9 mars 1995

Le système de déclenchement du détecteur ALR+ pour l'expérience L3

DOCUMENT ANNEXE AU TRAVAIL DE DIPLOME

SCIARRINO Déborah

TRAVAIL DE DIPLOME

SOUS LA DIRECTION DE PROFESSEUR M. BOURQUIN

UNIVERSITE DE GENEVE

Table des matières

- **ANNEXE 1** : Documentation des photodiodes
- **ANNEXE 2** : Comportement dans le temps des 9 modules
- **ANNEXE 3** : Programme 'Leadring.car'
- **ANNEXE 4** : Programme 'Leadring.ffread'
- **ANNEXE 5** : Programme 'Install-lead'
- **ANNEXE 6** : Programme 't3-alr.for'
- **ANNEXE 7** : Tableau de correspondance des dB avec le facteur d'atténuation
- **ANNEXE 8** : Fichiers des données 5, 20, 30, 45, 60 et 90 GeV
- **ANNEXE 9** : Carte de sommation de deux canaux pour les tests
- **ANNEXE 10** : Comparaison des décibels mesurés (amplitude signal) avec la théorie
- **ANNEXE 11** : Programmes : 'initialisation.for', 'init-talr.pas', 'rout.for', 'matrix.dat'

ANNEXE 1

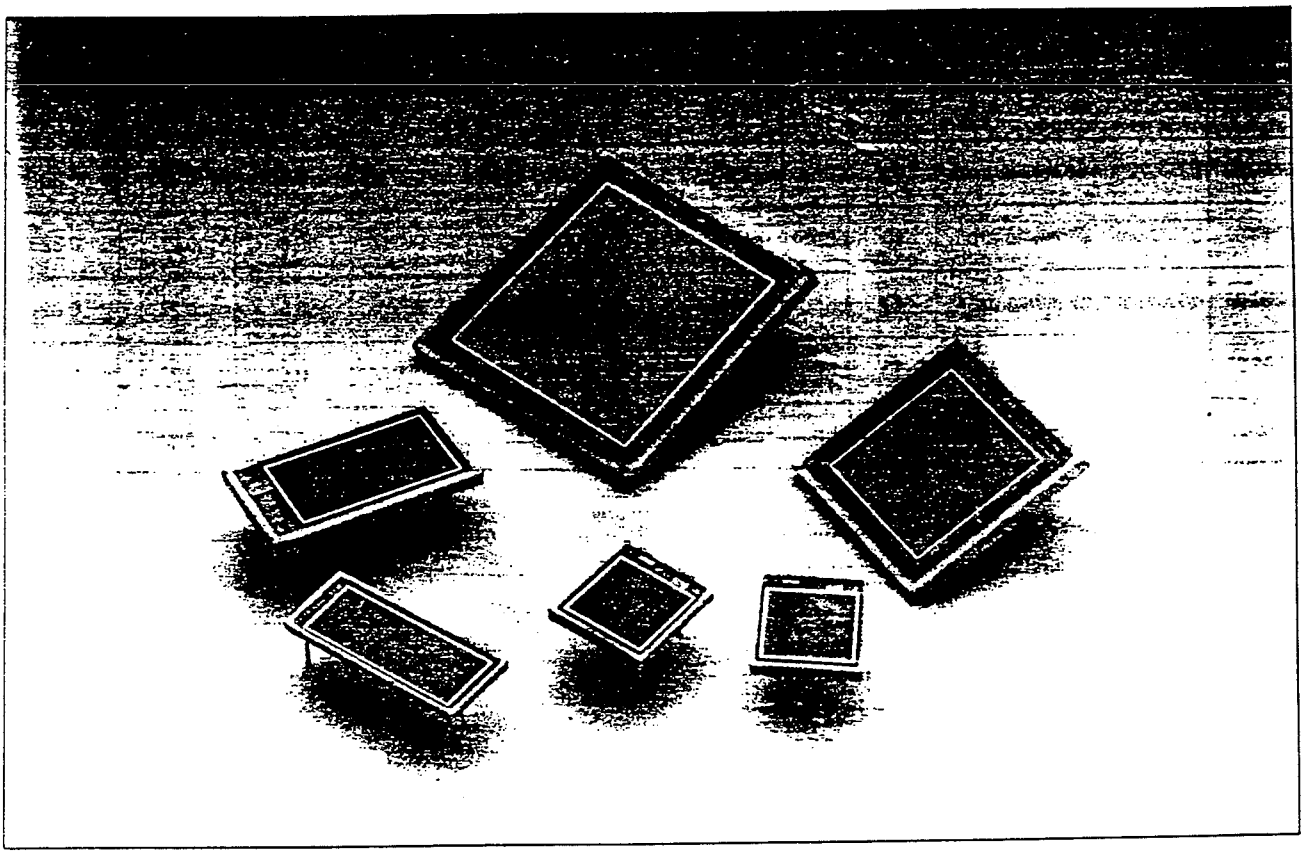
Documentation des photodiodes

HAMAMATSU

TECHNICAL DATA No.S-505-01

new

LARGE-AREA PIN SILICON PHOTODIODES FOR HIGH ENERGY PHYSICS AND NUCLEAR PHYSICS



Silicon photodiodes are promising devices for scintillation detectors in high energy physics experiments and nuclear physics. They are very compact, have low power demand, high quantum efficiency and are insensitive to magnetic fields.

Hamamatsu offers a variety of PIN silicon photodiodes specifically designed for scintillation detectors in high energy physics experiments and nuclear physics. These photodiodes have large, uniform sensitive-areas of square or rectangular shapes. In spite of large sensitive areas, they exhibit low dark current, low junction capacitance and low series resistance, assuring high-speed response, low noise and long-term high stability. Package configurations are well designed for good optical coupling to scintillators such as BGO and CsI crystals. Windowless types are also provided for direct high energy particle detection.

Standard types are made of a high-resistivity silicon wafer with 300 μm thickness that provides low junction capacitance at a low bias voltage. Also, smaller area types are available with a 200 μm thick wafer that enables low capacitance at even lower bias operation. 500 μm thick wafer is also used for 10 x 10 mm and larger area types that can be operated at a higher bias voltage, and are suited for direct particle detection.

FEATURES

- **Large sensitive area: 10 x 10mm, 10 x 20mm, 30 x 30mm, etc.**
- **High quantum efficiency**
- **Low dark current**
- **Low bias voltage operation**
- **Sensitivity matching with BGO scintillators**
- **High-speed time response**
- **Low noise and high stability**
- **Good energy resolution**

APPLICATIONS

- **Scintillation detectors**
- **Calorimeters**
- **Hodoscopes**
- **TOF counters**
- **Air shower counters**

etc.

Characteristics (at 25°C)							Maximum Ratings				Type No.
Dark Current I_d		Bias Reverse Voltage (Full Depletion Voltage)	Junction Capacitance C_j f=1MHz	Ⓓ Rise Time t_r	Cutoff Frequency f_c	Ⓔ NEP	Reverse Voltage V_R max.	Current I max.	Temperature Range		
Typ. (nA)	Max. (nA)	(V)	Typ. (pF)	Typ. (ns)	Typ. (MHz)	Typ. (W/Hz$^{1/2}$)	(V)	(mAp)	Operating (°C)	Storage (°C)	
2	10	70	50	10	45	2×10^{-13}	100	2	−20 ~ +60	−20 ~ +80	S3590-0
3	10	70	75	15	30	2×10^{-13}	100	2	−20 ~ +60	−20 ~ +80	S2662-0
4	15	70	100	20	25	4×10^{-13}	100	2	−20 ~ +60	−20 ~ +80	S2744-0
6	20	70	140	25	15	4×10^{-13}	100	2	−20 ~ +60	−20 ~ +80	S3204-0
10	30	70	330	70	5	5×10^{-13}	100	2	−20 ~ +60	−20 ~ +80	S3584-0
4	15	70	55	10	40	2×10^{-13}	100	2	−20 ~ +60	−20 ~ +80	S3588-0

Figure 1: Spectral Response

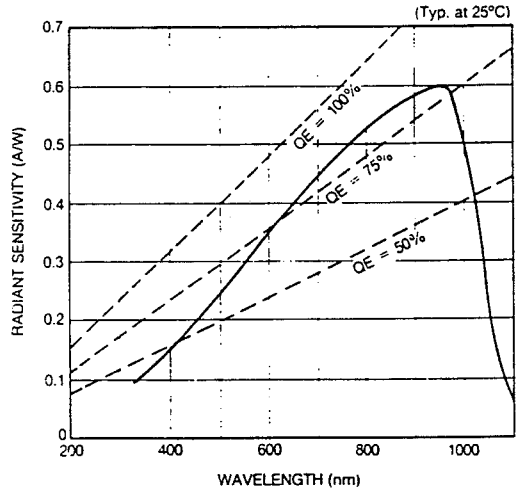


Figure 2: Junction Capacitance vs. Reverse Voltage

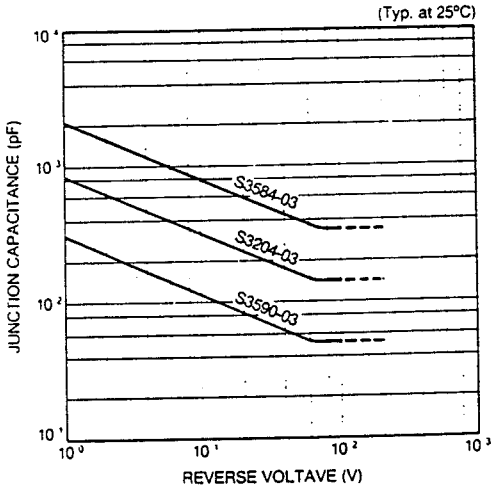


Figure 3: Dark Current vs. Reverse Voltage

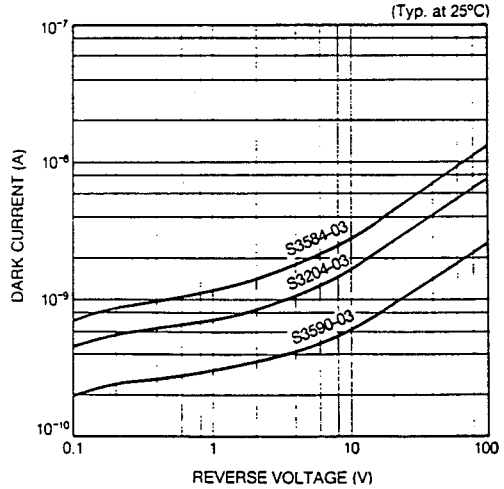
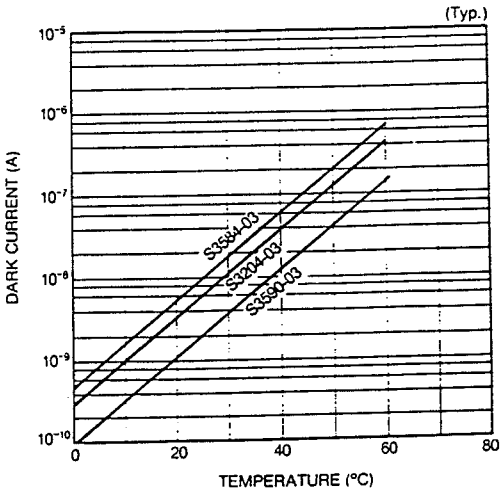


Figure 4: Dark Current vs. Temperature



Characteristics (at 25°C)							Maximum Ratings				Type No.
Dark Current I_d		Bias Reverse Voltage (Full Depletion Voltage)	Junction Capacitance C_j f=1MHz	Ⓓ Rise Time t_r	Cutoff Frequency f_c	Ⓔ NEP	Reverse Voltage V_R max.	Current I max.	Temperature Range		
Typ. (nA)	Max. (nA)	(V)	Typ. (pF)	Typ. (ns)	Typ. (MHz)	Typ. (W/Hz$^{1/2}$)	(V)	(mAp)	Operating (°C)	Storage (°C)	
1	5	30	70	15	30	2×10^{-13}	50	2	−20 ~ +60	−20 ~ +80	**S3590-01**
2	10	12	120	30	20	2×10^{-13}	50	2	−20 ~ +60	−20 ~ +80	**S2662-01**
1.5	5	30	80	15	30	2×10^{-13}	50	2	−20 ~ +60	−20 ~ +80	**S3588-01**
8	30	100	30	7	60	5×10^{-13}	150	2	−20 ~ +60	−20 ~ +80	**S3590-05**
15	50	100	80	15	30	5×10^{-13}	150	2	−20 ~ +60	−20 ~ +80	**S3204-05**
30	100	100	200	40	10	1×10^{-12}	150	2	−20 ~ +60	−20 ~ +80	**S3584-05**

