095.0
           4       .25006E-01 .55355E+01    .0       4095.0
           5       .24990E-01 .56565E+01    .0       4095.0
           6       .24987E-01 .55054E+01    .0       4095.0
           7       .24989E-01 .51374E+01    .0       4095.0
           8       .25009E-01 .48957E+01    .0       4095.0
           9       .24997E-01 .54565E+01    .0       4095.0
           10      .25007E-01 .53875E+01    .0       4095.0
           11      .25023E-01 .51702E+01    .0       4095.0
           12      .25008E-01 .59390E+01    .0       4095.0
           13      .25022E-01 .56630E+01    .0       4095.0
           14      .25007E-01 .57460E+01    .0       4095.0
           15      .25004E-01 .54732E+01    .0       4095.0
           16      .25031E-01 .56249E+01    .0       4095.0
```

2. **ADC description**

```
*MODULE
c          ADC type number nchannels
           'LRS2249A'1     12
c          subaddr slope      intercept     max_count
           1       1.         0.            1023.
           2       1.         0.            1023.
           3       1.         0.            1023.
```

4	1.	0.	1023.
5	1.	0.	1023.
6	1.	0.	1023.
7	1.	0.	1023.
8	1.	0.	1023.
9	1.	0.	1023.
10	1.	0.	1023.
11	1.	0.	1023.
12	1.	0.	1023.

3. scintillation counter description

*SCINTILLATION COUNTER number 6 plane 2 first run : 1 last run : 0

c	v0	x0	adc	tdc	delay	walk correction constants
	15.484	0.83919	24	30	0.34	0.0 0.0 0.0
			12	14	0.46	0.0 0.0 0.0

c	xsize	ysize	zsize	X0	Y0	Z0
	270.0	20.0	2.0	-112.0	50.	508.0

5.4 Histogramming for scintillation counters

5.4.1 Reserved histogram ID's

Histogram ID's 120000-129999 are reserved for scintillation counters. To the moment the following histograms are booked and filled by TOR if requested.

ID = 120150 : total number of hit counters
120150+i : number of hit sc. counters in plane i
120155 : $(n3 + 10n4)vs(n1 + 10n2)$

Parameters of track reconstructed from scintillation counters are histogrammed in:

ID =
120163 : chi**2 (XZ)
120164 : rho**2 (XZ)
120173 : chi**2 (YZ)
120174 : rho**2 (YZ)
120181 : track theta
120182 : track phi

Hit profile histograms for different planes :

120200+i : hit profile for plane number i

5.4.2 Histogram sets booked for scintillation counters

The following histogram sets are booked for scintillation counters by user request.

'tdc' : ID = 121001-121064 : raw TDC spectra;

'adc' : ID = 122001-122048 : raw ADC spectra;

'time' : ID = 120071-120094 : T(trigger)-T(hit)
ID = 120301-120364 : times measured in 64 channels

'xhit' : ID = 120371-120394 : X-distributions for sc counters

5.4.3 *Histogramming parameters calculated for TTR scintillation counters*

The following parameters which can be stored into ntuples are calculated by TOR.

hp.sc(i)	: TDC counts in i-th channel
hp.sc(64+i)	: number of the 1st counter hit in plane i
hp.sc(69)	: number of the first counter hit
hp.sc(70)	: number of the last counter hit
hp.sc(70+i)	: T(plane)-T(trigger) for i-th counter
hp.sc(100+i)	: ADC counts in i-th channel
hp.sc(150)	: total number of hit scintillation counters
hp.sc(150+i)	: number of scintillation counters hit in i-th plane
hp.sc(160)	= 1 if sc_track was found
	= 0 if sc_track was NOT found

If there were 2 or more scintillation planes hit, track reconsruction is performed using scintillation counters hits. Track is fitted separately in XZ and YZ planes. The corresponding track parameters are :

hp.sc(161)	: A0 (x = A0*z+A1)
hp.sc(162)	: A1
hp.sc(163)	: chi**2 in XZ plane
hp.sc(164)	: rho**2 in XZ plane
hp.sc(171:174)	: the same as hp.sc(161-164) but for YZ plane
hp.sc(181)	: track polar angle (THETA)
hp.sc(182)	: track asimuthal angle (PHI)

When scintillation planes are moved to cover the maximal area there are usually 2 scintillation counters hit per event : one hit in top planes and one - in the bottom. But in general sc_track may have up to 4 hits. The following parameters of the 1st and the last hits are calculated and stored for histogramming.

