while the last 2 events have only 1 *bad* slice each.

The 12 events with 2 *bad* slices all have a difference between actual and expected energy sum equal to either +256 or −256, as can be approximately seen in the case of Fig. 10 for ASIC #2. This fact agrees with AW's recent explanation for the *bad* events [10]: that in these cases only the 4 upper bits (most significant bits) of the 12-bit energy sums are *late*. This is possible because the memories and data paths of the Cluster Finding Module (the ASIC circuit board) are only 8 bit wide.

The 2 events with just one *bad* slice each cannot be explained in this way. Their expected and actual energy sums are listed in Appendix A.5.

4 Summary

A summary of the most important findings in the data is given in table 6. Run number ranges are inclusive. The trigger channel *Group* labels are defined in Sec. 1. For example, the high pedestal rms in run 455 only concerns trigger cell (1,4), which belongs to the A1 group (channels Actively connected to ADC card #1).

Table 7 lists some hardware changes that may have caused some of the effects in the data.

Run(s)	*Group*	*Channel(s)*	*Effect*	*Sec.*
316–328		Some	Ped value peaks/dips	2.4
172–382±4	All	≈ all	Ped value unstable	2.4
455	A1	(1,4)	Pedestal rms high in beam cell	2.2.1
462	A1	(2,4)	Pedestal rms high in non-beam cell	2.2.1
563–576	A2,#3	All	Pedestal rms at higher & wider level	2.1,2.3
577–590	A2,#3	All	Pedestal rms at even higher level	2.3
591–618	A2,#3	All	Pedestal rms at highest level	2.3
591		Many	Pedestal value jumps to new level	2.4
607	A3	(5,3)	Pedestal rms high ⇐ *double event*	2.2.1
607	#2,#3	All	*Low frequency noise* up to ±10 GeV	2.2.2, 2.3
607	A1,A2(,A3)	All	*High frequency noise* (not large)	2.2.3
607	A#	All	*Cross-talk* up to ≈ 15 GeV	3.1
607	—	—	2/1000 unexplained *bad* energy sums	3.2

Table 6: Summary of effects in data

An increasing *low frequency noise* is found in the last runs, shifting the pedestal of some events by as much as 10 ADC counts (≈ 10 GeV) up or down. Additionally, a systematic *cross-talk* up to 17 counts (GeV) is found for the 300 GeV e^- in run 607.

The low frequency noise is confined to ADC cards #2 and #3, strongest on the latter. It apparently originated from the immediate vicinity of the ADC cards — e.g. from something in the rack in which they were mounted — since also some ADC-end terminated channels were affected.

The cross-talk is seen in all connected channels, with a strength generally decreasing with the distance to the beam cell channel. The origin of the cross-talk must be sought outside the ADC cards — before the signals arrive to them — since the ADC-end terminated channels are completely unaffected. Other authors find in earlier runs a cross-talk in unused channels that

Run(s)	*Group*	*Channel(s)*	*Hardware configuration*	*Sec.*
172–576	P#	(1–6,1)	Terminated in 50Ω at calorimeter end	1,2.3
3xx–3yy	All(?)	All(?)	RD3 high voltage scan, runs uncertain	2.4
zzz–381	A3	(5,5)	Lemo cable at fan-in output is damaged	2.4
vvv–387	A2	(4,3+5)	Fan-in is faulty	2.4
563–618	A#	All	Connected to new RD3 electronics (GaAs)	1,2.3
563–576	P#	All	Terminated at calorimeter end	2.3
577–618	A#	All	Three depth samplings instead of two	2.3
577–590	P#	Some	Terminated at calorimeter end	2.3
577–590	P#	Rest	Terminated at fan-in input	2.3
591–618	P#	All	Terminated at ADC end	2.3

Table 7: Some relevant hardware changes from logbook

eliminates the RD3 calorimeter+shapers from further consideration. This still leaves the RD27 NIM linear fan-ins, the Lemos leading into them and the Lemo+BNC+Lemo cable chain after them as possible sources.