Figure 5: Spectral Response

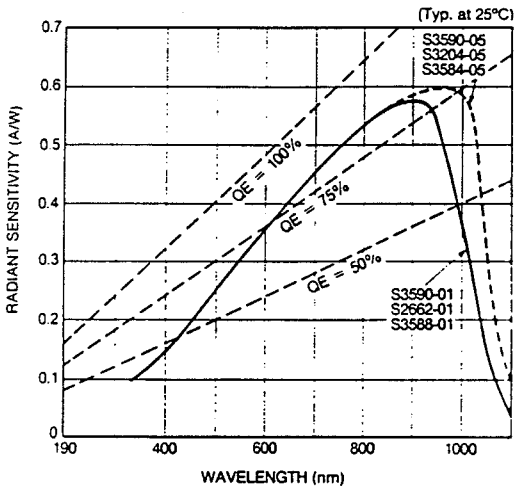


Figure 6: Junction Capacitance vs. Reverse Voltage

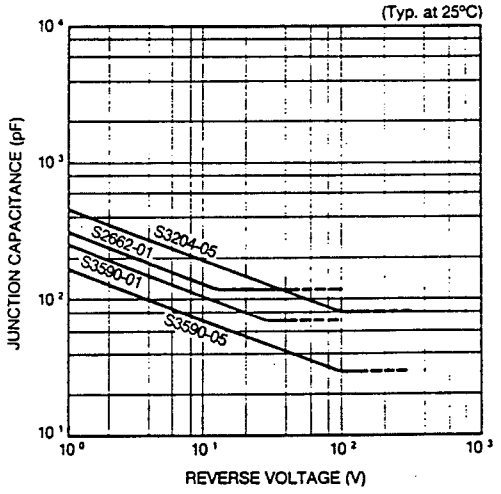


Figure 7: Dark Current vs. Reverse Voltage

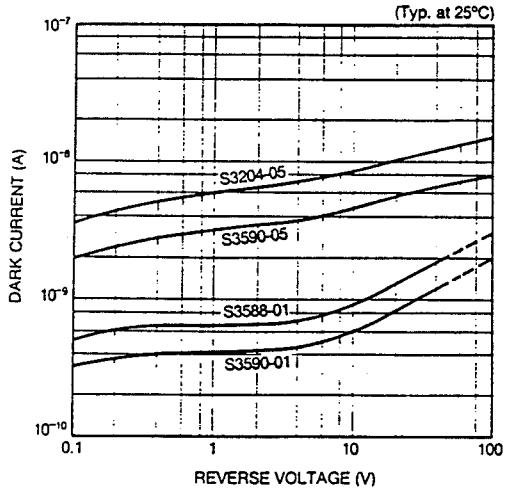
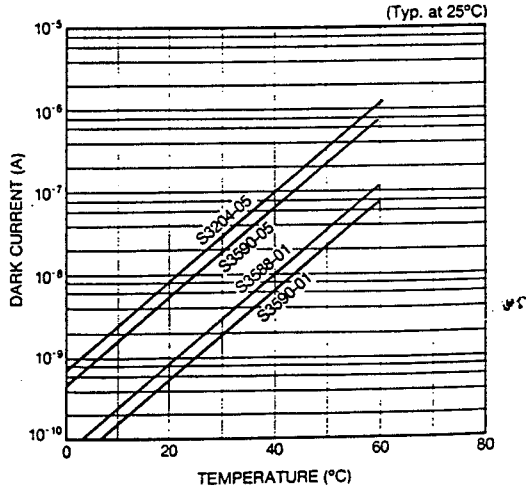
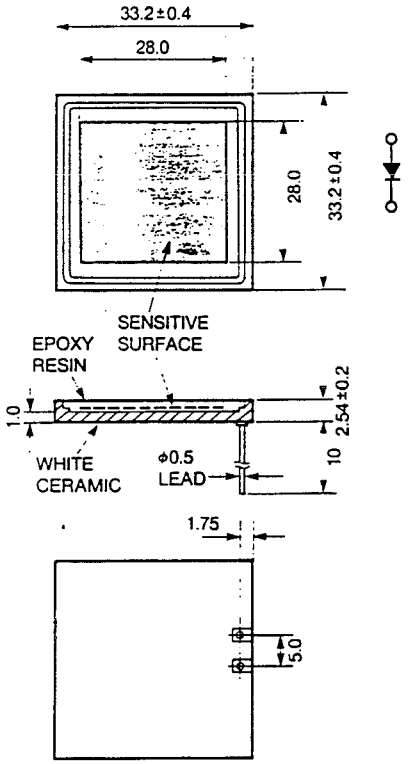


Figure 8: Dark Current vs. Temperature

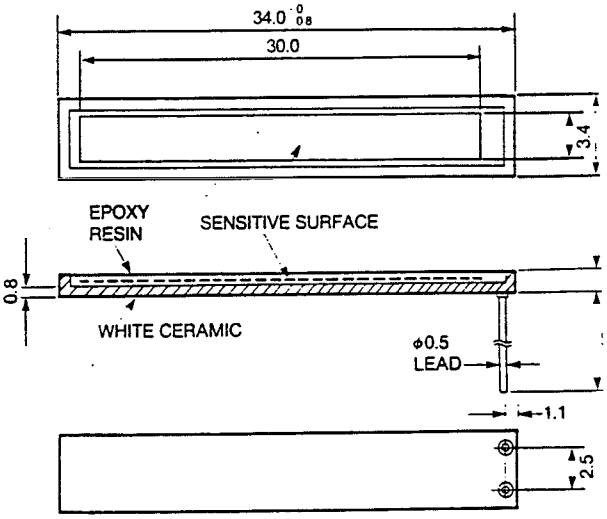


Unit

❺ S3584 Series



❻ S3588 Series



Windowless Types Available

Our standard types have clear epoxy resin windows. However, windowless types are also available. (Bonding wires are protected at one point by resin coating.)

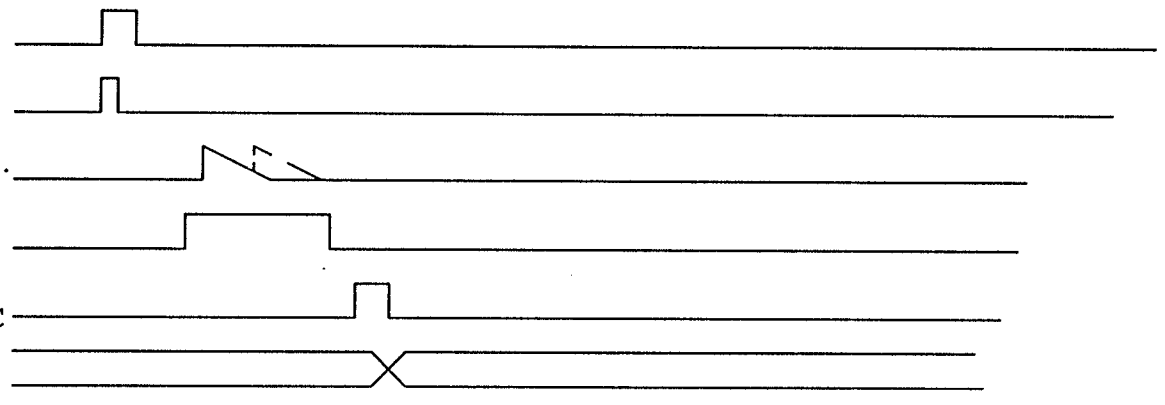
Quick reference for Type No. suffix is as follows.

Suffix No.	Wafer Thickness	Window
-01	200 μm	Epoxy resin
-02		Non
-03	300 μm	Epoxy resin
-04		Non
-05	500 μm	Epoxy resin
-06		Non