hp.sc(191) : first hit x-coordinate
hp.sc(192) : first hit y-coordinate
hp.sc(193) : first hit z-coordinate
hp.sc(194) : last hit x-coordinate
hp.sc(195) : last hit y-coordinate
hp.sc(196) : last hit z-coordinate

Some additional parameters :

hp.sc(300+i) : time measured in i-th channel
hp.sc(370+i) : X(hit) for i-th counter

6 TTR Iarocci tubes

There are 4 planes of Iarocci tubes at TTR. Two of them are placed above the magnet, the other two - below the magnet. The main purpose of TTR Iarocci tubes is to provide momentum cut for incident muons. Currently this cut is not used in data analysis, but 2 top planes are used for reconstruction of track, external with respect to muon chambers tested.

6.1 TTR Iarocci tubes data format

The TTR data aquisition system reads information from 4 planes of Iarocci tubes. The information read out from each plane contains data on clusters and 4 2-byte words of plane test pattern. Cluster information is packed into one (usually) or several (for clusters with number of strips hit more than 7) 16-bit words in the following way:

bits 0-2	: cluster size (number of hit strips in the cluster). If cluster size exceeds 7, then cluster information is stored in several words.
bits 3-5	: readout chain number (chain corresponds to one plane)
bits 7-16	: number of the last strip in the cluster

6.2 The format of TTR Iarocci constant file

Text file containing description of TTR iarocci tube system is read by TOR at the initialization stage using the FFREAD [4] package. It's name is defined in **tor.init** file. The following data cards and records are recognized.

1. technology type 13 (TTR iarocci tubes) should be defined in the beginning of the constant file by data card

TECHNOLOGY 13

2. Each plane of Iarocci tubes is described by data block starting with keyword *IAROCCI which has the following format.

*IAROCCI
first_run last_run plane chain nstrips nxstrips nystrips pitch xc yc zc xsize ysize

where

first_run (integer) : not used

last_run (integer) : not used

plane (integer) : number of Iarocci plane. The planes are numbered from the top, so the most top one is plane number 1.

chain (integer) : readout chain number corresponding to given plane

nstrips (integer) : total number of strips in this plane

nxstrips (integer) : number of X-strips (strips parallel to X axis)

nystrips (integer) : number of Y-strips (strips parallel to Y axis)

pitch (real) : strip pitch (assumed to be the same for X- and Y-strips)

xc, yc, zc (real) : X-, Y-, Z-coordinates of the plane center

xsize, ysize (real) : X- and Y-size of the plane

6.3 Examples

1. **Iarocci plane description**

*IAROCCI plane description

c	1st	last	pln	chain	nstrips	nxstrips	strip	XC	YC	ZC	xsize	ysize
1	0	1	0	608	192		1.0	-0.4	0.0	455.0	416.	192.

6.4 Histogramming for Iarocci's

6.4.1 Reserved histogram ID's

Histogram ID's reserved for Iarocci's are 130000-139999. To the moment the following histograms are booked and filled by TOR if requested.

ID = 130001 : total number of clusters in all 4 planes
130002 : number of planes hit
130003 : (N_3+10*N_4) vs (N_1+10*N_2), where N_i - number of clusters in i-th plane

Histograms filled for each plane separately :

ID = 130000+100*ipl+1 : nyclust vs nxclust for plane ipl
130000+100*ipl+2 : nclusters for plane ipl
130000+100*ipl+3 : hit profile in plane ipl
130000+100*ipl+4 : X-cluster size in plane ipl
130000+100*ipl+5 : Y-cluster size in plane ipl
130000+100*ipl+6 : Y-residuals in plane ipl (from **sc_track**)
130000+100*ipl+7 : X-residuals in plane ipl (from **sc_track**)

6.4.2 Histogram sets booked for Iarocci tubes

There are no histogram sets booked for TTR Iarocci tubes to the moment.