The pedestal mean values were flickering erratically in the first half of the run period, i.e. approximately until a faulty fan-in and a damaged cable were replaced. The pedestal rms's were generally stable, until each of the following changes caused the increased low frequency noise in the above-mentioned channels, so that the rms's jumped in proportion to the noise, to a higher and wider level:

- the movement of all the trigger equipment to other sectors of the calorimeter
- the re-cabling from 2 to 3 depth samplings of the calorimeter
- the movement of the terminators for passive channels from calo/fan-in to ADC end

Double events, i.e. events close in time, can explain some of the very high pedestal rms's, but not all of them.

The hit flags and energy sums in run 607 are significantly affected by the cross-talk, which makes an e^- look more like a hadron jet in the calorimeter.

Two among the first 1000 events of run 607 each contain a single incorrect energy sum that apparently cannot be explained by the well-known timing problem of the April/May 1993 ASICs.

Acknowledgements

Fig. 1 and Fig. 2 have been designed by unnamed RD27 members. They are reproduced in the present note with the permission of N. Ellis.

Thanks go to John R. Hansen, and especially to Richard Carney and Alan Watson, for their constructive criticism of the draft of this note.

Finally, I would like to thank Nick Ellis for his well-balanced supervision of my work at CERN.

A Files on CERNVM used for the analysis

All files used for the analysis can be found on the IBM 3090 VM/CMS machine CERNVM (`cernvm.cern.ch`). Currently, a few of them are on the disk ELLISN 193, while the rest are on DICKOW 191. The DICKOW files will stay there at least until the Summer of 1994. Afterwards they may be moved to an ELLISN disk or to VXCERN.

A.1 The electronic runlog; runs used in the pedestal analysis

The file RUNLOG APRIL93 on disk ELLISN 193 at CERNVM is the "electronic runlog" from Ian P. Brawn, edited to correct the errors found during the present work. The runlog entries for the 5 runs with the highest pedestal rms's in some channel (top 5 of table 2) are shown below, in addition to the the first and the last entry of the runlog.

```
RUN |P.U.|Ebyte,|XZtape,|Backup,|Nsli,|Clock|CTA,CTB; | Beam,    |Approx.
    |    |File  |File   |File   |Offst|     |ITA,ITB  | Cell     |N.evt.
----+----+------+-------+-------+-----+-----+---------+----------+-------
    |    |      |       |       |     |     |         |          |
172 |5557| 0004,|XZ0403,|       | 60, |Ext.,|150,180; |e-, 200GeV|5669
    |    |  1   |  1    |       | -45 |25ns |4095,4095|  (16,13) |
...
455 |008B| 0045,|XZ0484,|       | 20, |Ext.,|200,200; |e-, 300GeV|7000
    |    |  6   |  6    |       | -45 |25ns |4095, 360|  (3,14)  |
...
462 |0092| 0046,|XZ0485,|       | 20, |Ext.,|200,200; |e-, 300GeV|7004
    |    |  6   |  6    |       | -45 |25ns |4095, 360|  (10,14) |
...
492 |00AF| 0050,|XZ0489,|       | 20, |Ext.,|200,200; |e-, 300GeV|5925
    |    |  7   |  7    |       | -45 |25ns |4095, 360|  (9,13)  |
...
519 |00C9| 0055,|XZ0174,|       | 20, |Ext.,|200,200; |e-, 300GeV|7004
    |    |  5   |  5    |       | -45 |25ns |4095, 360|  (13,10) |
...
607 |011B| 0067,|XZ0186,|       | 20, |Ext.,|200,200; |e-, 300GeV|7004
    |    |  5   |  5    |       | -45 |25ns |4095, 360|  (43,11) |
...
618 |0123| 0069,|XZ0188,|       | 20, |Ext.,|200,200; |TEST PULSE|1877
    |    |  2   |  2    |       | -45 |25ns |4095, 360|          |
```

The total number of runs is apparently: $618 - 172 + 1 = 447$. The electronic runlog only lists 416 of these. The skipped runs were used for machine development (runs 303–315), tests of the data aquisition system (e.g. runs 169–171) and similar purposes.