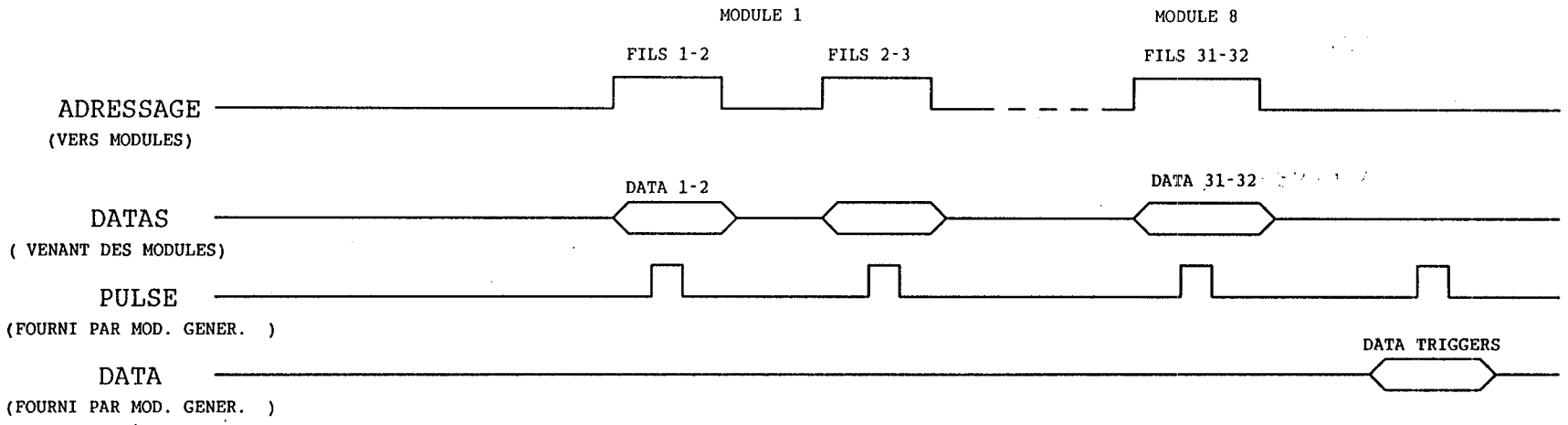
ANNEXE 2

Comportement dans le temps des 9 modules

TIMING
DE CHACUN
DES 8 MODULES



MODULE
GENERAL



ANNEXE 3

Programme 'Leadring.car'

```
+TITLE.
* LEADRING  0.0 /00  912210  1530 GEANT EXAMPLES
*
* L3 Lead ring calorimeter.
*
* UPDATES
* -------
*  22-10-91 : First version
*
+PATCH,LEADRING.
+DECK,BLANKDEK.
+KEEP,PVOLUM.
      COMMON/PVOLUM/ IMAT1,IMAT2,RPMAX(6),RPMIN(6),DZP(6),
     +               RSMAX(3),RSMIN(3),DZS(3),
     +               RCMAX(2),RCMIN(2),DZC(2),NCOUCHE(2),
     +               NZ,ZPOS,
     +               OFFPHI2,OFFPHI3,THEMIN,THEMAX,PHIMIN,PHIMAX
      COMMON/CUGEOM/HFRAM(3)
+KEEP,UVAR.
      PARAMETER (NPETA=16,NTIM=10)
      COMMON/RINGEV/IEVT,EBEAM,THETAN,PHIN,ESEEN,S1TOT,S2TOT,
     +              S3TOT,C1TOT,C2TOT,S1(NPETA),S2(NPETA),
     +              S3(NPETA),C1(100),C2(100)
+KEEP,UTFILES.
      COMMON/UTFILE/HNAME
      CHARACTER HNAME*16
+DECK,MAIN.
*
      PROGRAM MAIN
      PARAMETER (NGEANT=900000,NHBOOK=500000)
      COMMON/PAWC   /H(NHBOOK)
      COMMON/GCBANK/Q(NGEANT)
*
      CALL GZEBRA( NGEANT)
      CALL HLIMIT(-NHBOOK)
*
* *** GEANT initialisation
      CALL UGINIT
*
* *** Start events processing
      CALL GRUN
*
* ***  End of RUN
      CALL UGLAST
*
      STOP
      END
+DECK,UGINIT
      SUBROUTINE UGINIT
*
*     To initialise GEANT/USER  program and read data cards
*
+SEQ,GCKINE.
+SEQ,PVOLUM.
+SEQ,UTFILES.
+SEQ,GCBANK,GCLIST.
```



```
*
* *** Initialise GEANT
      CALL GINIT
*
      CALL VZERO (PKINE,10)
      ISETU=0
      IMAT1=13
      IMAT2=23
      RPMAX(1)=16.75
      RPMAX(2)=16.75
      RPMAX(3)=16.75
      RPMAX(4)=17.75
      RPMAX(5)=17.75
      RPMAX(6)=17.0
*
```

```
      RPMIN(1)=7.5
      RPMIN(2)=7.65
      RPMIN(3)=7.85
      RPMIN(4)=8.0
      RPMIN(5)=8.2
      RPMIN(6)=9.2
*
      DZP(1)=1.4
      DZP(2)=1.1
      DZP(3)=1.1
      DZP(4)=1.1
      DZP(5)=1.1
      DZP(6)=6.2
*
      RSMAX(1)=16.75
      RSMAX(2)=16.75
      RSMAX(3)=16.75
*
      RSMIN(1)=8.7
      RSMIN(2)=8.7
      RSMIN(3)=9.0
*
      DZS(1)=1.4
      DZS(2)=1.4
      DZS(3)=1.4
*
      RCMAX(1)=16.75
      RCMAX(2)=17.75
*
      RCMIN(1)=7.8
      RCMIN(2)=8.15
*
      DZC(1)=1.7
      DZC(2)=1.7
*
* 5 IS 1/0.2 cm
      NCOUCHE(1)=INT((RCMAX(1)-RCMIN(1))*5)
      NCOUCHE(2)=INT((RCMAX(2)-RCMIN(2))*5)
*
      ZPOS=108.0
      NZ=16
```



```
*
      OFFPHI2 = 0.
      OFFPHI3 = 0.
*
      IKINE=103
      PKINE(1)= 10.
      PKINE(2)=  6.
      PKINE(3)= 11.
*
      THEMIN = 4.7
      THEMAX = 9.3
      PHIMIN = 0.
      PHIMAX = 360.
*
      HNAME = 'leadring3.hist '
*
      CALL FFKEY('MATI',IMAT1,2,'REAL')
      CALL FFKEY('LAYE',RPMAX(1),37,'MIXED')
      CALL FFKEY('PFIX',PREFIX,1,'MIXED')
      CALL FFKEY('THPH',THEMIN,4,'REAL')
      CALL FFKEY('ROTA',OFFPHI2,2,'REAL')
*
* *** read data cards
      OPEN(UNIT=55,FILE='leadring.ffread',STATUS='UNKNOWN')
      CALL FFSET('LINP',55)
      CALL GFFGO
      CLOSE(55)
*
      DO I=1,6
        PRINT *,'RPMAX(',I,')',RPMAX(I)
        PRINT *,'RPMIN(',I,')',RPMIN(I)
        PRINT *,'DZP(',I,')',DZP(I)
      ENDDO
      DO I=1,3
        PRINT *,'RSMAX(',I,')',RSMAX(I)
```

```
        PRINT *,'RSMIN(',I,')',RSMIN(I)
        PRINT *,'DZS(',I,')',DZS(I)
      ENDDO
      DO I=1,2
        PRINT *,'RCMAX(',I,')',RCMAX(I)
        PRINT *,'RCMIN(',I,')',RCMIN(I)
        PRINT *,'DZC(',I,')',DZC(I)
        PRINT *,'NCOUCHE(',I,')',NCOUCHE(I)
      ENDDO
*
      CALL UFILES

*
* *** achieve initialization
      CALL GZINIT
*
      CALL GPART
*       Geometry and materials description
      CALL UGEOM

* *** Energy loss and cross-sections initialisations
```



```
      CALL GPHYSI
*
* *** Define user histograms
      CALL UHINIT
*
* *** Print control
*     CALL GPRINT('MATE',0)
*     CALL GPRINT('TMED',0)
      CALL GPRINT('VOLU',0)
*     CALL GPSETS('*','*')
*
      END

+DECK,UGEOM.
      SUBROUTINE UGEOM
*
* *** Define user geometry set up
*
+SEQ,GCBANK.
+SEQ,GCONST,GCLIST.
+SEQ,PVOLUM.
+SEQ,UVAR.
*
      DIMENSION PAR( 6)

      DIMENSION ZLG(6),ALG(6),WLG(6)
      DIMENSION A(3),Z(3),WMAT(3)
      DIMENSION AF(3),ZF(3),WMATF(3)
      DIMENSION AURAN(2),ZURAN(2),WURAN(2)
      DIMENSION AWATER(2),ZWATER(2),WWATER(2)
      DIMENSION AISOBU(3),ZISOBU(3),WGAS(3)
      DIMENSION ASCI(2),ZSCI(2),WSCI(2)
C
C             Argon/Isobuthane compound
C
      DATA AISOBU/12.01,1.01,39.95/
      DATA ZISOBU/6.,1.,18./
C
C             Water compound parameters
C
      DATA AWATER/1.,16./
      DATA ZWATER/1.,8./
      DATA WWATER/2.,1./
C
C             Lead glass mixture parameters
C
      DATA ZLG/  82.00,  19.00,  14.00,  11.00,  8.00,  33.00/
      DATA ALG/ 207.19,  39.102,  28.088,  22.99, 15.999,  74.922/
      DATA WLG/ .65994, .00799, .126676, .0040073,.199281, .00200485/
C
C             BGO compound parameters
C
      DATA A/208.98,72.59,15.999/
      DATA Z/83.,32.,8./
      DATA WMAT/4.,3.,12./
```

```
C
C             Iron+Nickel+Crome  compound parameters
C
      DATA AF/55.847,58.71,51.998/
      DATA ZF/26.,28.,24./
      DATA WMATF/0.703964,0.099,0.197/
      DATA ALAR,ZLAR,WLAR,DLAR,NLAR/40.,18.,1.,1.40,-1/
C
C             Uranium mixture
C
      DATA AURAN/235.,238./
      DATA ZURAN/92.,92./
      DATA WURAN/0.004,0.996/
C
C             Scintillator
C
      DATA ASCI/12.,1./
      DATA ZSCI/6.,1./
      DATA WSCI/1.,1./
      DATA DSCI/1.032/
C
C ***          Defines USER particular materials
C
*
*             Argon/Isobuthane mixture (60% Ar and 40% Isobuthane)
*             First define Isobuthane compound and relative weights
*
      DISO   =0.00267
      DENS1  =0.002136
      WGAS(1)=4.
      WGAS(2)=10.
      CALL GSMIXT(4,'ISOBUTHAN$',AISOBU,ZISOBU,0.40*DISO,-2,WGAS)
      WGAS(1)=0.40*WGAS(1)
      WGAS(2)=0.40*WGAS(2)
      WGAS(3)=0.60
      CALL GSMIXT( 5,'ARG/ISOBU$',AISOBU,ZISOBU,DENS1, 3,WGAS)
      CALL GSMATE( 9,'ALUMINIUM$', 26.98,13.,2.7   , 8.9,37.2,0,0)
      CALL GSMATE(11,'COPPER$    ', 63.54,29.,8.96  ,1.43,14.8,0,0)
      CALL GSMATE(12,'LEAD$      ',207.19,82.,11.35 ,0.56,18.5,0,0)
      CALL GSMATE(13,'LEAD$      ',207.19,82.,11.35 ,0.56,18.5,0,0)
*      CALL GSMATE(14,'URANIUM$   ',238.03,92.,18.95 ,0.32,12. ,0,0)
      CALL GSMIXT(14,'URANIUM$   ',AURAN,ZURAN,18.95 ,2,WURAN)
      CALL GSMATE(15,'AIR$       ', 14.61,7.3,0.001205,30423.,6750.,0,0)
      CALL GSMATE(16,'VACUUM$ ',1.E-16,1.E-16,1.E-16,1.E+16,1.E+16,0,0)
      CALL GSMIXT(10,'IRON(COMPOUND)$',AF,ZF,7.8,3,WMATF)
      CALL GSMIXT(21,'BGO(COMPOUND)$',A,Z,7.1,-3,WMAT)
      CALL GSMIXT(22,'LEAD GLASS$',ALG,ZLG,5.2,6,WLG)
      CALL GSMATE(23,'PLAST SC$',  6.25,3.4,1.032 ,   43.0,   437.,0,0)
      CALL GSMIXT(24,'SCINTILLATOR$',ASCI,ZSCI,DSCI,-2,WSCI)
      CALL GSMIXT(25,'liq. Argon$',ALAR,ZLAR,DLAR,NLAR,WLAR)
      CALL GSMIXT(26,'Water$$',AWATER,ZWATER,1.,-2,WWATER)
      CALL GSMATE(32,'LEADLEGER$',207.19,82.,6. ,0.56,18.5,0,0)
      CALL GSMATE(33,'LEADLEGER$',207.19,82.,6. ,0.56,18.5,0,0)
*
* *** Defines USER tracking media parameters
      FIELDM =  0.0
      IFIELD =  0
```



```
      TMAXFD = 10.0
      DMAXMS =  0.50
      DEEMAX =  0.20
      EPSIL  =  0.0100
      STMIN  =  0.0100
      CALL GSTMED( 1,'ABSORBER$'              ,IMAT1, 0 , IFIELD,
     *              FIELDM,TMAXFD,DMAXMS,DEEMAX, EPSIL, STMIN, 0 , 0 )
      CALL GSTMED( 2,'ACTIVE MEDIUM$'         ,IMAT2, 1 , IFIELD,
     *              FIELDM,TMAXFD,DMAXMS,DEEMAX, EPSIL, STMIN, 0 , 0 )
```

```
      CALL GSTMED( 3,'VACCUUM$'                    ,16, 0 , IFIELD,
     *                FIELDM,TMAXFD,DMAXMS,DEEMAX, EPSIL, STMIN, 0 , 0 )
      CALL GSTMED( 4,'ALUM$'                       , 9, 0 , IFIELD,
     *                FIELDM,TMAXFD,DMAXMS,DEEMAX, EPSIL, STMIN, 0 , 0 )


      IF(IMAT2.EQ.24)THEN
         CALL GSTPAR(2,'BIRK1',1.)
         CALL GSTPAR(2,'BIRK2',0.013)
         CALL GSTPAR(2,'BIRK3',9.6E-6)
      ENDIF


C
C ***          Defines USER'S VOLUMES
C
      JMA     = LQ(JMATE-IMAT1)
      X1      = Q(JMA+9)
      ABS1    = Q(JMA+10)
      JMA     = LQ(JMATE-IMAT2)
      X2      = Q(JMA+9)
      PRINT *,'X1 =',X1,' X2 =',X2,'ABS1 =',ABS1
*
      THX=90.
      PHX=OFFPHI2
      THY=90.
      PHY=90.
      THZ=0.
      PHZ=0.
      CALL GSROTM(1,THX,PHX,THY,PHY,THZ,PHZ)
      PHX=OFFPHI3
      CALL GSROTM(2,THX,PHX,THY,PHY,THZ,PHZ)
*
* Mother FRAM
*     zpos=0
      HFRAM(1) =  0.
      HFRAM(2) = 1.1*RPMAX(5)
      HFRAM(3) = 0
      DO I=1,6
        HFRAM(3)=HFRAM(3)+DZP(I)
      ENDDO
      DO I=1,3
        HFRAM(3)=HFRAM(3)+DZS(I)
      ENDDO
      DO I=1,2
        HFRAM(3)=HFRAM(3)+DZC(I)
```



```
      ENDDO
      PAR(3)=0.5*HFRAM(3)
      HFRAM(3) = 1.1*(ZPOS+HFRAM(3))
      CALL GSVOLU('FRAM', 'TUBE', 3, HFRAM, 3, IVOL)
*
      PAR(1)=0.
      PAR(2)=RPMAX(1)
      PAR(3)=0.5*DZP(1)
      CALL GSVOLU('PB0X', 'TUBE', 1, PAR, 0, IVOL)
      CALL GSVOLU('SS0X', 'TUBE', 2, PAR, 0, IVOL)
      CALL GSDVN ('SSPH' , 'SS0X' ,NZ ,2)
      CALL GSVOLU('SC01', 'TUBE', 2, PAR, 0, IVOL)
      CALL GSDVN ('SCT1' , 'SC01' ,NCOUCHE(1) ,1)
      CALL GSVOLU('SC02', 'TUBE', 2, PAR, 0, IVOL)
      CALL GSDVN ('SCT2' , 'SC02' ,NCOUCHE(2) ,1)
      CALL GSVOLU('ALUM', 'TUBE', 4, PAR, 0, IVOL)

      XPOS=ZPOS
      PAR(1)=0.
*PB1
      PAR(1)=RPMIN(1)
      PAR(2)=RPMAX(1)
      PAR(3)=0.5*DZP(1)
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('PB0X',1,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZP(1)
*ALU1
      PAR(1)=RPMIN(1)
      PAR(2)=RPMAX(1)
```

```
      PAR(3)=0.125/2.
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('ALUM',1,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*S1
      PAR(1)=RSMIN(1)
      PAR(2)=RSMAX(1)
      PAR(3)=0.5*DZS(1)
      ZPOS=XPOS+PAR(3)
      PAR(3)=0.5
      CALL GSPOSP('SS0X',1,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZS(1)
*ALU2
      PAR(1)=RPMIN(1)
      PAR(2)=RPMAX(1)
      PAR(3)=0.125/2.
      ZPOS=XPOS-PAR(3)
      CALL GSPOSP('ALUM',2,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*PB2
      PAR(1)=RPMIN(2)
      PAR(2)=RPMAX(2)
      PAR(3)=0.5*DZP(2)
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('PB0X',2,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZP(2)
*ALU3
      PAR(1)=RPMIN(2)
      PAR(2)=RPMAX(2)
```



```
      PAR(3)=0.125/2.
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('ALUM',3,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*C1
      PAR(1)=RCMIN(1)
      PAR(2)=RCMAX(1)
      PAR(3)=0.5*DZC(1)
      ZPOS=XPOS+PAR(3)
      PAR(3)=0.6
      CALL GSPOSP('SC01',1,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZC(1)
*ALU4
      PAR(1)=RPMIN(2)
      PAR(2)=RPMAX(2)
      PAR(3)=0.125/2.
      ZPOS=XPOS-PAR(3)
      CALL GSPOSP('ALUM',4,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*PB3
      PAR(1)=RPMIN(3)
      PAR(2)=RPMAX(3)
      PAR(3)=0.5*DZP(3)
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('PB0X',3,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZP(3)
*ALU5
      PAR(1)=RPMIN(3)
      PAR(2)=RPMAX(3)
      PAR(3)=0.125/2.
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('ALUM',5,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*S2
      PAR(1)=RSMIN(2)
      PAR(2)=RSMAX(2)
      PAR(3)=0.5*DZS(2)
      ZPOS=XPOS+PAR(3)
      PAR(3)=0.5
      CALL GSPOSP('SS0X',2,'FRAM',0.,0., ZPOS,1,'ONLY',PAR,3)
      XPOS=XPOS+DZS(2)
*ALU6
      PAR(1)=RPMIN(3)
      PAR(2)=RPMAX(3)
      PAR(3)=0.125/2.
      ZPOS=XPOS-PAR(3)
      CALL GSPOSP('ALUM',6,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*PB4
      PAR(1)=RPMIN(4)
      PAR(2)=RPMAX(4)
      PAR(3)=0.5*DZP(4)
      ZPOS=XPOS+PAR(3)
```

```
      CALL GSPOSP('PB0X',4,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZP(4)
*ALU7
      PAR(1)=RPMIN(4)
      PAR(2)=RPMAX(4)
      PAR(3)=0.125/2.
      ZPOS=XPOS+PAR(3)
```



```
      CALL GSPOSP('ALUM',7,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*C2
      PAR(1)=RCMIN(2)
      PAR(2)=RCMAX(2)
      PAR(3)=0.5*DZC(2)
      ZPOS=XPOS+PAR(3)
      PAR(3)=0.6
      CALL GSPOSP('SC02',1,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZC(2)
*ALU8
      PAR(1)=RPMIN(4)
      PAR(2)=RPMAX(4)
      PAR(3)=0.125/2.
      ZPOS=XPOS-PAR(3)
      CALL GSPOSP('ALUM',8,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*PB5
      PAR(1)=RPMIN(5)
      PAR(2)=RPMAX(5)
      PAR(3)=0.5*DZP(5)
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('PB0X',5,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZP(5)
*ALU9
      PAR(1)=RPMIN(5)
      PAR(2)=RPMAX(5)
      PAR(3)=0.125/2.
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('ALUM',9,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*S3
      PAR(1)=RSMIN(3)
      PAR(2)=RSMAX(3)
      PAR(3)=0.5*DZS(3)
      ZPOS=XPOS+PAR(3)
      PAR(3)=0.5
      CALL GSPOSP('SS0X',3,'FRAM',0.,0., ZPOS,2,'ONLY',PAR,3)
      XPOS=XPOS+DZS(3)
*ALU10
      PAR(1)=RPMIN(5)
      PAR(2)=RPMAX(5)
      PAR(3)=0.125/2.
      ZPOS=XPOS-PAR(3)
      CALL GSPOSP('ALUM',10,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
*PB6
      PAR(1)=RPMIN(6)
      PAR(2)=RPMAX(6)
      PAR(3)=0.5*DZP(6)
      ZPOS=XPOS+PAR(3)
      CALL GSPOSP('PB0X',6,'FRAM',0.,0., ZPOS,0,'ONLY',PAR,3)
      XPOS=XPOS+DZP(6)
*
* *** Close geometry banks. (obligatory system routine)
      CALL GGCLOS
      END

+DECK, UFILES.
      SUBROUTINE UFILES
```



```
+SEQ,UTFILES.
*
      OPEN(UNIT=8,FILE='leadring.meta',STATUS='UNKNOWN')
```

```
      OPEN(UNIT=34,FILE=HNAME,ACCESS='DIRECT',
     +     RECL=4096,STATUS='UNKNOWN')
      END

+DECK,UHINIT.
      SUBROUTINE UHINIT
+SEQ,UVAR.
*     To book the user's histograms
      CALL HRFILE(34,'HBOOK','N')
*
      CALL HBOOK1(  1,'total energy seen in scintillator
     + (in % of E inc.)',100, 0. ,   10. , 0.)
*
      CALL HBNT(100,'RING',' ')
      IEVT=0
      CALL HBNAME(100,'RINGEV',IEVT,
     +            'IEVT,EBEAM,THETA,PHI,ESEEN,S1TOT,S2TOT,'//
     +            'S3TOT,C1TOT,C2TOT,S1(16),S2(16),S3(16),'//
     +            'C1(100),C2(100)')
      END

+DECK,GUKINE
      SUBROUTINE GUKINE
*    Generates Kinematics for primary track
*    Kine : itype  energy  vx  vy  vz  teta  phi
+SEQ,GCBANK.
+SEQ,GCFLAG,GCKINE,GCONST.
+SEQ,UVAR.
+SEQ,PVOLUM.
      DIMENSION VERTEX(3),PLAB(3),RNDM(2)
      IF(IKINE.GT.100)THEN
         IK=IKINE-100
         TETA=PKINE(2)*DEGRAD
         PHI =PKINE(3)*DEGRAD
      ELSE
         IK=IKINE
         CALL GRNDM(RNDM,2)
         TETA = THEMIN + RNDM(1)*(THEMAX-THEMIN)
         TETA = TETA*DEGRAD
         PHI = PHIMIN + RNDM(2)*(PHIMAX-PHIMIN)
         PHI = PHI*DEGRAD
      ENDIF
*
      JPA = LQ(JPART-IK)
      XMASS = Q(JPA+7)
      ETOT  = PKINE(1)
      PMOM  = SQRT(PKINE(1)**2 - XMASS**2)
      PKINE(10) = ETOT
      VERTEX(1) = 0.
      VERTEX(2) = 0.
      VERTEX(3) = 0.
*
      PLAB(1) = PMOM*SIN(TETA)*COS(PHI)
```



```
      PLAB(2) = PMOM*SIN(TETA)*SIN(PHI)
      PLAB(3) = PMOM*COS(TETA)
*
      PHIN = PHI*RADDEG
      THETAN = TETA*RADDEG
      EBEAM  = ETOT
*
      CALL GSVERT(VERTEX,0,0,0,0,NVERT)
      CALL GSKINE(PLAB,IK,NVERT,0,0,NT)
* *** Kinematics debug (controlled by ISWIT(1) )
*      CALL GPRINT('VERT',0)
*      CALL GPRINT('KINE',0)
      END

+DECK,GUTREV
      SUBROUTINE GUTREV
*     User routine to control tracking of one event
*     Called by GRUN
+SEQ,UVAR.
* *** initialise some arrays
      CALL VZERO(S1,NPETA)
      CALL VZERO(S2,NPETA)
      CALL VZERO(S3,NPETA)
```

```
*
      CALL VZERO(C1,100)
      CALL VZERO(C2,100)
*
* *** Start the tracking of the event
      CALL GTREVE
      if (mod(ievt,50).eq.0.or.ievt.lt.5) then
      print*,'evt numb ',ievt+1
      endif
      END

+DECK,GUSTEP
      SUBROUTINE GUSTEP
*
*     User routine called at the end of each tracking step
*
+SEQ,GCKING,GCTMED,GCTRAK,GCVOLU.
+SEQ,GCKINE,GCSETS,GCCUTS.
+SEQ,UVAR.
*
* *** Debug event
      CALL GDEBUG
*
* *** Something generated ?
      IF(NGKINE.GT.0) CALL GSKING(0)
*
* *** find PETAL KR
      KR = 0
      KE = 0
      IF (NLEVEL.GE.3) THEN
* COUCHE
         KR = LINDEX(2)
* DIVISION
```



```
         KE = LINDEX(3)
C         WRITE(6,*)'L-TEST ',NUMED,KR,KE,DESTEP
      ENDIF
* *** Total energy deposited per R-module
      IF ((KE.GT.0).AND.(DESTEP.GT.0.)) THEN
*          NUMED NORMALEMENT TOUJOURS EGALE A 2.
         IF (NUMED.EQ.2) THEN
           IF(KR.EQ.3) S1(KE)=S1(KE)+DESTEP
           IF(KR.EQ.7) C1(KE)=C1(KE)+DESTEP
           IF(KR.EQ.11) S2(KE)=S2(KE)+DESTEP
           IF(KR.EQ.15) C2(KE)=C2(KE)+DESTEP
           IF(KR.EQ.19) S3(KE)=S3(KE)+DESTEP
         ENDIF
         IF (NUMED.NE.2) SUMABS = SUMABS + DESTEP
      ENDIF
*
      END

+DECK,GUOUT
      SUBROUTINE GUOUT
*
*     User routine called at the end of each event
*
+SEQ,GCKINE.
+SEQ,UVAR.
+SEQ,PVOLUM.
* *** Total energy deposited
      S1TOT=0
      S2TOT=0
      S3TOT=0
      C1TOT=0
      C2TOT=0
*
      DO I=1,NPETA
        S1TOT=S1TOT+S1(I)
        S2TOT=S2TOT+S2(I)
        S3TOT=S3TOT+S3(I)
      ENDDO
*
      DO I=1,NCOUCHE(1)
        C1TOT=C1TOT+C1(I)
      ENDDO
*
```

```
      DO I=1,NCOUCHE(2)
        C2TOT=C2TOT+C2(I)
      ENDDO
*
      ESEEN=S1TOT+S2TOT+S3TOT+C1TOT+C2TOT

      IEVT=IEVT+1
      CALL HFNT(100)
*
      ESEEN = 1.E2*ESEEN/PKINE(1)
      CALL HFILL (1,ESEEN, 0., 1.)
*
* *** Print current mean values
```



```
      EMEAN = HSTATI(1,1,' ',0)
      RMS   = HSTATI(1,2,' ',0)
      RMS   = 100.*RMS/EMEAN
      WRITE (LOUT,1001) IEVENT,EMEAN,RMS
 1001 FORMAT(1X,'Event number:',I3,
     +       3X,'ESEEN/E0 (%):',F5.2, '  rms/eseen (%):',F5.2)
*
      END

+DECK,UGLAST
      SUBROUTINE UGLAST
*     Termination routine to print histograms and statistics
+SEQ,UVAR.

      CALL GLAST
*
      CALL HROUT(0,ICYCLE,' ')
 28   CALL HREND('HBOOK')
*
* *** Print histograms
      CALL HISTDO
      END
```