6.4.3 Parameters calculated for TTR Iarocci tubes

The following Iarocci's parameters are calculated for each event processed by TOR.

hp.iar(1)	: total number of clusters in all planes
hp.iar(2)	: number of planes hit (with one or more cluster)
hp.iar(3)	: N_1+10N_2
hp.iar(4)	: N_3+10N_4
hp.iar(11-20)	: parameters of iarocci track in XZ plane (not implemented yet)
hp.iar(21-30)	: parameters of iarocci track in YZ plane
hp.iar(21)	: Y-slope of iarocci track
hp.iar(22)	: Y-intercept of iarocci track

hp.iar(100*ipl+ 1) : number of X clusters in plane **ipl**
hp.iar(100*ipl+ 2) : number of Y clusters in plane **ipl**
hp.iar(100*ipl+ 3) : total number of clusters in plane **ipl**
hp.iar(100*ipl+11) : X-coordinate of Y-cluster closest to **sc_track**
hp.iar(100*ipl+12) : Y-coordinate of closest X-cluster
hp.iar(100*ipl+13) : X-residual from **sc_track** of closest Y-cluster
hp.iar(100*ipl+14) : Y-residual from **sc_track** of closest X-cluster

7 TOR processing of pressurised drift tubes

TOR provides processing of both drift technolodies tested on TTR : drift chambers built in Dubna and that built in Michigan State University. For the historical reasons there are 2 names used for pressurised drift tubes : PDT (Pressurised Drift Tubes) and RDT (Round Drift Tubes) abbreviation PDTR is used for drift chamber built in Dubna (R stands for Russian) and abbreviation PDTU is used for drift chamber built in Michigan (U stands for US). In this section we describe processing of PDTR data, dealing with PDTU data is discussed in section 9.

7.1 PDTR data format

For data readout from PDTR chamber Philips TDC's 7186H operating in COMMON START mode are used. Data part of PDTR group consists of 128 words (1 word/channel), each of them contains time readings (bits 0-11) and CAMAC subaddress (bits 12-15). First 64 channels correspond to chamber number 1, last 64 - to chamber number 2.

7.2 PDTR constant file format

PDTR constant file is a text file, read by FFREAD routine [4] during the initialization stage. The name of this file is defined in **tor.init** file. The following data cards and records are recognized.

1. Technology type 3 (PDTR) should be defined in the beginning of the constant file by data card

TECHNOLOGY 3

2. Number of chambers to be processed and order of processing is define by data card

*NCHAMBERS nchambers n1 n2 ...

where

nchambers : total number of chambers to be processed

n1 n2 ... : sequential numbers of chambers

3. Data card

*CHAMBER_SCALE 64*1 64*2

defines that first 64 readout channels correspond to chamber number 1, and channels 65-128 correspond to chamber number 2. In general i-th input number is the number of chamber for readout channel number i.

4. Data card

*LAYER_SCALE 16*4 16*3 16*2 16*1 16*1 16*2 16*3 16*4

sets correspondence between readout channels and layer numbers. There are 4 layers per PDTR chamber and i-th input number is the number of layer (within the chamber) corresponding to i-th PDTR readout channel.

5. Data block

*TUBE_SCALE

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

sets the correspondence between tube numbers and readout channels. There are 16 tubes per layer in one chamber and i-th input number is number of the tube (within the layer) corresponding to i-th PDTR readout chnnel number.

6. Data card

*MODSCALE 16*1 16*2 16*3 16*4 16*5 16*6 16*7 16*8

sets the correspondence between CAMAC TDC modules and readout channels. I-th input number is TDC module number for readout channel number i.

6. Data block

*CHSCALE

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

sets the correspondence between readout channels and corresponding to them subaddresses within TDC modules. I-th input number is CAMAC subaddress corresponding to readout channel number i.