402 of the 416 runs currently have a corresponding RD27 pedestal file PED*nnn* DATA on ELLISN 193. Four of the remaining 14 runs can't ever have one, since only RD3 took data in these cases. Others have comments like "RUN ABORTED" or "PEDESTAL". Just a single of the 14 runs is fully normal. The 14 runlog entries can be found separately in the file RLOG-NNN APRIL93 on DICKOW 191.

398 of the 402 pedestal files were fed into PED2NTUP FORTRAN to create PED NTUPLE as described in Appendix A.2. The four unused pedestal files contain zeros for all trigger channels.

Three of those runs are probably unusable ("JUNK" or "NO BEAM"). The last one, run 178, failed several attempts of filling with PEDBAT EXEC (Appendix A.2), probably due to a missing tape.

A.2 From tape via pedestal files to a pedestal Ntuple

PEDBAT EXEC
is a RD27 interactive REXX script to produce a pedestal file from tape for a given test beam run number. It asks for runlog parameters (as found in RUNLOG APRIL93, see App. A.1), fills them into the template PED EXEC getting the result PEDTEMP EXEC, which it submits as a batch job. This job compiles and runs the Fortran program PED04 FORTRAN on ELLISN 193. It uses slice 3+4+5 of the first (max.) 1000 events on the tape to complete a pedestal data file PED*nnn* DATA, where *nnn* is the run number. PED172 DATA on DICKOW 191 is thus the pedestal file produced for run 172. All such pedestal files produced or corrected during the present work have been copied to the more official RD27 disk ELLISN 193.

PEDDATA ZOO
is a ZOO archive containing a single file that is a merge of all the 398 pedestal files available for the analysis. They were merged manually, one screen-full at a time, in ascending order of run number. (An attempt to do it automatically with a single CMS command resulted in descending order, which is less than ideal).

PED2NTUP FORTRAN
reads a merged file of pedestal files and stores most of the information in a Row Wise Ntuple in a HBOOK file called PED NTUPLE. The Ntuple has the following 76 fields (columns):

(run, nsl, offset, clockp, pedval11, pedval12, ..., pedval16, pedval21, ..., pedval66, pedrms11, pedrms12, ..., pedrms66)

as seen on this listing from PAW (NTUPLE/PRINT):

```
********************************************************
* NTUPLE ID=  300  ENTRIES=    398   Pedestals
********************************************************
*  Var numb  *   Name    *    Lower     *    Upper     *
********************************************************
*      1     * run       * 0.172000E+03 * 0.618000E+03 *
*      2     * nsl       * 0.200000E+02 * 0.256000E+03 *
*      3     * offset    * -.450000E+02 * 0.250000E+02 *
*      4     * clockp    * 0.120000E+02 * 0.250000E+02 *
*      5     * pedval11  * 0.217700E+02 * 0.297700E+02 *
*      6     * pedval12  * 0.182000E+02 * 0.245300E+02 *
*      7     * pedval13  * 0.196900E+02 * 0.281600E+02 *
*      8     * pedval14  * 0.210000E+02 * 0.244200E+02 *
*      9     * pedval15  * 0.230200E+02 * 0.292000E+02 *
*     10     * pedval16  * 0.245900E+02 * 0.299700E+02 *
*     11     * pedval21  * 0.227700E+02 * 0.273500E+02 *
*     12     * pedval22  * 0.200800E+02 * 0.291500E+02 *
*     13     * pedval23  * 0.210200E+02 * 0.264100E+02 *
```