ANNEXE 4

Programme 'Leadring.ffread'

```
LIST
TRIG  10000
RNDM  12345 54321
! CUTS  1=0.0001  3=0.001 4=0.00001 5=0.001 10=0.00204
! LOSS 3
! DRAY 1
! MULS 2
!---------------------------- KINE  103 10. 6. 11.
!
KINE  3 90. 6. 18.75
! PFIX 'e45g'
PFIX 'gxin'
! LAYE RMAX RMIN1 RMIN2 RMIN3 RMIN4 DZS1 DZS2 DZS3 DZPB1 DZPB2 DZPB3 DZPB4 NZ ZPOS
ROTA 7.5 15.
THPH 3.9 8.67 15. 37.5
DEBU   1   10
TIME 0. 0. 0
END
```

ANNEXE 5

Programme 'Install-lead'

```
!/bin/sh

Make the standard executable and .o file for the various
UNIX flavo(u)rs

This makes it possible to link your own stuff (for example, myfile.f)
to the .o file  by using the command:

$FC $FFLAGS myfile.f $NAME.o -o myfile.exe $LDFLAGS

where you have to sustitute $FC, $FFLAGS $NAME and $LDFLAGS from the
relevant lines below...(left as an exercise ;-> )

Gerhard Raven (graven@hpl3.cern.ch)
Modified by R Clare: use the combined l3 library

Set the default values for the L3/CERN tree
NOTE: you can always override these values if you do (for example)
      CERN_LEVEL=new ; export CERN_LEVEL (in case of sh,ksh,zsh,bash,..)
      setenv CERN_LEVEL new              (in case of csh,tcsh,..)

-z "$L3" ]          && L3=/l3
-z "$L3_LEVEL" ]    && L3_LEVEL=180
-z "$L3_ROOT" ]     && L3_ROOT=$L3/v$L3_LEVEL

-z "$CERN" ]        && CERN=/cern
-z "$CERN_LEVEL" ] && case $L3_LEVEL in
                          dev | 180 ) CERN_LEVEL=pro ;;
                          171 | 170 ) CERN_LEVEL=92a ;;
                          *   )       echo 'CERN_LEVEL for $L3_LEVEL not present'
                                      exit ;;
                      esac
-z "$CERN_ROOT" ]   && CERN_ROOT=$CERN/$CERN_LEVEL

the target OS; first try uname, if no succes try for Apollo...

-z "$UNAME" ]       && UNAME=`(uname) 2>/dev/null`
-z "$UNAME" -a -d /sys/node_data ] && UNAME="DomainOS"
-z "$UNAME" ]       && (echo could not determine hosttype ; exit)

 the target program name

IAME=leadring

 The libraries

.IBS="-L$L3_ROOT/lib -L$CERN_ROOT/lib -L/usr/lib/X11R4 -L/lib \
       -ldb -lgeant321 -lpawlib -lgraflib -lgrafX11 \
      -lmathlib -lpacklib -lkernlib -lgenlib  -lX11 -lm -lcl"

 Machine specific options

.f [ "$UNAME" = "HP-UX" ] ; then

    MACHINE=HPUX
    FC=fort77
```



```
    FFLAGS="-K -O +ppu +T"
    LDFLAGS="-v -Wl,-s $LIBS"
# to use gprof utilities
#    FFLAGS="-G -K -O +ppu +T"
#    LDFLAGS="-v $LIBS"

elif [ "$UNAME" = "DomainOS" ] ; then

    MACHINE=APOLLO
    FC=f77
    FFLAGS=" -O -W0,-nuc,-bounds_violation,-save,-indexl,-zero,-inline,!"
    if /usr/apollo/bin/prism ; then
        FFLAGS="$FFLAGS,-natural -A cpu,a88k"
    else
        FFLAGS="$FFLAGS,-nclines -A cpu,mathlib"
```

```
    fi
    LDFLAGS="-Wl,-u,recbst,-u,recbts,-u,respar,-u,rexymc,-u,axocmp,-u,apgfac,-u,refld \
             $LIBS"

elif [ "$UNAME" = "AIX" ] ; then

    MACHINE=RS
    FC=f77
    FFLAGS=
    echo 'RS/6000 with AIX not yet supported'
    exit

elif [ "$UNAME" = "IRIX" ] ; then

    MACHINE=SGI
    FC=f77
    FFLAGS="-trapuv -static -Nn4000 -backslash -G 3"
    LDFLAGS="-Wl,-u,recbst_,-u,recbts_,-u,respar_,-u,rexymc_,-u,axocmp_,-u,apgfac_,-u,r
             $LIBS -lshift"

else

    echo $UNAME is an unsupported system
    exit

fi


echo " "
echo " ################################################"
echo " ################################################"
echo " ## "
echo " ## installing $NAME for $MACHINE "
echo " ## using $CERN_ROOT and $L3_ROOT "
echo " ## "
echo " ################################################"
echo " ################################################"
echo " "

[ "x$L3_LEVEL" = "xdev" ] && touch l3corr$L3_LEVEL.car \
                          || crl3corr -v $L3_LEVEL -p l3corr$L3_LEVEL.car
```



```
ypatchy - =$NAME.f :GO << EOI
+EXE, CRA*.
+OPT, MAP, LOST, WARN.
+SELF.
+USE, $MACHINE.
+USE, TYPE, QCDE, FATCDES, GCDES, ERPRCDES, UVAR.
+SELF.
+USE, LEADRING, T=EXE.
+PAM, 11, R=GCDES,        T=C,T=A.  /l3/pam/geant351.car
+PAM, 11,                 T=C, T=A. leadring.car
+QUIT.
EOI

# make the executable and the object file...

#$FC $FFLAGS  /cern/pro/lib/gxint321.f $NAME.f -o $NAME.exe $LDFLAGS
$FC $FFLAGS  $NAME.f -o $NAME.exe $LDFLAGS
chmod 755 $NAME.exe
```