7. The drift chamber geometry is described by the following data block, starting with the keyword *GEOMETRY.

```
*GEOMETRY    ch_number    x0    y0    z0
             theta    psi    phi
             nlayers
```

nt_1 ... $nt_{nlayers}$
dy_1 ... $dy_{nlayers}$
dz_1 ... $dz_{nlayers}$
step
wlength
radius

where

ch_number (integer) : PDTR chamber number (1 or 2)

x0, y0, z0 (real) : coordinates of the chamber reference point in TTR system

theta, phi, psi (real) : chamber rotation angles in TTR coordinate system

nlayers (integer) : number of drift tube layers

nt_1 ... $nt_{nlayers}$ (integer) : **nlayers** numbers i-th of which is the number of drift tubes in i-th layer

dy_1 ... $dy_{nlayers}$ (real) : i-th from these **nlayers** numbers stands for Y-displacement of the 1-st wire in i-th layer with respect to chamber reference point

dz_1 ... $dz_{nlayers}$ (real) : i-th from these **nlayers** numbers stands for Z-displacement of the 1-st wire in i-th layer with respect to chamber reference point.

step (real) : Y step between wires in one layer (assumed to be the same for all the layers within the drift chamber).

wlength (real) : wire length

radius (real) : drift tube inner radius

8. The TDC module used for PDTR readout are described by *MODULE data block (see sect. 4.2)

7.3 Histogramming for PDTR

7.3.1 Histogram ID's reserved for PDTR

Histogram ID's 30000-39999 are reserved for PDTR. To the moment the following histograms are booked and filled by TOR if requested.

ID = 31130+i : sum of drift time histograms for plane **ipl**
31140 : sum of all drift time histograms
31141 : sum of drift time histograms for all tubes of PDTR chamber 1
31142 : sum of drift time histograms for all tubes of PDTR chamber 2
33000 : total number of hits in both PDTR chambers
33001 : number of hits in PDTR chamber number 1
33002 : number of hits in PDTR chamber number 2
33010+i : number of hits in plane i (assuming that planes 1-4 belongs to first chamber, planes 5-8 - to the second.

hspace0.2in Because there are 2 PDTR chambers, track found in PDTR may have 2 segments. Accordingly, we may speak about parameters of PDTR track **s0**, defined by both chambers together, and about parameters of each segment **s1 or s2**.

34001 : s0 slope
34002 : s0 intercept
34003 : error in s0 slope
34004 : error in s0 intercept
34005 : s0 chi^2
34006 : s0 rho^2
34007 : s0 nhits
34008 : number of iterations, required for fit

34101-34108 : the same for s1
34201-34208 : the same for s2

7.3.2 Histogram sets booked for PDTR

The following histogram sets are booked for scintillation counters by user request.

'tdc' : ID = 30001-30128 : raw TDC spectra measured from PDTR tubes.
ID = 30000+i corresponds to i-th readout channel;

'time' : ID = 31001-31128 : time spectra measured in PDTR tubes
ID = 30000+i corresponds to i-th readout channel;

7.3.3 Histogramming parameters calculated for PDTR

Histogramming parameters calculated for PDTR have 4-character symbolic names **'pxxx'**, where xxx stands for parameter index in **hp.pdtr** array

hp.pdtr(i) : TDC counts in i-th channel
hp.pdtr(130+i) : drift time in i-th channel
hp.pdtr(260+i) : drift distance in the tube corresponding to i-th channel
hp.pdtr(400) : total number of hits in PDTR chambers
hp.pdtr(401) : number of hits in PDTR chamber number 1
hp.pdtr(402) : number of hits in PDTR chamber number 2
hp.pdtr(403) :
hp.pdtr(404) :
hp.pdtr(410+i) : number of hits in i-th PDTR layer
hp.pdtr(419) : max number of hits per layer found in PDTR chambers
hp.pdtr(420) : PDTR track flag
hp.pdtr(420+i) : number of track hits in i-th PDTR layer

Parameters of the track **s0** (defined by both chambers) are :

hp.pdtr(431) : s0 slope
hp.pdtr(432) : s0 intercept
hp.pdtr(433) : error in s0 slope
hp.pdtr(434) : error in s0 intercept
hp.pdtr(435) : chi^2 of s0 fit
hp.pdtr(436) : rho^2 of s0 fit
hp.pdtr(437) : number of s0 hits
hp.pdtr(438) : number of s0 fit iterations

hp.pdtr(441:448) : the same as hp.pdtr(431:438), but for s1 - track segment reconstructed in PDTR chamber number 1.
hp.pdtr(451:458) : the same as hp.pdtr(431:438), but for s2 - track segment reconstructed in PDTR chamber number 2.