14	pedval24	0.208500E+02	0.246700E+02	
15	pedval25	0.221700E+02	0.278100E+02	
16	pedval26	0.201200E+02	0.244200E+02	
17	pedval31	0.208400E+02	0.284700E+02	
18	pedval32	0.229100E+02	0.284900E+02	
19	pedval33	0.250600E+02	0.292900E+02	
20	pedval34	0.230200E+02	0.285900E+02	
21	pedval35	0.249100E+02	0.316900E+02	
22	pedval36	0.258500E+02	0.309800E+02	
23	pedval41	0.241200E+02	0.290800E+02	
24	pedval42	0.252700E+02	0.311100E+02	
25	pedval43	0.240100E+02	0.307000E+02	
26	pedval44	0.230200E+02	0.298100E+02	
27	pedval45	0.254800E+02	0.328300E+02	
28	pedval46	0.215200E+02	0.284000E+02	
29	pedval51	0.280200E+02	0.327400E+02	
30	pedval52	0.269600E+02	0.298300E+02	
31	pedval53	0.280000E+02	0.307400E+02	
32	pedval54	0.280500E+02	0.311700E+02	
33	pedval55	0.257000E+02	0.302400E+02	
34	pedval56	0.259800E+02	0.284100E+02	
35	pedval61	0.296400E+02	0.341900E+02	
36	pedval62	0.251800E+02	0.280800E+02	
37	pedval63	0.286500E+02	0.331000E+02	
38	pedval64	0.273300E+02	0.293700E+02	
39	pedval65	0.271300E+02	0.305000E+02	
40	pedval66	0.253500E+02	0.291300E+02	
41	pedrms11	0.470000E+00	0.146000E+01	
42	pedrms12	0.000000E+00	0.144000E+01	
43	pedrms13	0.570000E+00	0.142000E+01	
44	pedrms14	0.000000E+00	0.703000E+01	
45	pedrms15	0.000000E+00	0.131000E+01	
46	pedrms16	0.440000E+00	0.125000E+01	
47	pedrms21	0.330000E+00	0.145000E+01	
48	pedrms22	0.400000E+00	0.215000E+01	
49	pedrms23	0.520000E+00	0.125000E+01	
50	pedrms24	0.470000E+00	0.549000E+01	
51	pedrms25	0.470000E+00	0.107000E+01	
52	pedrms26	0.000000E+00	0.900000E+00	
53	pedrms31	0.000000E+00	0.920000E+00	
54	pedrms32	0.000000E+00	0.113000E+01	
55	pedrms33	0.440000E+00	0.518000E+01	
56	pedrms34	0.440000E+00	0.324000E+01	
57	pedrms35	0.550000E+00	0.151000E+01	
58	pedrms36	0.460000E+00	0.101000E+01	
59	pedrms41	0.470000E+00	0.109000E+01	
60	pedrms42	0.000000E+00	0.168000E+01	

```
*     61     * pedrms43  * 0.570000E+00 * 0.341000E+01 *
*     62     * pedrms44  * 0.470000E+00 * 0.456000E+01 *
*     63     * pedrms45  * 0.470000E+00 * 0.173000E+01 *
*     64     * pedrms46  * 0.470000E+00 * 0.204000E+01 *
*     65     * pedrms51  * 0.000000E+00 * 0.158000E+01 *
*     66     * pedrms52  * 0.470000E+00 * 0.161000E+01 *
*     67     * pedrms53  * 0.000000E+00 * 0.531000E+01 *
*     68     * pedrms54  * 0.370000E+00 * 0.415000E+01 *
*     69     * pedrms55  * 0.450000E+00 * 0.228000E+01 *
*     70     * pedrms56  * 0.000000E+00 * 0.159000E+01 *
*     71     * pedrms61  * 0.000000E+00 * 0.157000E+01 *
*     72     * pedrms62  * 0.000000E+00 * 0.157000E+01 *
*     73     * pedrms63  * 0.470000E+00 * 0.230000E+01 *
*     74     * pedrms64  * 0.430000E+00 * 0.255000E+01 *
*     75     * pedrms65  * 0.290000E+00 * 0.407000E+01 *
*     76     * pedrms66  * 0.000000E+00 * 0.158000E+01 *
****************************************************
```

The result for these data is 398 rows, with run increasing from 172 to 618. The fact that 13 channels have a minimum rms equal to zero is found to be entirely caused by run 616. According to the electronic runlog, this run uses test pulses, and has an approximate number of events equal to zero. Probably the actual number of events was one.