ANNEXE 6

Programme 't3-alr.for'

```
      REAL FUNCTION t3_alr(XDUMMY)
************************************************************
*                                                          *
* This file was generated by HUWFUN.                       *
*                                                          *
************************************************************
*                                                          *
*     Ntuple Id:      100                                  *
*     Ntuple Title:   RING                                 *
*     Creation:       30/08/94 15.43.55                    *
*                                                          *
************************************************************

      LOGICAL         CHAIN
      LOGICAL debug
      LOGICAL debug10
      LOGICAL lpass1

      CHARACTER*128   CFILE

      PARAMETER (nthebin=25,nphibin=20)
*
      INTEGER IDNEVT
      INTEGER NCHEVT
      INTEGER ICHEVT
      INTEGER NEVTOT
      INTEGER nphi
      INTEGER ntheta
      INTEGER ista(16,3,nthebin,nphibin)
*
      PARAMETER (NEVTOT = 6048)
*
      REAL VIDN1
      REAL VIDN2
      REAL VIDN3
      REAL VIDN(10)
*
      REAL wphi(2)
      REAL wtheta(2)
      REAL energy(16,3,nthebin,nphibin)
      REAL phistep
      REAL thetastep
      REAL s_can_1(nthebin,nphibin)
      REAL c_can_1(nthebin,nphibin)
      REAL s_effi(nthebin,nphibin)
      REAL sc_effi(nthebin,nphibin)
      REAL sc_uneffi(nthebin,nphibin)
      REAL B(nthebin,nphibin)
      REAL sc_coup(nthebin,nphibin)
      REAL s_coup(nthebin,nphibin)
      REAL coup_tot(nthebin,nphibin)
*
      DATA emin/0.001/
      DATA lpass1/.false./
*
      COMMON/WORK/wphi,wtheta,energy,
     +            phistep,thetastep,nphi,ntheta,ssigma,
     +            ista,debug,debug10

*
      COMMON /PAWIDN/ IDNEVT,VIDN1,VIDN2,VIDN3,VIDN
      COMMON /PAWCHN/ CHAIN, NCHEVT, ICHEVT
      COMMON /PAWCHC/ CFILE

*--   Ntuple Variable Declarations
*
      REAL EBEAM
      REAL THETA
      REAL PHI
      REAL ESEEN
      REAL S1TOT
      REAL S2TOT
      REAL S3TOT
      REAL C1TOT
      REAL C2TOT
      REAL S1(16)
      REAL S2(16)
      REAL S3(16)
      REAL C1(100)
      REAL C2(100)
*
      INTEGER IEVT
*
```

```
      COMMON /PAWCR4/ IEVT,EBEAM,THETA,PHI,ESEEN,S1TOT,S2TOT,S3TOT,
     +                C1TOT,C2TOT,S1,S2,S3,C1,C2

*--   Enter user code here
*
***************** Initialisation step *************************

       debug   = .false.
       debug10 = .false.
       if (ievt.le.1) then

         do i = 1,25
           do j = 1,20

             B(i,j) = 0
             c_can_1(i,j) = 0
             s_coup(i,j) = 0
             sc_effi(i,j) = 0
             s_can_1(i,j) = 0
             sc_coup(i,j) = 0
             s_effi(i,j) = 0
             sc_uneffi(i,j) = 0

           enddo
         enddo

         do is = 1,16
           do il = 1,3
             do i = 1,25
               do j = 1,20

                 ista(is,il,i,j) = 0

               enddo
             enddo
           enddo
         enddo


         call interactif

*         call bdata

*2345678901234567890123456789012345678901234567890123456789012345678901234567890190
**************************************************************************


         call hbook1(10,'EBEAM',100,0.,100.,0.)

         do it =1,ntheta
           do ip = 1,nphi

             call hbook1(10*it+ip,'s1tot',100,0.,0.45,0.)
             call hbook1(221*it+ip,'s2tot',100,0.,1.4,0.)
             call hbook1(4441*it+ip,'s3tot',100,0.,1.,0.)
             call hbook1(88841*it+ip,'s1tot+s2tot',100,0.,2.,0.)
             call hbook1(1776841*it+ip,'s1tot+s2tot+s3tot',100,0.,5.,0.)

             write (6,*) it,ip

           enddo
         enddo

         call hbook2(1,'efficiency (S scintillators)',
     +                 20,3.9,8.7,20,7.5,30.,0.)
         call hbook2(2,'evts number (S scintillators)',
     +                 20,3.9,8.7,20,7.5,30.,0.)
         call hbook2(3,'total efficiency (S+C scintillators)',
     +                 20,3.9,8.7,20,7.5,30.,0.)
         call hbook2(4,'evts number (C scintillators)',
     +                 20,3.9,8.7,20,7.5,30.,0.)


         write(6,*) 'Bienvenue dans le monde merveilleux de ALR+ ...!'
         write(6,*) ' '
         write(6,*) '*********initialisation *********'
         write(6,*) ' '

         debug = .true.

         write(6,*) '*********`alr.out` opened *********',ios
         write(6,*) ' '
```

```
c  ************* Calculate the step in phi ************************
         phistep = (wphi(2)-wphi(1))/nphi
c ************* Calculate the step in theta ************************
         thetastep = (wtheta(2)-wtheta(1))/ntheta
         if(debug) write(6,101) phistep
 101     format('You have defined a step of ', F5.2, ' degrees in phi')
         if(debug) write(6,102) thetastep
 102     format(' and of',F6.2, ' degrees in theta ')
         if(debug) write(6,*) ' '
c ********* Array length : 16 secteurs * 3 layeurs * 25 * 20 *******
         ncell =16*3*nthebin*nphibin
c ************* Procedure initialization ***************************
         call vzero(energy,ncell)
       endif
c ******** end of the  Initialisation step ************************
       call hfill(10,ebeam,0.,1.)
c ************************ Cells definition ************************
c   it = number of steps in theta
c   ip = number of steps in phi
c   is = number of sector
       if (idnevt.le.nevtot) then
         do it=1,ntheta
           do ip=1,nphi
             themin=wtheta(1)+(it-1)*thetastep
             themax=wtheta(1)+(it)*thetastep
             if (theta.ge.themin.and.theta.lt.themax) then
               phimin=wphi(1)+(ip-1)*phistep
               phimax=wphi(1)+(ip)*phistep
               if(debug) write(6,*) ' I m going to fill the variables'
               if(debug) write(6,*) ' '
               if (phi.ge.phimin.and.phi.lt.phimax) then
                 call hf1(10*it+ip,s1tot,1.)
                 call hf1(221*it+ip,s2tot,1.)
                 call hf1(4441*it+ip,s3tot,1.)
                 call hopera (10*it+ip,'+',221*it+ip,88841*it+ip,1.,1.)
                 call hopera (88841*it+ip,'+',4441*it+ip,
     +                                        17776841*it+ip,1.,1.)
                 B(it,ip) = B(it,ip) + 1
                 s_can_1(it,ip)=s1(1)+s2(1)+s3(1)+s1(2)+s2(2)+s3(2)
                 if (s_can_1(it,ip).ge.0.6) then
                   s_coup(it,ip) = s_coup(it,ip) + 1
                 endif
                 c_can_1(it,ip) = c1tot+c2tot
           if (s_can_1(it,ip).lt.0.6.and.c_can_1(it,ip).lt.1.2) then
                   sc_coup(it,ip) = sc_coup(it,ip) + 1
                 endif

*************************************************************************
c    *** Here I fill my variables
c        if the energy found is greater than 'emin' ( = 0.001 Gev = 1 Mev ). **
```

```
c    *** I take into account that contribution energy
c        will comtain at the end the requested quantity 'ista'
c        will contain the number of efficient events. ***

c    *** The mean energy will be the ration of those 2 variables ... see later

                  do is=1,16

                    if (s1(is).gt.emin) then

                      ista(is,1,it,ip)= ista(is,1,it,ip)+1
                      energy(is,1,it,ip)= energy(is,1,it,ip)+s1(is)

                    endif

                    if (s2(is).gt.emin) then

                      ista(is,2,it,ip)= ista(is,2,it,ip)+1
                      energy(is,2,it,ip)= energy(is,2,it,ip)+s2(is)

                    endif

                    if (s3(is).gt.emin) then

                      ista(is,3,it,ip)= ista(is,3,it,ip)+1
                      energy(is,3,it,ip)= energy(is,3,it,ip)+s3(is)

                    endif

                  enddo
                endif
              endif
            enddo
          enddo

          if (idnevt.eq.nevtot) then

            do it =1,ntheta
              do ip = 1,nphi

                if (B(it,ip).ne.0) then

                  s_effi(it,ip) = s_coup(it,ip)/B(it,ip)
                  s_effi(it,ip) = s_effi(it,ip)*100

                  sc_uneffi(it,ip) = sc_coup(it,ip)/B(it,ip)
                  sc_uneffi(it,ip) = sc_uneffi(it,ip)*100
                  sc_effi(it,ip) = 100. - sc_uneffi(it,ip)

                  themin=wtheta(1)+(it-1)*thetastep
                  phimin=wphi(1)+(ip-1)*phistep

                  call hf2(1,themin+0.01,phimin+0.01,s_effi(it,ip))
                  call hf2(2,themin+0.01,phimin+0.01,s_coup(it,ip))
                  call hf2(3,themin+0.01,phimin+0.01,sc_effi(it,ip))
                  call hf2(4,themin+0.01,phimin+0.01,sc_coup(it,ip))

c                 write(6,*) 'B(',it,'-',ip,')= ', B(it,ip)
c                 write(6,*) 'can_1(',it,'-',ip,')= ',can_1(it,ip)
c                 write(6,*) 'coup(',it,'-',ip,')= ',coup(it,ip)
                  write(6,*) 's_effi(',it,'-',ip,')= ',s_effi(it,ip)
                  write(6,*) 'sc_effi(',it,'-',ip,')= ',sc_effi(it,ip)

                endif

              enddo
            enddo

          endif

        endif

        if(debug) write(6,*) ' Now, all the loops have been done '
        if(debug) write(6,*) ' '

c   ************************ Call an output file ****************************

        if (idnevt.eq.nevtot)call Doutput(s_effi,sc_effi)

        t3_alr = 1.

        end

*************************************************************************

        SUBROUTINE Doutput(s_effi,sc_effi)
```

```
*************************************************************************
* This file will write a paper that contains the distribution of the
* deposit energy in each cell of each scintillator.
*         (c)  L. FREDJ   review D. SCIARRINO
*************************************************************************

      CHARACTER*5   sc(6)
      CHARACTER*18  pthe,pphi
*
      PARAMETER (nthebin=25,nphibin=20)
*
      REAL wphi(2)
      REAL wtheta(2)
      REAL energy(16,3,nthebin,nphibin)
      REAL tension(16,3,nthebin,nphibin)
      REAL phistep
      REAL thetastep
      REAL canal(16,nthebin,nphibin)
      REAL voie(16,nthebin,nphibin)
      REAL tcanal(16,nthebin,nphibin)
      REAL tcan(16,nthebin,nphibin)
      REAL tvoie(16,nthebin,nphibin)
      REAL tv(16,nthebin,nphibin)
      REAL s_effi(nthebin,nphibin)
      REAL sc_effi(nthebin,nphibin)
      REAL MAX(nthebin,nphibin)
      REAL VM

*
      INTEGER nphi
      INTEGER ntheta
      INTEGER ista(16,3,nthebin,nphibin)
      INTEGER DB(16,3,nthebin,nphibin)
      INTEGER de(16,3,nthebin,nphibin)
      INTEGER VMAX(nthebin,nphibin)
      INTEGER VMIN
      INTEGER wm
      INTEGER ONOFF1
      INTEGER ONOFF2
      INTEGER a,b

      INTEGER pt
*
      LOGICAL debug
      LOGICAL debug10
*
      COMMON/WORK/wphi,wtheta,energy,
     +            phistep,thetastep,nphi,ntheta,ssigma,
     +            ista,debug,debug10

*
      REAL EBEAM
      REAL THETA
      REAL PHI
      REAL ESEEN
      REAL S1TOT
      REAL S2TOT
      REAL S3TOT
      REAL C1TOT
      REAL C2TOT
      REAL S1(16)
      REAL S2(16)
      REAL S3(16)
      REAL C1(100)
      REAL C2(100)

      INTEGER IEVT
*
      COMMON /PAWCR4/ IEVT,EBEAM,THETA,PHI,ESEEN,S1TOT,S2TOT,S3TOT,
     +                C1TOT,C2TOT,S1,S2,S3,C1,C2

*
      do is = 1,16
        do il = 1,3
          do i = 1,25
            do j = 1,20

              tension(is,il,i,j)=0
              DB(is,il,i,j)=0
              db(is,il,i,j)=0

            enddo
          enddo
        enddo
```

```
      enddo

      do is = 1,16
        do i = 1,25
          do j = 1,20

            canal(is,i,j) = 0
            voie(is,i,j) = 0
            tcanal(is,i,j) = 0
            tcan(is,i,j) = 0
            tv(is,i,j) = 0

          enddo
        enddo
      enddo

      do i = 1,25
        do j = 1,20

          MAX(i,j)=0
          VMAX(i,j)=0

        enddo
      enddo

      VMIN=0
      ONOFF1=0
      ONOFF2=0

      write (6,*) 'yup'
      Write(56,*)' Alr simulation program version  1.1 '
      Write(56,*)' '
      Write(56,*)' The beam energy is ',EBEAM, ' GeV'
      Write(56,*)' '
      Write(56,100)
      Write(57,*)' Alr simulation program version  1.1 '
      Write(57,*)' '
      Write(57,*)' The beam energy is ',EBEAM, ' GeV'
      Write(57,*)' '
      Write(57,*)' WARNING : ALL THE TENSIONS ARE IN MILLIVOLTS '
      Write(57,*)' '
      Write(57,100)

*** Calculate the mean energy deposit in this cell for each scintillator ***

      do j=1,nphi
        do i=1,ntheta
          do is=1,16
            do il=1,3

              if(ista(is,il,i,j).gt.0)
     + energy(is,il,i,j) = 1000.*energy(is,il,i,j)/(ista(is,il,i,j))

              tension(is,il,i,j) = energy(is,il,i,j)*0.16

              if (ista(is,il,i,j).le.0.and.energy(is,il,i,j).gt.0.)then

                write(56,*) ' I found problems in the following case:'
                write(56,*) ' Cell number in phi ',i
                write(56,*) ' Cell number in phi ',j
                write(56,*) ' Sector number  ',is
                write(56,*) ' Layer number  ',il
                write(56,*) ' No entry but energy deposit  =',
     +                      energy(is,il,i,j)

              endif


            enddo
          enddo
        enddo
      enddo


c
c ****** Calcule l'energie par canal  (somme de 3 layers sur 1 petale) ******
c ****** Calcule l'energie par voie  (somme de 3 layers sur 2 petales) ******
c
      do is=1,16
        do i=1,ntheta
          do j=1,nphi
            do il=1,3

              canal(is,i,j) = canal(is,i,j)+energy(is,il,i,j)
              tcanal(is,i,j)  = canal(is,i,j)*0.16
```

```
          enddo
c         write(6,*) 'canal (',is,'-',i,'-',j,')=',canal(is,i,j)
        enddo
      enddo
    enddo

    do is=1,16
      do i=1,ntheta
        do j=1,nphi

          if(is.NE.16) voie(is,i,j)=canal(is,i,j)+canal(is+1,i,j)
          if(is.Eq.16) voie(is,i,j)=canal(is,i,j)+canal(1,i,j)

          tvoie(is,i,j) = voie(is,i,j)*0.16

        enddo
      enddo
    enddo
*
    i=0
    do is=1,2
      do il=1,3

        i=i+1
        write(sc(i),1)is,il

      enddo
    enddo

    write(6,*)' I m preparing the output file ...please wait ! '

    write(56,2)sc
    write(57,2)sc
    write(58,13)

    write(56,100)
    write(57,100)
    write(58,100)

    i=0
    pt=0
    do it=1,ntheta
      do ip=1,nphi

        If (ievt.le.1) then

          VM = tension(1,1,1,1)

        endif

        do ia=1,2
          do ib=1,3

            if (tension(ia,ib,it,ip).gt.VM) then

              VM = tension(ia,ib,it,ip)

            endif

          enddo
        enddo

      enddo
    enddo

    VM = 1.12*VM
    wm = INT(VM+0.5)
    wm = 250

    write(6,*) VM,wm

    do it=1,ntheta
      do ip=1,nphi

        k=k+1
        pt=INT((100*it+ip)*100+EBEAM)

        write(58,14) pt
        write(6,*) k,EBEAM,it,ip

        MAX(it,ip) = tension(1,1,it,ip)

        do ia=1,2
          do ib=1,3
```

```
          if (tension(ia,ib,it,ip).gt.MAX(it,ip)) then
            MAX(it,ip) = tension(ia,ib,it,ip)
          endif
        enddo
      enddo
      MAX(it,ip) = MAX(it,ip)*1.1
      VMAX(it,ip) = INT(MAX(it,ip)+0.5)
      do i=1,2
        do j=1,3
          DB(i,j,it,ip) =99
          de(i,j,it,ip) =99
          if (tension(i,j,it,ip).le.0.) then
            write(6,*) i,j,it,ip,tension(i,j,it,ip)
          endif
          if (tension(i,j,it,ip).gt.0.) then
            DB(i,j,it,ip) = INT(20*(ALOG10(MAX(it,ip)
     +                      /tension(i,j,it,ip)))+0.5)
            de(i,j,it,ip) =INT(20*(ALOG10((wm)
     +                      /tension(i,j,it,ip)))+0.5)
          endif
          if (DB(i,j,it,ip).gt.48.or.DB(i,j,it,ip).eq.99) then
            write(6,*) i,j,it,ip,de(i,j,it,ip)
            DB(i,j,it,ip) = 48
          endif
          if (de(i,j,it,ip).gt.48.or.de(i,j,it,ip).eq.99) then
            write(6,*) i,j,it,ip,de(i,j,it,ip)
            write(62,*) pt,de(1,1,it,ip),
     +                  de(1,2,it,ip),de(1,3,it,ip),
     +                  de(2,1,it,ip),de(2,2,it,ip),
     +                  de(2,3,it,ip),tcanal(1,it,ip),
     +                  tcanal(2,it,ip),tvoie(1,it,ip)
            de(i,j,it,ip) = 48
          endif
        enddo
      enddo
      phimin=wphi(1)+(ip-1)*phistep
      phimax=wphi(1)+(ip)*phistep
      themin=wtheta(1)+(it-1)*thetastep
      themax=wtheta(1)+(it)*thetastep
      write(pthe,3)themin,themax
      write(pphi,4)phimin,phimax
      write(56,5) pthe
      write(56,5) pphi
      write(57,5) pthe
      write(57,5) pphi
      write(56,6) energy(1,1,it,ip),energy(1,2,it,ip),
     +            energy(1,3,it,ip),energy(2,1,it,ip),
     +            energy(2,2,it,ip),energy(2,3,it,ip)
      write(57,7) tension(1,1,it,ip),tension(1,2,it,ip),
     +            tension(1,3,it,ip),tension(2,1,it,ip),
     +            tension(2,2,it,ip),tension(2,3,it,ip)
      write(56,8) ista(1,1,it,ip),ista(1,2,it,ip),
     +            ista(1,3,it,ip),ista(2,1,it,ip),
```

```
     +                 ista(2,2,it,ip),ista(2,3,it,ip)

           write(57,8) ista(1,1,it,ip),ista(1,2,it,ip),
     +                 ista(1,3,it,ip),ista(2,1,it,ip),
     +                 ista(2,2,it,ip),ista(2,3,it,ip)


           write(56,9) canal(1,it,ip),canal(2,it,ip)
           write(56,10) voie(1,it,ip)

           write(56,11) s_effi(it,ip)
           write(56,12) sc_effi(it,ip)
           write(56,100)

           write(57,9) tcanal(1,it,ip),tcanal(2,it,ip)
           write(57,10) tvoie(1,it,ip)

           write(58,15) MAX(it,ip),tension(1,1,it,ip),
     +                 tension(1,2,it,ip),tension(1,3,it,ip),
     +                 tension(2,1,it,ip),tension(2,2,it,ip),
     +                 tension(2,3,it,ip),tcanal(1,it,ip),
     +                 tcanal(2,it,ip),tvoie(1,it,ip)

           tcan(1,it,ip) = INT(tcanal(1,it,ip)+0.5)
           tcan(2,it,ip) = INT(tcanal(2,it,ip)+0.5)
           tv(1,it,ip) = INT(tvoie(1,it,ip)+0.5)

           write(59,16) pt,VMAX(it,ip),DB(1,1,it,ip),
     +                 DB(1,2,it,ip),DB(1,3,it,ip),
     +                 DB(2,1,it,ip),DB(2,2,it,ip),
     +                 DB(2,3,it,ip),tcan(1,it,ip),
     +                 tcan(2,it,ip),tv(1,it,ip)

           write(61,17) pt,wm,de(1,1,it,ip),
     +                 de(1,2,it,ip),de(1,3,it,ip),
     +                 de(2,1,it,ip),de(2,2,it,ip),
     +                 de(2,3,it,ip),VMIN,tv(1,it,ip),
     +                 ONOFF1,ONOFF2

           write(57,11) s_effi(it,ip)
           write(57,12) sc_effi(it,ip)
           write(57,100)
           write(58,100)

         enddo
       enddo


   1   format('S',i2,'L',i1)
   2   format(' point                ',6(5x,a5))
   3   format(' theta',f5.2,'-',f5.2)
   4   format(' phi  ',f5.2,'-',f5.2)
   5   format(a17)
   6   format(' deposite energy',4x,6(1x,f8.3,1x))
   7   format(' tension',12x,6(1x,f8.3,1x))
   8   format(' nber of evts',6x,6(1x,i8,1x))
   9   format(' S1tot-S2tot',8x,11x,F8.3,1x,21x,F8.3,1x)
  10   format(' S1+S2 ',39x,F8.3,1x)
  11   format(' S efficiency %',32x,F6.2,1x)
  12   format(' S+C efficiency %',30x,F6.2,1x)
  13   format(' point',2x,'S1',6x,'S2',6x,'S3',6x,'S4',6x,
     +          'S5',6x,'S6',6x,'C1',6x,'C2',6x,'C',6x)
  14   format(I6)
  15   format(10(F7.2,1x))
  16   format((I6,3x),(I4,6x),6(I3,3x),3(4x,I3))
  17   format(' ',I6,' ',11(I3,' '))
 100   format(80('-'))

       write(6,*) ' '
       write(6,*) ' My job is finished..Do yours..Bye and Good Luck !'
       write(6,*) ' '

       return
       end

*********************************************************************

       SUBROUTINE vzero(x,n)

*********************************************************************

       dimension x(30000)

       do i=1,n

         x(i)=0
```

```
      enddo

      return

      end
*************************************************************************

      SUBROUTINE bdata

*************************************************************************

      PARAMETER (nthebin=25,nphibin=20)
*
      REAL wphi(2)
      REAL wtheta(2)
      REAL energy(16,3,nthebin,nphibin)
      REAL phistep
      REAL thetastep
*
      INTEGER nphi
      INTEGER ntheta
      INTEGER ista(16,3,nthebin,nphibin)
*
      LOGICAL debug
      LOGICAL debug10
*
      COMMON/WORK/wphi,wtheta,energy,
     +            phistep,thetastep,
     +            nphi,ntheta,ssigma,
     +            ista,debug,debug10
*


      nphi =  2
      wphi(1)= 7.5
      wphi(2)=30.

      ntheta =  2
      wtheta(1)= 3.9
      wtheta(2)=8.7

      return

      end
*************************************************************************

      SUBROUTINE interactif

*************************************************************************

      PARAMETER (nthebin=25,nphibin=20)
*
      REAL wphi(2)
      REAL wtheta(2)
      REAL energy(16,3,nthebin,nphibin)
      REAL phistep
      REAL thetastep
*
      INTEGER nphi
      INTEGER ntheta
      INTEGER ista(16,3,nthebin,nphibin)
*
      LOGICAL debug
      LOGICAL debug10
*
      COMMON/WORK/wphi,wtheta,energy,
     +            phistep,thetastep,
     +            nphi,ntheta,ssigma,
     +            ista,debug,debug10
*
      lpass1=.true.

      write(6,*) ' '
      write(6,*) 'Please,enter the number of cuts in phi (<5)'
      read(5,*) nphi
      write(6,*) ' '
      write(6,*) 'Please,enter the number of cuts in theta (<5)'
      read(5,*) ntheta
      write(6,*) ' '
      write(6,*) 'Please,enter the work-phi min (0<wphim<7.5)'
      read(5,*) wphi(1)
      write(6,*) ' '
      write(6,*) 'Please,enter the work-phi max (wphim<wphi<30)'
```

```
      read(5,*) wphi(2)
      write(6,*) ' '
      write(6,*) 'Please,enter the work-theta min (3.9<wthetam<8.7)'
      read(5,*) wtheta(1)
      write(6,*) ' '
      write(6,*) 'Please,enter the work-theta max (3.9<wtheta<8.7)'
      read(5,*) wtheta(2)
      write(6,*) ' '

      return
      end
```