After the track was found the following parameters of track hits are defined :

hp.pdtr(460+i) : drift time for i-th track hit
hp.pdtr(490+i) : drift distance corresponding to i-th track hit
hp.pdtr(520+i) = 5. if Y(wire) > Y(track)
= -5. if Y(wire) < Y(track)
hp.pdtr(550+i) : delta(R) = $R_{hit} - R_{track}$, where R_{track} is the distance from the track to the wire
hp.pdtr(580+i) : delta(Y) = $Y_{hit} - Y_{track}$ at $X = X_{hit}$
hp.pdtr(610+i) : plane number for i-th hit
hp.pdtr(640+i) : tube number for i-th hit
hp.pdtr(670+i) : distance from the track to the wire for i-th hit
hp.pdtr(700+i) : X-coordinate of i-th hit
hp.pdtr(730+i) : Y-coordinate of i-th hit
hp.pdtr(760+i) : Z-coordinate of i-th hit
hp.pdtr(790+i) : delta(Z) = $Z_{hit} - Z_{track}$ at $X = X_{hit}$

8 Processing of Michigan pressurized drift tube chamber

Abbreviation PDTU (Pressurised Drift Tube USA) is used for drift chamber built in Michigan State University.

8.1 PDTU data format

For data readout from PDTU chamber multihit LeCroy TDC's 2277 operating in COMMON STOP mode are used. Data part of PDTU group contains 2 16-bit words per hit. The format is as follows.

1-st word :	bit0	: leading edge :	1 = COMMON STOP mode
			0 = COMMON START mode
	bits 1-5	: TDC channel (0-31)	
	bits 6-15	: CAMAC slot (1-23)	
2-nd word :	bits 0-15	: hit time readings	

To the moment only the first hit in each channel is used, but number of hits in each channel is calculated by TOR.

8.2 PDTU constant file format

The format of PDTU constant file is the same as the format of PDTR constant file with the only difference in technology code (PDTU technology code = 4) and in the number of channels in TDC module description (TDC's used for Michigan drift chamber readout are different from that used for Dubna drift chambers readout)

8.3 Histogramming for PDTU

8.3.1 Histogram ID's reserved for PDTU

Histogram ID's from 40000 to 49999 are reserved for PDTU histograms. PDTU histogram ID = 40000+i is exactly the same as PDTR histogram ID = 30000+i (but there is only one PDTU chamber!)

8.3.2 Histogram sets booked for PDTU

The histogram sets booked for PDTU are exactly the same as that booked for PDTR.

'tdc' : ID = 40001-40128 : raw TDC spectra measured from PDTU tubes.
ID = 40000+i corresponds to i-th readout channel;

'time' : ID = 41001-41128 : time spectra measured in PDTU tubes
ID = 40000+i corresponds to i-th readout channel;

8.3.3 Histogramming parameters calculated for PDTU

Histogramming parameters calculated for PDTU have 4-character symbolic names **'uxxx'**, where xxx stands for parameter index in **hp.pdtu** array. The meaning of parameters is exactly the same as for corresponding PDTR parameters (see sect. 6.3.3.)

9 TOR processing of cathode strip chambers (CSC)

To the moment data only from the cathode strip chamber built in Dubna are processed by TOR. The abbreviation CSCR (Cathode Strip Chamber Russian) will be used for this chamber below.

9.1 CSCR data format

The data part of CSCR group contains 1 word per readout channel (currently 96 channels are read). Each of data words contains in bits 0-10 charge read out from the corresponding strip. Bit 11 = 1 means channel overflow. In this case data read out from this channel are meaningless.

9.2 CSCR constant file format

CSCR constant file is a text file, read at the initialization stage. The name of this file is defined in **tor.init** file. The following data cards and records are recognized.

1. Technology type 8 (CSCR) should be defined in the beginning of the constant file by data card

TECHNOLOGY 8

2. Number of chambers to be processed and order of processing is define by data card

*NCHAMBERS nchambers n1 n2 ...

where

nchambers : total number of chambers to be processed

n1 n2 ... : sequential numbers of chambers. Currently there is only one CSCR chamber present.