The program can easily be extended to include more fields, such as cluster/isolation thresholds, number of events used for the pedestal calculation etc. Numerical constants are defined in the separate include file PED2NTUP INC.

PED2CWN FORTRAN
almost works as PED2NTUP but creates a Column Wise Ntuple (CWN) instead, in a file named PED HBOOK. This was tried first, but abandoned eventually because of the limitations in PAW 2.02's parsing of NTUPLE/SCAN and NTUPLE/PLOT command lines involving CWNs. (Versions 2.03 and 2.04 of PAW supposedly improve on this point). A useful side effect of the program was that the HBOOK CWN range-checks caught several errors in the pedestal files, e.g. a cluster threshold with the value 700 instead of 70.

PED2NTUP EXEC and PED2CWN EXEC
are REXX scripts to execute the corresponding pre-compiled Fortran programs, after making the appropriate input/output file definitions. They do so much internally that all one has to do is to simply write e.g. PED2NTUP in front of the name of a merged pedestal file in a CMS FILELIST. Shortly after, the corresponding result files PED NTUPLE and PED2NTUP OUT are available. The last one contains a copy of the input, mixed with any error messages that might have been produced.

A.3 From tape to a raw Ntuple

RAW2NT FORTRAN
reads raw data for a single run from tape and stores the full information (except camac data) for selected events in a Column Wise Ntuple in the file BOLRAW HBOOK, for later analysis with