ANNEXE 7

Tableau de correspondance des dB avec le facteur d'atténuation

DB	ATTENUATION OU GAIN EN TENSION	
1	1.1	1.122019
2	1.3	1.258926
3	1.4	1.412538
4	1.6	1.584893
5	1.8	1.778279
6	2	1.995262
7	2.2	2.238721
8	2.5	2.511886
9	2.8	2.818383
10	3.2	3.162278
11	3.5	3.548134
12	4	3.981072
13	4.5	4.466836
14	5	5.011873
15	5.6	5.623413
16	6.3	6.309573
17	7.1	7.079458
18	7.9	7.943283
19	8.899999	8.91251
20	10	10
21	11.2	11.22018
22	12.6	12.58926
23	14.1	14.12537
24	15.8	15.84893
25	17.8	17.7828
26	20	19.95262
27	22.4	22.38721
28	25.1	25.11886
29	28.2	28.18383
30	31.6	31.62278
31	35.5	35.48134
32	39.8	39.81072
33	44.7	44.66836
34	50.1	50.11873
35	56.2	56.23413
36	63.1	63.09574
37	70.8	70.79458
38	79.4	79.43284
39	89.1	89.12513
40	100	100
41	112.2	112.2018
42	125.9	125.8925
43	141.3	141.2538
44	158.5	158.4893
45	177.8	177.828
46	199.5	199.5262
47	223.9	223.872
48	251.2	251.1887
49	281.8	281.8382
50	316.2	316.2279
51	354.8	354.8134

ANNEXE 8

Fichiers des données 5, 20, 30, 45, 60 et 90 GeV

10105	250	35	30	39	48	48	48	0	17	0	0
10205	250	48	31	38	35	45	48	0	18	0	0
10305	250	48	45	39	34	31	47	0	18	0	0
20105	250	32	28	37	48	48	48	0	22	0	0
20205	250	48	29	35	32	44	48	0	23	0	0
20305	250	48	44	37	31	28	47	0	23	0	0
30105	250	32	30	41	48	48	48	0	18	0	0
30205	250	48	31	40	32	46	48	0	19	0	0
30305	250	48	46	41	32	31	48	0	18	0	0

10115	250	30	21	27	48	48	48	0	44	0	0
10215	250	48	21	26	30	37	48	0	47	0	0
10315	250	48	37	27	29	21	39	0	48	0	0
20115	250	27	18	24	47	48	48	0	61	0	0
20215	250	48	19	23	27	37	48	0	63	0	0
20315	250	48	37	25	27	19	38	0	63	0	0
30115	250	27	21	29	48	48	48	0	44	0	0
30215	250	48	22	29	27	41	48	0	44	0	0
30315	250	48	41	29	27	21	43	0	47	0	0

10130	250	27	15	20	48	48	48	0	85	0	0
10230	250	48	16	19	28	33	45	0	87	0	0
10330	250	48	34	19	27	16	33	0	90	0	0
20130	250	24	13	17	47	48	48	0	114	0	0
20230	250	47	13	16	24	32	46	0	119	0	0
20330	250	48	33	17	24	13	32	0	119	0	0
30130	250	24	15	22	47	48	48	0	82	0	0
30230	250	48	16	22	24	37	48	0	82	0	0
30330	250	48	36	23	24	16	38	0	83	0	0