3. Number of readout channels is defined by card

*CHANNELS nchannels

where **nchannels** (integer) is the total number of CSCR readout channels to be processed by TOR.

4. Data card

*CHAMBER_SCALE 96*1

defines that first 96 readout channels correspond to chamber number 1. In general i-th input number is the number of chamber for readout channel number i.

5. Data card

*LAYER_SCALE 64*2 32*1

defines that first 64 readout channels correspond to the second layer within the chamber, the rest 32 readout channels - to the first layer. In general i-th input number is the layer number, corresponding to i-th readout channel.

6. Data block

*STRIP_SCALE
c 1st layer

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64

c 2nd layer

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

sets correspondence between readout channels and strip numbers (within the layer). I-th number read is the strip number, corresponding to readout channel number i.

7. CSCR geometry is described by GEOMETRY data block having the following format (very similar to GEOMETRY block format for drift chambers) .

*GEOMETRY ch_number x0 y0 z0
nplanes
$theta_1$ ... $theta_{nplanes}$
psi phi
$nstrips_1$ ... $nstrips_{nplanes}$
dy_1 ... $dy_{nplanes}$
dz_1 ... $dz_{nplanes}$
pitch
strip_width

where

ch_number (integer) : CSCR chamber number (currently - 1)

x0, y0, z0 (real) : coordinates of the chamber reference point in TTR system

nplanes (integer) : number of planes in strip chamber

theta$_1$ ... **theta**$_{nplanes}$ (integer) : rotation angles around TTR Z axis (different strip boards are not assumed to be parallel)

phi, psi (real) : 2 other chamber rotation angles (not used to the moment)

nstrips$_1$ ... **nstrips**$_{nplanes}$ (integer) : number of strips in different planes in i-th layer

dy$_1$ ... **dy**$_{nplanes}$ (real) : i-th from these **nplanes** numbers stands for Y-displacement of the 1-st strip in i-th plane with respect to chamber reference point

dz$_1$ ... **dz**$_{nplanes}$ (real) : i-th from these **nplanes** numbers stands for Z-displacement of the 1-st strip in i-th plane with respect to chamber reference point.

pitch (real) : readout pitch (assumed to be the same for all the layers within the strip chamber).

strip_width (real) : real strip width (not used to the moment)

7. Pedestal values in readout channels are defined by PEDESTAL data block which has the format as follows.

*PEDESTALS PED_1 ... PED_n

where PED_1 ... PED_n (real) are pedestal positions (i-th number on input corresponds to i-th readout channel)

8. Pedestal widths in readout channels are defined by PWIDTHS data block which has the format as follows.

*PWIDTHS W_1 ... W_n

where W_1 ... W_n (real) are pedestal peak widths (i-th number on input corresponds to i-th readout channel)

9. Calibration curves for CSCR channels are currently parametrized with 3-rd order polynomials. Coefficients of polinomials are defined by SLOPES data block. The format of data block is as follows.

*SLOPES anorm

$c0_1$ $c1_1$ $c2_1$ $c3_1$
$c0_2$ $c1_2$ $c2_2$ $c3_2$
.....
$c0_n$ $c1_n$ $c2_n$ $c3_n$

where

anorm is common normalizing coefficient by which all the polynomial coefficients are divided

c0$_i$, **c1**$_i$, **c2**$_i$, **c3**$_i$ are coefficients of 3-rd order polynomial, defining calibration curve in i-th readout channel (these coefficients are divided then by **anorm**.

9.3 Histogramming for CSCR

9.3.1 Histogram ID's reserved for CSCR

Histogram ID's 80000-89999 are reserved for CSCR. To the moment the following histograms are booked and filled by TOR if requested.

9.3.2 Histogram sets booked for CSCR

The following histogram sets are booked for CSCR by user request.

'adc' : ID = 81001-81096 : raw amplitude spectra from CSCR channels.
ID = 81000+i corresponds to i-th readout channel;

'char' : ID = 82001-82096 : charge spectra (amplitudes corrected according to calibration procedure) in CSCR channelss
ID = 82000+i corresponds to i-th readout channel;

'nois' : ID = 880001-88096 : correlated noise spectra in CSCR channels.