PAW (version 2.02 or above). The Ntuple has the following 57 columns:
(run, ievent, islice, iadc11, iadc21, ..., iadc66, iesum1, ..., iesum9, ihit1, ..., ihit9) giving this listing from PAW (for run 607):

```
*****************************************************************
* Ntuple ID = 2      Entries = 2540      Raw data: ADC & ASIC output
*****************************************************************
* Var numb * Type * Packing *    Range     *  Block   *  Name    *
*****************************************************************
*      1   * I*4  *    15   * [0,9999]     * RAWKEY   * run
*      2   * I*4  *    21   * [1,999999]   * RAWKEY   * ievent
*      3   * I*4  *    10   * [1,256]      * RAWKEY   * islice
*      1   * I*4  *    9    * [0,255]      * RAWADC   * iadc11
*      2   * I*4  *    9    * [0,255]      * RAWADC   * iadc21
*      3   * I*4  *    9    * [0,255]      * RAWADC   * iadc31
*      4   * I*4  *    9    * [0,255]      * RAWADC   * iadc41
*      5   * I*4  *    9    * [0,255]      * RAWADC   * iadc51
*      6   * I*4  *    9    * [0,255]      * RAWADC   * iadc61
*      7   * I*4  *    9    * [0,255]      * RAWADC   * iadc12
*      8   * I*4  *    9    * [0,255]      * RAWADC   * iadc22
*      9   * I*4  *    9    * [0,255]      * RAWADC   * iadc32
*     10   * I*4  *    9    * [0,255]      * RAWADC   * iadc42
*     11   * I*4  *    9    * [0,255]      * RAWADC   * iadc52
*     12   * I*4  *    9    * [0,255]      * RAWADC   * iadc62
*     13   * I*4  *    9    * [0,255]      * RAWADC   * iadc13
*     14   * I*4  *    9    * [0,255]      * RAWADC   * iadc23
*     15   * I*4  *    9    * [0,255]      * RAWADC   * iadc33
*     16   * I*4  *    9    * [0,255]      * RAWADC   * iadc43
*     17   * I*4  *    9    * [0,255]      * RAWADC   * iadc53
*     18   * I*4  *    9    * [0,255]      * RAWADC   * iadc63
*     19   * I*4  *    9    * [0,255]      * RAWADC   * iadc14
*     20   * I*4  *    9    * [0,255]      * RAWADC   * iadc24
*     21   * I*4  *    9    * [0,255]      * RAWADC   * iadc34
*     22   * I*4  *    9    * [0,255]      * RAWADC   * iadc44
*     23   * I*4  *    9    * [0,255]      * RAWADC   * iadc54
*     24   * I*4  *    9    * [0,255]      * RAWADC   * iadc64
*     25   * I*4  *    9    * [0,255]      * RAWADC   * iadc15
*     26   * I*4  *    9    * [0,255]      * RAWADC   * iadc25
*     27   * I*4  *    9    * [0,255]      * RAWADC   * iadc35
*     28   * I*4  *    9    * [0,255]      * RAWADC   * iadc45
*     29   * I*4  *    9    * [0,255]      * RAWADC   * iadc55
*     30   * I*4  *    9    * [0,255]      * RAWADC   * iadc65
*     31   * I*4  *    9    * [0,255]      * RAWADC   * iadc16
*     32   * I*4  *    9    * [0,255]      * RAWADC   * iadc26
*     33   * I*4  *    9    * [0,255]      * RAWADC   * iadc36
*     34   * I*4  *    9    * [0,255]      * RAWADC   * iadc46
*     35   * I*4  *    9    * [0,255]      * RAWADC   * iadc56
```

```
*      36   * I*4  *    9    * [0,255]       * RAWADC   * iadc66
*       1   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum1
*       2   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum2
*       3   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum3
*       4   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum4
*       5   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum5
*       6   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum6
*       7   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum7
*       8   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum8
*       9   * I*4  *    13   * [0,4095]      * RAWESUM  * iesum9
*       1   * U*4  *    2    * [0,3]         * RAWHIT   * ihit1
*       2   * U*4  *    2    * [0,3]         * RAWHIT   * ihit2
*       3   * U*4  *    2    * [0,3]         * RAWHIT   * ihit3
*       4   * U*4  *    2    * [0,3]         * RAWHIT   * ihit4
*       5   * U*4  *    2    * [0,3]         * RAWHIT   * ihit5
*       6   * U*4  *    2    * [0,3]         * RAWHIT   * ihit6
*       7   * U*4  *    2    * [0,3]         * RAWHIT   * ihit7
*       8   * U*4  *    2    * [0,3]         * RAWHIT   * ihit8
*       9   * U*4  *    2    * [0,3]         * RAWHIT   * ihit9
******************************************************************
*  Block    *  Entries  * Unpacked * Packed *   Packing Factor    *
******************************************************************
* RAWKEY    *  2540     * 12       * 6      *       2.000         *
* RAWADC    *  2540     * 144      * 41     *       3.512         *
* RAWESUM   *  2540     * 36       * 15     *       2.400         *
* RAWHIT    *  2540     * 36       * 3      *      12.000         *
* Total     *    ---    * 228      * 65     *       3.508         *
******************************************************************
* Blocks = 4           Variables = 57           Columns = 57      *
******************************************************************
```

The "run" field allows data for several runs to be merged, as described later. Note that what PAW calls an "event" is here only a single time slice of a physical RD27 event. The RD27 events are "unfolded" in this way, because PAW 2.02/06 can't handle array syntax in NTUPLE/PLOT statements. RAW2NT selects events in this way: it takes the first 100 events of the run (skipping the very first 16), and fills up with extra events until 20 *bad* and 20 *wild* events have been saved, or end of file or event no. 1000 has been reached. *Bad* means having a bad energy sum (not simply delayed). *Wild* means having a pedestal value in slice 3, 4 or 5 which deviates more than 8.0 ADC counts from the originally calculated mean. All the limits are configurable Fortran parameters in the separate include file RAW2NT INC. Numerous comments have been made in the source code.