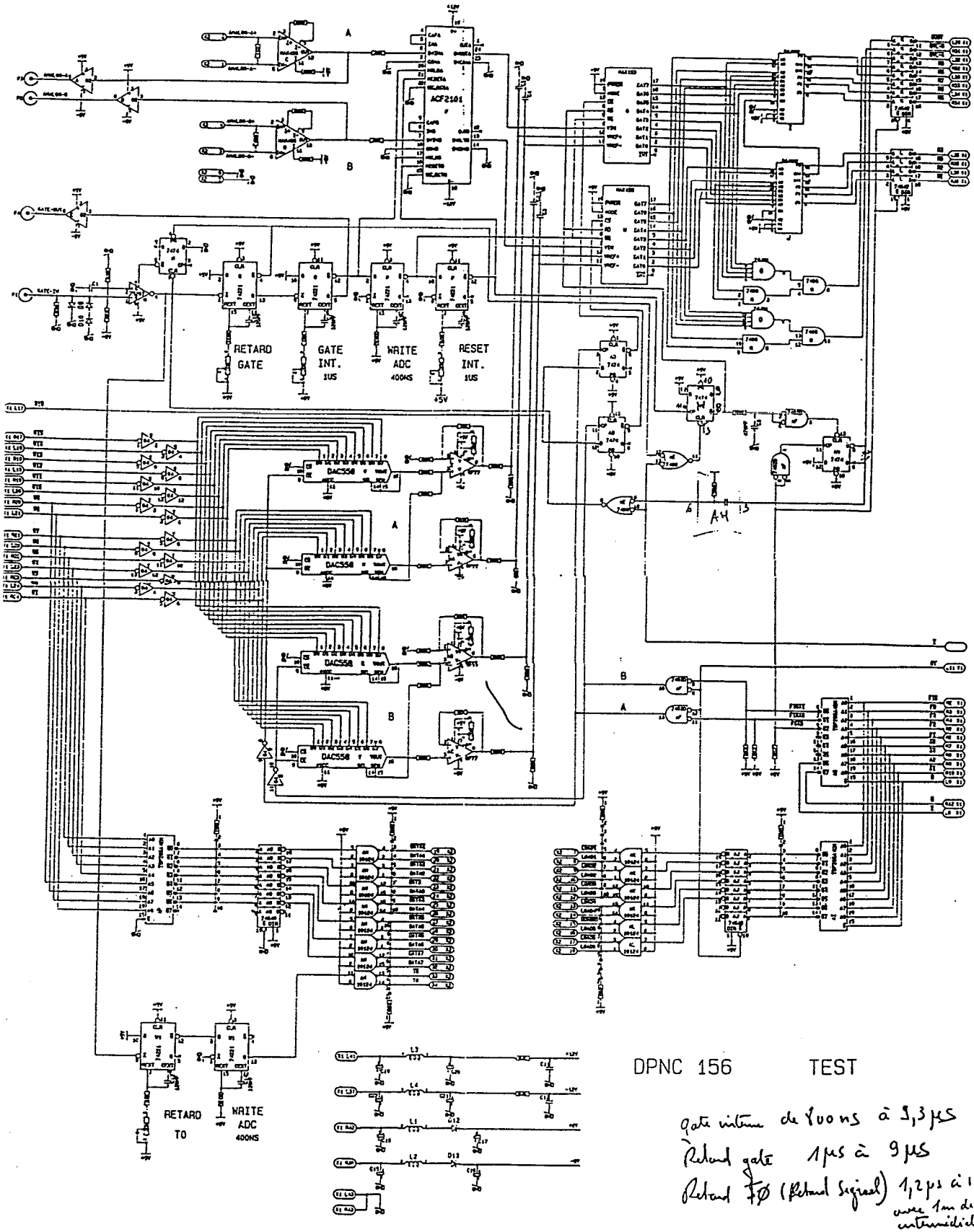
10145	250	26	13	16	47	46	48	0	121	0	0
10245	250	47	13	14	26	31	42	0	129	0	0
10345	250	48	31	16	26	13	29	0	131	0	0
20145	250	22	9	13	47	48	48	0	165	0	0
20245	250	46	10	11	22	30	44	0	174	0	0
20345	250	48	30	13	22	10	28	0	174	0	0
30145	250	22	11	18	46	48	48	0	114	0	0
30245	250	48	13	17	22	34	48	0	119	0	0
30345	250	48	34	18	22	13	34	0	119	0	0

10160	250	25	10	13	46	45	47	0	157	0	0
10260	250	47	11	12	25	29	40	0	167	0	0
10360	250	48	29	13	25	10	27	0	167	0	0
20160	250	21	7	10	45	47	48	0	216	0	0
20260	250	45	8	9	21	28	42	0	227	0	0
20360	250	48	28	10	21	8	26	0	227	0	0
30160	250	21	10	16	47	48	48	0	144	0	0
30260	250	46	10	15	21	32	47	0	153	0	0
30360	250	48	33	16	21	10	32	0	152	0	0

10190	250	23	7	8	46	43	45	0	235	0	0
10290	250	46	7	8	24	27	37	0	244	0	0
10390	250	48	26	9	23	7	23	0	248	0	0
20190	250	19	4	6	45	44	48	0	314	0	0
20290	250	45	4	5	19	26	38	0	331	0	0
20390	250	48	26	6	19	5	22	0	331	0	0
30190	250	20	7	12	46	48	48	0	206	0	0
30290	250	44	8	11	20	29	44	0	213	0	0
30390	250	48	30	12	20	7	29	0	217	0	0
10190	250	23	7	8	46	43	45	0	235	0	0

ANNEXE 9

Carte de sommation de deux canaux pour les tests



DPNC 156 TEST

gate interne de 400 ns à 9,3 µs

Retard gate 1 µs à 9 µs

Retard FØ (Retard Signal) 1,2 µs à 10 µs avec 1m de câble intermédiaire

estimation avec les 40m de câble
1,4 µs à 9,4 µs

ANNEXE 10

Comparaison des décibels mesurés (amplitude signal) avec la théorie

Attenuation (dB)	V1 (mV)	V2 (mV)	V3 (mV)	V4 (mV)	V5 (mV)	V6 (mV)	Vtheorie (mV)	ecart max (pourcent)
1	425	469	475	469	419	425	477	0.878406709
10	150	150	150	150	150	150	164	0.914634146
20	47	48	48	48	48	48	52.5	0.914285714
30	15	15	15	15	15	15	16.6	0.903614458
35	8.5	8.5	8.5	8.5	8.5	8.5	9.34	0.91006424

V = Amplitude du signal de depart = 525 mV

V1...V6 = Amplitude des 6 canaux d'attenuateurs mesuree selon l'attenuation portee a V

Vtheorie = Amplitude que l'on aurait du mesurer si les attenuateurs etaient parfaits

ANNEXE 11

- Programme : 'initialisation.for'
- Programme : 'init-talr.pas'
- Programme : 'matrix.dat'
- Programme : 'rout.for'

```
      PROGRAM initialisation

************************************************
*                                              *
* This program is a preliminary one in order to *
* see how to make the different parts of the    *
* initialisation of the ALR TRIGGER.            *
*                                              *
*   D. SCIARRINO  12-08-94                     *
*                                              *
************************************************

      IMPLICIT NONE

      INTEGER NLINE,i,j

      PARAMETER (NLINE = 32)

      REAL SUMPED(NLINE,2)

      COMMON/VAR/SUMPED

      i=0
      j=0

      DO i=1,NLINE

        DO j=1,2

          SUMPED(i,j)=0

        ENDDO

      ENDDO

      CALL run_mode


      END

**************************************

      SUBROUTINE run_mode

**************************************

      INTEGER run_condition
      INTEGER NLINE

      PARAMETER(NLINE = 32)

      REAL SUMPED(NLINE,2)

      COMMON/VAR/SUMPED
```



```
      CHARACTER test


      WRITE(6,*) ' '
      WRITE(6,*) ' 1 : TEST RUN '
      WRITE(6,*) ' '
      WRITE(6,*) ' 2 : PEDESTAL RUN '
      WRITE(6,*) ' '
      WRITE(6,*) ' 3 : NORMAL RUN '
      WRITE(6,*) ' '
      WRITE(6,*) ' 0 : NO RUN '
      WRITE(6,*) ' '
      WRITE(6,*) ' '
      WRITE(6,*) '     CHOOSE ONE OF THESE RUN MODES '
      WRITE(6,*) ' '
```

```
      READ(5,*) run_condition
      WRITE(6,*) ' '

      IF (run_condition.eq.1) then
        WRITE(6,*) ' 1 : TEST RUN '
      ENDIF
      IF (run_condition.eq.2) then
      write(6,*) 'call pedestal '
        CALL pedestal
       write(6,*) 'called pedestal '

        OPEN(unit=10,file='init_talr.log',status='old',
     +       access='sequential')
 5      READ(10,*, END=10)
        GOTO 5
10      WRITE(10,*) 'SALUT DU FORTRAN'
        close (10)
          endif
       IF (run_condition.eq.3) then
        CALL prepare_run
        OPEN(unit=10,file='test.dat',status='unknown',
     +    access='sequential')
        READ(10,*)
        WRITE(10,*) 'SALUT DU FORTRAN'

        CLOSE(10)
```



```
        ENDIF
        IF (run_condition.eq.0) then
* halt from VAX
          WRITE(6,*) ' 0 : NO RUN '
        ENDIF
        RETURN
        END
*******************************************
        SUBROUTINE prepare_run
*******************************************
        WRITE(6,*) ' 3 : RUN '

        CALL read_sumped
        CALL check_sumoverflow
        CALL find_ped

        RETURN
        END

*******************************************
```

```
        SUBROUTINE read_sumped
*****************************************
* SUMPED  : are the sum of two chanels : 2*16 sumped in the file
* SUMOVER : are the OR of the overflow of 2 sumated chanels
*           SUMOVER = 1 if at least one of the ADC is in overflow

        INTEGER NLINE
        INTEGER n

        PARAMETER (NLINE = 32)

        REAL SUMPED(NLINE,2)
        REAL SUMOVER(NLINE)

        COMMON/VAR/SUMPED
```



```
*       OPEN(access='direct',unit=10,recl=1024,
*     +       form='formatted', file='ped.dat',status='unknown')

        OPEN(access='sequential',unit=10,
     +       file='sumped.dat',status='unknown')

        DO n=1,NLINE

          READ(10,*)(SUMPED(n,j),j=1,2)
*          WRITE(6,*)(SUMPED(n,j),j=1,2)

        ENDDO

        CLOSE(10)

        WRITE(6,*) ' '
        WRITE(6,*) 'Hi from sumped'
        WRITE(6,*) ' '


        RETURN

        END


*****************************************
        SUBROUTINE check_sumoverflow
*****************************************

        INTEGER NLINE

        PARAMETER(NLINE = 32)

        REAL SUMOVER(NLINE)

        REAL SUMPED(NLINE,2)

        COMMON/VAR/SUMPED

        DO j=1,NLINE
*         WRITE(6,*)'SUMPED =',(SUMPED(j,i),i=1,2)

          SUMOVER(j) = SUMPED(j,2)

          WRITE(6,*) j, SUMOVER(j)

        ENDDO

        DO i=2,15
```

```
          IF (SUMOVER(i).eq.1.) then
            IF ((SUMOVER(i-1).ne.1.).
     +          and.(SUMOVER(i+1).ne.1.)) then

             WRITE(6,*)' I had some pbs while reading
     + the overflow ',i,' : they are
     + incredible...PLEASE CHECK THE ADCs!'

*             WRITE(6,*)i
            ENDIF
          ENDIF

        ENDDO

        IF (SUMOVER(16).eq.1.)then

          IF((SUMOVER(15).ne.1.).
     +   and.(SUMOVER(1).ne.1.))then


          WRITE(6,*)' I had some pbs while reading
     + the overflow 16 : they are
     + incredible...PLEASE CHECK THE ADCs!'

          ENDIF
        ENDIF

        DO i=18,31

          IF (SUMOVER(i).eq.1.)then

            IF((SUMOVER(i-1).ne.1.).
     +          and.(SUMOVER(i+1).ne.1.)) then


            WRITE(6,*)' I had some pbs while reading
     +the overflow',i,': they are
     +incredible...PLEASE CHECK THE ADCs!'

           ENDIF
          ENDIF

        ENDDO

        IF (SUMOVER(32).eq.1.)then

          IF((SUMOVER(31).ne.1.).
     +          and.(SUMOVER(17).ne.1.)) then


          WRITE(6,*)' I had some pbs while reading
     + the overflow 32 : they are
     + incredible...PLEASE CHECK THE ADCs!'

          ENDIF
         ENDIF
```



```
        WRITE(6,*) ' '
        WRITE(6,*) 'No pb from check_overflow'
        WRITE(6,*) ' '

 100    FORMAT(x,i2,x,i1)

        RETURN
```

```
        END


*****************************************

        SUBROUTINE find_ped

*****************************************


        DIMENSION R(50)

        REAL  A(50,16), B(50,16), A2(50,16), B2(50,16)
        INTEGER i,j,k,l,m,n
        INTEGER ARRAY_DIM

        PARAMETER(ARRAY_DIM=16)

        OPEN(unit=10,file='matrix.dat',status='unknown')
        OPEN(unit=11,file='sumped.dat',status='unknown')

        IFAIL=0
        IFAIL2=0
        i=0
        j=0
        k=0
        l=0
        m=0
        n=0


        DO i=1,50

          DO j=1,16

            A(i,j)=0
            B(i,j)=0
            A2(i,j)=0
            B2(i,j)=0

          ENDDO

        ENDDO
```



```
        DO i=1,ARRAY_DIM

            READ(10,*,end=5) (A(i,j),j=1,ARRAY_DIM)
*            WRITE(6,*)   (A(i,j),j=1,ARRAY_DIM)

        ENDDO

        DO i=1,ARRAY_DIM

          DO j=1,ARRAY_DIM

            A2(i,j)=A(i,j)
*            WRITE(6,*) A2(i,j)

          ENDDO

        ENDDO

  5     PRINT *,'EOF'


        CLOSE(10)

        DO k=1,32

          READ(11,110) B(k,1)
*           WRITE(6,111) k,B(k,1)

          If (k.ge.17) then
```

```
            B2((k-16),1) = B(k,1)
*            WRITE(6,112) (k-16), B2((k-16),1)
            B(k,1) = 0
           ENDIF
           WRITE(6,*) k,B(k,1), B2((k-16),1)
         ENDDO
         CLOSE(11)
         CALL REQINV(ARRAY_DIM,A,50,R,IFAIL,1,B)
         WRITE(6,*)'IFAIL = ',IFAIL
         CALL REQINV(ARRAY_DIM,A2,50,R,IFAIL2,1,B2)
         WRITE(6,*)'IFAIL2= ',IFAIL2

         DO l=1,ARRAY_DIM
           WRITE(6,115) l, B(l,1), l, B2(l,1)
```



```
*          WRITE(6,112) l, B2(l,1)
         ENDDO
*         DO m=1,ARRAY_DIM
*           DO n=1,ARRAY_DIM
*             WRITE(6,113) m, n, a(m,n)
*             WRITE(6,114) m, n, a2(m,n)
*           ENDDO
*         ENDDO

   110   FORMAT(F4.0)
   111   FORMAT(' B (',i2,',1) = ',F5.0)
   112   FORMAT(' B2(',i2,',1) = ',F5.0)
   113   FORMAT(' A (',i2,',',i2,') = ',F3.0)
   114   FORMAT(' A2(',i2,',',i2,') = ',F3.0)
   115   FORMAT(' B (',i2,',1) = ',F5.0,' B2(',i2,',1) = ',F5.0)
         RETURN
         END
```

```
[INHERIT (
          'JT$PGM:[LIBS.GEFBLIB]CAMdef.PEN',
          'JT$PGM:[LIBS.utilities]node.pen'
         )]
module  INIT_Talr(output);

const
 branch=4;
 crate=6;
 slot_decision=9;

type

{side = cote rb24a a rb26d : huit modules}
{vr   = dans les huit modules il y a 4 ADC, 1 pour chaque canal>8 fois 4 DAQ}
{stat = etat de l'ADC : ON ou OFF > 8 fois 4  etats }
{decision = dans la logique : single tag, double tag, ET logique avec TEC}

  side=(rb24a,rb24b,rb24c,rb24d,rb26a,rb26b,rb26c,rb26d);
  vr=array[rb24a..rb26d,1..4]of unsigned;
  tab=array[1..32]of integer;
  stat=array[rb24a..rb26d,1..4]of boolean;
  decision=array[1..3]of unsigned;
  fourint=array[side]of unsigned;

var
 slot:fourint;

 test,inp,t2,t3,t4,t5 : text;
 t1 :string(100);
[global]procedure init(vrefm:vr;vrefp,switch,add_sub:vr;on:stat;
  var bt,thBB,thGG,thST,psBB,psST,onoff:unsigned;var ok:boolean);

var
 patt:unsigned;
 btdac:unsigned;
 thBB_ST:unsigned;
 psBB_ST:unsigned;
 i:integer;
 j:side;

const
 machine='VXL3j0';

begin

 ok:= true;

 OPEN(test,'init_talr.log');
 REWRITE(test);

{ CAMAC ne marche que sur la machine vxl3j0 }

 if not is_node(machine) then begin
  writeln(' this program must run on ',machine);
```



```
  ok := false;
{  halt;}
 end;

 init_camac;

{ rb24 modules CAMAC }

 slot[rb24a]:=1;
 slot[rb24b]:=2;
 slot[rb24c]:=3;
 slot[rb24d]:=4;

{ rb26 modules CAMAC }
```

```
slot[rb26a]:=5;
slot[rb26b]:=6;
slot[rb26c]:=7;
slot[rb26d]:=8;

if not testonline(branch,crate) then begin
  writeln(' not online branch ',branch,' crate ',crate);
  ok:=false;
  halt;
end;

{ ecrit les vref- et vref+}

for j:=rb24a to rb26d do begin
 for i:=1 to 4 do begin
  if (vrefm[j,i]>127) then begin
    writeln('not good vrefm for side ',j,' number ',i,' value ',vrefm[j,i]);
    ok:=false;
    vrefm[j,i]:=0;
    WRITELN(test,'vrefm was wrong(=',vrefm[j,i],')so I put it to 0');
    WRITELN(test,'ok = false');
  end;

  if (vrefp[j,i]>255) then begin
    writeln('not good vrefp for side ',j,' number ',i,' value ',vrefp[j,i]);
    ok:=false;
    vrefp[j,i]:=255;
    WRITELN(test,'vrefp was wrong(=',vrefp[j,i],')so I put it to 255');
    WRITELN(test,'ok = false');
  end;

{ 2**7 pour commencer a 128 pour vrefp }

  patt:=vrefm[j,i]+2**7*vrefp[j,i];

{ 2**15 pour le 16 eme bit }

  if (on[j,i]) then patt:=patt+2**16;

{ ecrit vref- sur 7 bits, vref+ sur 8 bits, et ONOFF sur le dernier }
```



```
   camac(branch,crate,slot[j],i,16,patt);

   if not campass.xcamac then begin
     writeln('no x camac for side ',j,' number ',i);
     ok:=false;
     WRITELN(test,'no x camac for side ',j,' number ',i);
     WRITELN(test,'ok = false');

   end;
   btdac := bt + 2**8*switch[j,i] + 2**12*add_sub[j,i];
  end;

 { seuil du discri pour le "bunch tagging" = 1 DAQ }

  camac(branch,crate,slot[j],4,16,btdac);

  if not campass.xcamac then begin
    writeln('no x camac for btdac side ',j);
    ok:=false;

     WRITELN(test,'no x camac for btdac side ',j);
     WRITELN(test,'ok = false');

  end;
 end;

 { 9 eme module = module de decision, on ecrit les seuils des 3 differents trig

  thBB_ST := thBB + 2**8 * thST;
  thGG := thGG + 2**8 * onoff;
  psBB_ST := psBB + 2**8 * psST;

  CAMAC(branch,crate,slot_decision,0,16,thBB_ST);
```

```
 for i:=rb24a to rb26d do begin
```



```
  for j:=1 to 4 do begin
   vrefp[i,j]:=255;
   vrefm[i,j]:=0;
   switch[i,j]:= 0;
   add_sub[i,j]:=0;
   on[i,j]:=true;
  end;
 end;

 open(inp,'alr_ini.dat',history:=old);
 reset(inp);
 readln(inp,t1);
 writeln('t1 = ',t1);
 readln(inp,th1);
  writeln('th1= ',th1);
 readln(inp,thbb_st,thgg,psbb_st);
  writeln('thbb_st,thgg,psbb_st = ',thbb_st,thgg,psbb_st);
 readln(inp,modu);
   writeln('modu =  ',modu);

 for k:=1 to 32 do begin

   readln(inp,a[k],b[k],c[k],d[k]);
   writeln(a[k],b[k],c[k],d[k]);

 end;
 readln(inp,module);
 writeln('module =',module);
  for k:=1 to 8 do begin

   readln(inp,e[k],f[k]);
   writeln(e[k],f[k]);
  end;
close(inp);


 init(vrefm,vrefp,switch,add_sub,ON,bt,thBB,thGG,thST,psBB,psST,onoff,ok);
 if ok then writeln('AHOUTCH, it worked !!') else writeln('Sniff, it failed..');



end;
end.
```

```
 if not campass.xcamac then begin

  writeln('no x camac for thBB_ST ');
  ok:=false;

  WRITELN(test,'no x camac for thBB_ST ');
  WRITELN(test,'ok = false');

 end;

CAMAC(branch,crate,slot_decision,1,16,thGG);

 if not campass.xcamac then begin

  writeln('no x camac for thGG ');
  ok:=false;

  WRITELN(test,'no x camac for thGG ');
  WRITELN(test,'ok = false');
```



```
  end;


 CAMAC(branch,crate,slot_decision,2,16,psBB_ST);

  if not campass.xcamac then begin

   writeln('no x camac for psBB_ST');
   ok:=false;

   WRITELN(test,'no x camac for psBB_ST');
   WRITELN(test,'ok = false');

  end;


  IF OK then WRITELN(test,'PASS')
  ELSE WRITELN(test,'FAILED');
  CLOSE(test);
end;

[global]procedure pedestal;
var
 b: tab;
 a,c,d,e,f:tab;
 vrefm:vr;
 vrefp:vr;
 on:stat;
 bt:unsigned;
 onoff:unsigned;
 thbb_st : unsigned;
 psbb_st : unsigned;
 switch : vr;
 add_sub : vr;
 thBB:unsigned;
 thST:unsigned;
 thGG:unsigned;
 psBB:unsigned;
 psST:unsigned;
 i:side;
 j,k:integer;
 ok:boolean;

 th1,modu,module:string(100);

 begin

  thBB := 0;
  thST := 0;
  thGG := 0;
  psBB := 0;
  psST := 0;
```