ID = 88000+i corresponds to i-th readout channel;

'corr' : ID = 890001-89096 : amplitude spectra, corrected to correlated noise in CSCR channels.
ID = 89000+i corresponds to i-th readout channel;

9.3.3 Histogramming parameters calculated for CSCR

Histogramming parameters calculated for CSCR have 4-character symbolic names **'cxxx'**, where xxx stands for parameter index in **hp.cscr** array

hp.cscr(001-200)	: ADC counts for CSCR channels
hp.cscr(201-400)	: charges induced on CSCR strips
hp.cscr(401)	: total number of strips hit (in both planes)
hp.cscr(402)	: total number of clusters found (in both planes)
hp.cscr(410+plane)	: number of strips hit in given plane
hp.cscr(430+plane)	: number of clusters in given plane
hp.cscr(440+plane)	: size of the 1st cluster in the given plane
hp.cscr(450+plane)	: charge of the 1st cluster in the given plane
hp.cscr(460+plane)	: common pedestal shift in the given plane
hp.cscr(470+plane)	: Y-residual of the closest cluster
hp.cscr(490+plane)	: Y-coordinate of the closest cluster
hp.cscr(500+plane)	: X-coordinate of the closest cluster
hp.cscr(510+plane)	: Y-coordinate of RDT track in the given plane as reconstructed from gaussian fit to 3 most significant strips (really it is exact solution for the case of equal errors)
hp.cscr(520+plane)	: X-coordinate of **sc_track** in the given plane
hp.cscr(540+plane)	: r1 = q1/qmax for the cluster closest to RDT track
hp.cscr(550+plane)	: r2 = q2/qmax for the cluster closest to RDT track
hp.cscr(560+plane)	: delta = Y(cluster)-Y(PDTR+CSCR) track for the cluster coordinate defined by by gaussian solution
hp.cscr(570+plane)	: central strip number for the cluster closest to PDTR track
hp.cscr(580+plane)	: central strip charge of the cluster closest to PDTR track
hp.cscr(590+plane)	: sigma of the cluster closest to RDT track as given by gaussian solution

hp.cscr(600+plane) : alpha = arctg((1-r1)/(1-r2)) for the cluster closest to RDT track
hp.cscr(610+plane) : deltav = Y(cluster)-Y(central strip) for cluster Y-coordinate reconstructed

by ratio method
hp.cscr(620+plane) : yv - cluster coordinate reconstructed by ratio method
hp.cscr(630+plane) : dyv - Y-residual from the (PDTR+CSCR) track of the cluster reconstructed by ratio method
hp.cscr(800+i) : reduced charge in i-th channel
hp.cscr(900+i) : amplitude in i-th channel, corrected to correlated noise

10 Creating personal version of TOR

TOR is installed on HP cluster of SSCL PDSF computing facility and executable file named **tor.exe** can be found in the directory **~murat/tor/common_use**. Nevertheless, if user wants to perform some extra actions he can create his personal version of TOR. There are 3 TOR user routines : **tor_user_init**, **tor_user_event** and **tor_user_term** (dummy by default) intended to be writen by the user for this purpose.

tor_user_init is called once in the very end of initialization stage. It serves for nonstandard initialization of program constants and variables, including constants and variables defined by the user. Call format :

call tor_user_init (without parameters)

tor_user_event is called in the end of processing of each event, but before the histogram filling. In this routine user may perform additional calculations necessary for him (such as calculation of additional parameters to be put into HBOOK ntuples). Call format :

call tor_user_event (without parameters,

event parameters are available via TOR common blocks.

tor_user_term is called once in the end of run and allows user to perform necessary to him termination actions (such as closing files etc.) Call format :

call tor_user_term (without parameters)

To create personal version of TOR user should write necessary to him versions of TOR user routines, to compile them and to link them with TOR library, which can be found in **~/murat/lib/common_use** directory under the name **libtor.a**. Global TOR variables and constants are described in TOR include files (.inc files in **~/murat/tor/common_use** directory)

11 References

[1] K.McFarlane, F.Stocker, G.Yost. TTR goals and schedule. GEM TN-92-80.

[2] R.Brun et al. PAW. The Complete Reference (version 1.14), Program Library Q121. CERN, 1992.