RAW2NT EXEC
is analogous to the pedestal REXX EXECs described in the previous section.

RAW2NT OUT607
is the text output from RAW2NT FORTRAN's processing of the first 1000 events of run 607. All

bad ASIC energy sums ("Esums") and *wild* pedestal values encountered are noted there. The 5 lines of text for each *bad* event are conveniently extracted on VMS (e.g. on VXCERN) with the following command: `$ search/window raw2nt.out607 Timing`.

BOLRAW HBOOK
contains the raw Column Wise Ntuple mentioned above, for run 607. (It also contains some 1D histograms that are side effects of RAW2NT). On CERNVM, it uses 78 disk blocks of 4Kb each, i.e. 312Kb for the 127 events (and negligible histograms). A crucial reason for using a CWN instead of a RWN here was that the corresponding RWN would take up 3–4 times more disk space.

It has been successfully copied with FTP to a Copenhagen VAX/VMS system, the record size reset to 4096 bytes and a bit of reanalysis done with PAW 2.02/06. Almost another factor of 4 in disk space was saved, when a backup copy of the Ntuple was compressed with a Public Domain Lempel-Ziv compression program (LZCOMP), resulting in just 83.5Kb for the 127 events.

If somebody wants to analyze raw Ntuple data for several runs together, then at least two options can be suggested. Either rename BOLRAW HBOOK to e.g. RAW*nnn* HBOOK for run *nnn*, and use PAW's NTUPLE/CHAIN command to obtain a single virtual Ntuple. Or change a single letter in the line of RAW2NT FORTRAN that currently *creates* the Ntuple file, so that it instead opens an existing BOLRAW HBOOK for *update*.

A.4 From Ntuples to PAW plots using KUMACs and UWfuncs

Almost all of the plots were made using KUip MACros containing the appropriate PAW commands. A total of about 50 such KUMACs were produced. Most of them are just small modifications of the previous one, usually with only a change of numerical constants. For this reason, the comment in the beginning the KUMAC is often wrong, because it wasn't changed to reflect the new parameters. Five other warnings to potential users should be given here:

- Concerning plots with more than one histogram (of channels or ASICs) on one page: All of them are displayed rotated 90 degrees clockwise compared to the RD27 standard used in the present note, except the few with postfix "R" (e.g. SADC_A3R KUMAC).

- Since the author had no prior experience with Ntuples or histograms in PAW, the first KUMACs made are not optimal. Specifically, all the 2-variable pedestal plots use 2D histograms, even though most of them could in fact have been made with 1D. When the 1D method was discovered, the old way with the pedestal plots was nevertheless retained, so as not to make visual comparisons between the plots too difficult.

- In some of the raw Ntuple KUMACs there is a "run=[run]" cut, in others not. Until now, the Ntuple has in practice never contained more than one run (run 607), so the cut hasn't made any difference, but future users should be aware of it.

- Two types of changes in the raw Ntuple KUMACs were so small that new file names weren't created for them: adding or removing an "ievent=[event]" cut, i.e. plotting either one event or (the sum of) all events; and changing the default event number for the cut.

- All UWfuncs are written without array syntax, because PAW/COMIS couldn't sensibly accept any attempts of declaring the Ntuple variables as arrays in the UWfunc. Only after the end of the CERN stay, a NEWS item appeared which explained that PAW needs the original Ntuple variable names in order to only load those columns that are actually needed from the Column-Wise Ntuple on disk. Another news suggested to use an EQUIVALENCE statement for the declarations. The combination of these approaches turn out to work fine.