1.	-1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	1.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	1.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	1.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	1.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	1.	1.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	1.	1.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	1.	1.	0.	0.	0.	0.	0.	0
0.	0.	0.	0.	0.	0.	0.	0.	1.	1.	0.	0.	0.	0.	0
0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	1.	0.	0.	0.	0
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	1.	0.	0.	0
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	1.	0.	0
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	1.	0
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	1
1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1

```
*****************************************

      program rout
c special acquistion camac pour TALR
*****************************************

      INTEGER   month
      INTEGER   day
      INTEGER   year

      DIMENSION time(8)

      INTEGER   i
      INTEGER   nline

      PARAMETER (nline = 16)
      PARAMETER (NWPAWC = 200000)

      INTEGER*2 I_alr(nline,2)
      INTEGER*2 data(nline,2)
      INTEGER   over(nline,2)
      INTEGER   bunchlet(nline,2)

      COMMON/VAR/I_alr,data,over,bunchlet
      COMMON/PAWC/H(NWPAWC)

      CALL IDATE(month,day,year)

      WRITE(6,*) month,day,year

C      CALL TIME(time)

      OPEN(UNIT=10,FILE='sumped.dat',STATUS='new')

      CALL HLIMIT(NWPAWC)

      CALL HROPEN(1,'example','myhisto.hsto','N',1024,Istat)

      CALL hbook2(100,'evts.vs.sum_ped.vs.chanel(RB26)',
     +            16,1.,17.,767,-255.,512.,0.)

      CALL hbook2(101,'nb_evts.vs.sum_pedestal.vs.chanel(RB24)',
     +            16,1.,17.,767,-255.,512.,0.)

      CALL hbook2(102,'nb_evts.vs.pedestal.vs.chanel(RB26)',
     +            16,1.,17.,256,0.,256.,0.)

      CALL hbook2(103,'nb_evts.vs.pedestal.vs.chanel(RB24)',
     +            16,1.,17.,256,0.,256.,0.)


      CALL reading

      CALL fill_file

      CLOSE(10)

      CALL prepare_run

      CALL HCDIR('//example',' ')

      CALL HROUT(0,ICYCLE,' ')

      CALL HREND('example')

      CLOSE(1)

      END

*********************************************

      SUBROUTINE reading

*********************************************

      INTEGER   nline

      PARAMETER (nline = 16)

      INTEGER*2 I_alr(nline,2)
      INTEGER*2 data(nline,2)
      INTEGER   over(nline,2)
      INTEGER   bunchlet(nline,2)
```

```
        ENDDO

        DO j= 1,2

          IF (btest(I_alr(1,j),8)) THEN

            data(1,j) = (NOT(data(1,j)).and.'1ff'x) + 1

            WRITE(6,*)' '
            WRITE(6,*)'cote           = ',j
            WRITE(6,*)'(notdata) + 1 = ',data(1,j)

            data(1,j) = - data(1,j)

            WRITE(6,*)'data           = ',data(1,j)
            WRITE(6,*)' '

          ENDIF

        ENDDO

        DO i = 1,nline
          write(6,*) data(i,2)
          CALL hfill(100,float(i),float(data(i,1)),1.)
          CALL hfill(101,float(i),float(data(i,2)),1.)

        ENDDO

  100   FORMAT(' data(',i2,',',i2,')=',i8)
  110   FORMAT(' over(',i2,',',i2,')=',i1)
  120   FORMAT(' bunchlet(',i2,',',i2,')=',i1)


        RETURN

        END

*****************************************

        SUBROUTINE fill_file

*****************************************

        INTEGER   nline
        INTEGER   line

        PARAMETER (nline = 16)
        PARAMETER (line  = 32)

        INTEGER*2 I_alr(nline,2)
        INTEGER*2 data(nline,2)
        INTEGER   over(nline,2)
        INTEGER   bunchlet(nline,2)
        INTEGER   sumped(line)
        INTEGER   sumover(line)

        COMMON/VAR/I_alr,data,over,bunchlet

        DO i = 1, nline

          sumped(i)        = data(i,1)
          sumover(i)       = over(i,1)
          sumped(i+16)     = data(i,2)
          sumover(i+16)    = over(i,2)

        ENDDO

        DO j = 1, line

          WRITE(10,130) sumped(j), sumover(j)

        ENDDO

  130   FORMAT(x,i4,2x,i1)

        RETURN

        END

*****************************************

        SUBROUTINE prepare_run

*****************************************
```

```
      DO i=2,15

        IF (SUMOVER(i).eq.1.) then
          IF ((SUMOVER(i-1).ne.1.).
     +        and.(SUMOVER(i+1).ne.1.))  then

            WRITE(6,*)' I had some pbs while reading
     + the overflow ',i,' : they are
     + incredible...PLEASE CHECK THE ADCs!'

*             WRITE(6,*)i
          ENDIF
        ENDIF

      ENDDO

      IF (SUMOVER(16).eq.1.)then

        IF((SUMOVER(15).ne.1.).
     +   and.(SUMOVER(1).ne.1.))then
          WRITE(6,*)' I had some pbs while reading
     + the overflow 16 : they are
     + incredible...PLEASE CHECK THE ADCs!'

        ENDIF

      ENDIF

      DO i=18,31

        IF (SUMOVER(i).eq.1.)then

          IF((SUMOVER(i-1).ne.1.).
     +        and.(SUMOVER(i+1).ne.1.))  then


            WRITE(6,*)' I had some pbs while reading
     +the overflow',i,': they are
     +incredible...PLEASE CHECK THE ADCs!'

          ENDIF

        ENDIF

      ENDDO

      IF (SUMOVER(32).eq.1.)then

        IF((SUMOVER(31).ne.1.).
     +         and.(SUMOVER(17).ne.1.)) then

          WRITE(6,*)' I had some pbs while reading
     + the overflow 32 : they are
     + incredible...PLEASE CHECK THE ADCs!'

        ENDIF

       ENDIF

      WRITE(6,*) ' '
      WRITE(6,*) 'Overflows checked'
      WRITE(6,*) ' '

100   FORMAT(x,i2,x,i1)

      RETURN

      END


*******************************************

      SUBROUTINE find_ped

*******************************************


      DIMENSION R(50)

      REAL  A(50,16), B(50,16), A2(50,16), B2(50,16)
      INTEGER i,j,k,l,m,n
      INTEGER ARRAY_DIM

      PARAMETER(ARRAY_DIM=16)
```

```
          CALL hfill(103,float(l),B2(l,1),1.)
*          WRITE(6,112) l, B2(l,1)
        ENDDO
*        DO m=1,ARRAY_DIM
*          DO n=1,ARRAY_DIM
*            WRITE(6,113) m, n, a(m,n)
*            WRITE(6,114) m, n, a2(m,n)
*          ENDDO
*        ENDDO
  110   FORMAT(x,F4.0)
  111   FORMAT(' B (',i2,',1) = ',F7.2)
  112   FORMAT(' B2(',i2,',1) = ',F7.2)
  113   FORMAT(' A (',i2,',',i2,') = ',F3.0)
  114   FORMAT(' A2(',i2,',',i2,') = ',F3.0)
  115   FORMAT(' B (',i2,',1) = ',F7.2,' B2(',i2,',1) = ',F7.2)
        RETURN
        END
```