[3] R.Brun, M.Goossens and J.Zoll. ZEBRA Users Guide, Program Library Q100. CERN, 1991.

[4] R.Brun et al. FFREAD USER GUIDE

[5] R.Brun et al. HBOOK users guide (version 4.15), Program Library Y250. CERN, 1992

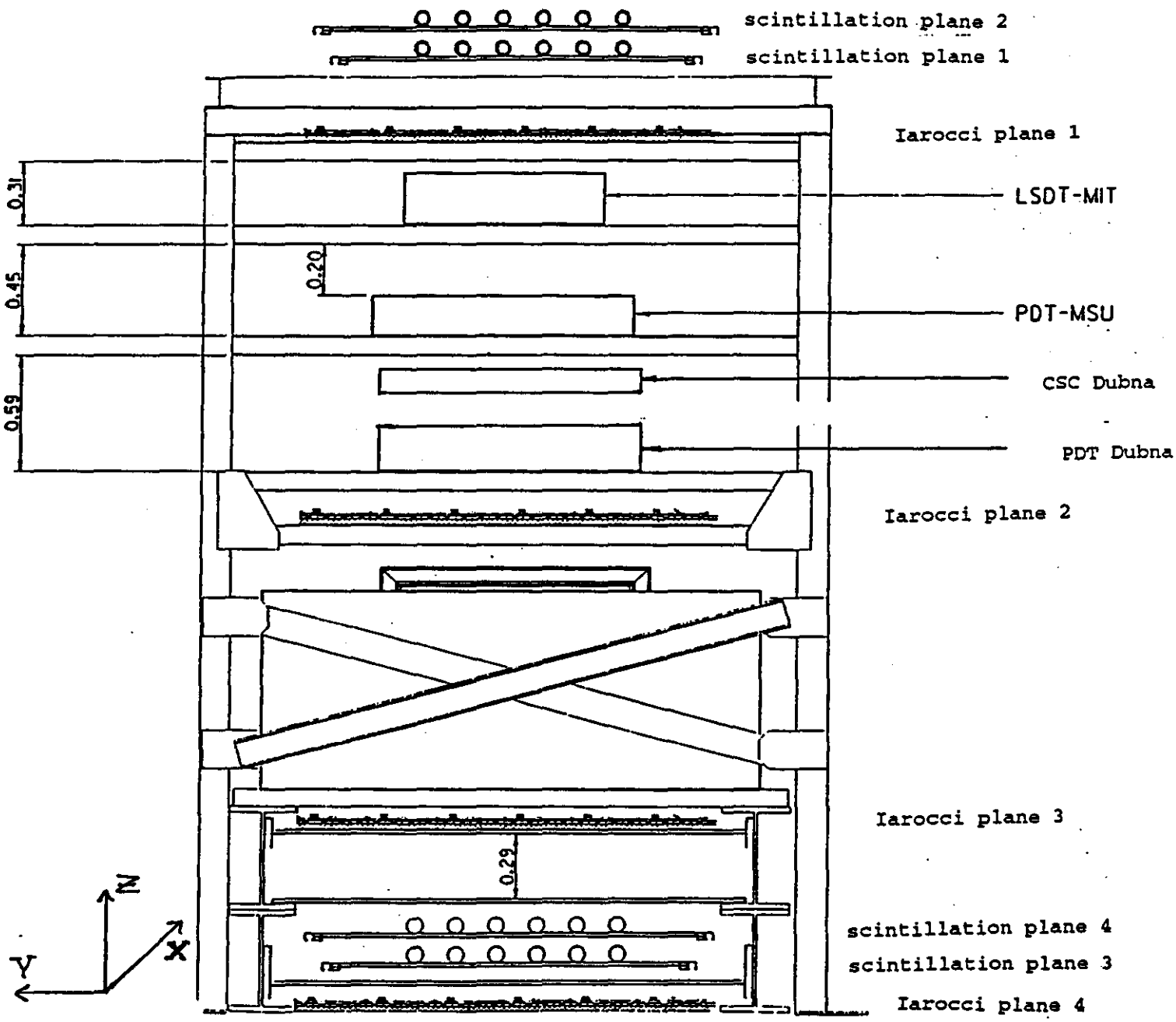

Fig.1 Texas Test Rig schematic layout