Most of the macros and UWfuncs are described in the following. Obsolete or test macros are omitted. The macros producing the plots in the figures of the present note are:

Figure 4:	RPR_53A KUMAC
Figure 5:	SADC_53H KUMAC
Figure 6:	SADC_A3R KUMAC
Figure 7:	RPV_12 KUMAC
Figure 8:	A53A55 KUMAC
Figure 9:	SHIT3_H KUMAC
Figure 10:	SESUME_R KUMAC

The files on DICKOW 191 used during PAW analysis, including the Ntuple files, are known to work equally well with PAW 2.x under the operating systems VMS and Unix.

PAWLOGON KUMAC

connects the pedestal and raw Ntuples to the logical units expected by all the other KUMACs. Also does a few other things, such as enabling automatic macro execution.

TOFILE KUMAC

is a handy home-made macro that produces the plot file corresponding to a given parameterless macro. Usage:

PAW> tofile *macro* [*filename* [*metafile*]]

Default *filename* is "tofile.ps", default *metafile* is -111 (PostScript). See comments inside it for more information. It was made for PAW 1.14; it could be made more intelligent with PAW 2.x.

RPV_*xx*H KUMAC

creates a (Run, Pedestal Value) plot for a quadrant of 3×3 channels. The quadrant is denoted by Upper/Lower Left/Right, corresponding to *xx* equal to UL, UR, LL or LR. The suffix H is historical, standing for Histogram. They use 2D histograms.

RPR_*xx*H KUMAC

creates 3×3 (Run, Pedestal Rms) plots analogously to the pedestal value plots. The 4 paper copies at CERN are glued together to a big plot of all 6×6 channels. Characteristic features and exceptionally large rms's are marked out with ball pen.

RPR_FEAT KUMAC

emphasizes the most prominent FEATure of the (Run, Pedestal Rms) plots, namely the large, widespread rise after run 562. It does so by plotting the 2×3 A3-channels for run 541–end.

RPR_FEAD KUMAC

uses exactly the same selection of pedestal rms's as the previous macro, but makes a single 2D plot of the Differences, to show how closely they follow each other.

RPR_FETP KUMAC and RPR_FEDP KUMAC
correspond to the two previous macros, but plot the 6 P3-channels instead. RPR_FETP is the only (Run, Pedestal) macro using 1D histograms internally.

RPR_A2D KUMAC
as RPR_FEAD KUMAC, but for the A2-channels.

RPR_A2FS KUMAC
as RPR_FEAT KUMAC, but superimposes the 6 A2-channels on the same plot.

RUNCUT KUMAC
histograms the frequency of the pedestal values in channel (2,2) in two ways: for all runs, and for the cut 399.5<run<540.5. The cut covers most of the runs in the stable period identified in Sec. 2.4.

RUNCUT2 KUMAC
works as the previous macro, but for pedestal rms's instead of values. (Also excludes a single run with exceptional rms).

RMISC KUMAC
plots MISCellaneous run parameters as a function of the Run number, namely: number of slices, offset and clock period.

UWMAXRMS FORTRAN, UWMRMSIJ FORTRAN, UWNUMRMS FORTRAN
are Ntuple selection functions that return the following for the current run:

- UWMAXRMS: The largest of the 36 rms's
- UWRMSIJ: Channel id ($10 \times i + j$) for largest rms
- UWNUMRMS(*rmscut*): Number of rms's larger than *rmscut*.

They have been used with NTUPLE/SCAN to find the corresponding numbers given in table 2. These or similar functions might be useful for pedestal analysis with NTUPLE/PLOT in future KUMACs.

ADC_xx KUMAC
histograms the frequency of the ADC values in the 3×3 channels indicated by xx (Upper/Lower Left/Right), summed over all slices. These macros were quickly made — to get an overview of the channel data in the raw Ntuple — and so the titles were not correctly set.

SADC_A3 KUMAC
makes a (Slice, ADC value) plot for each of the 2×3 A3-channels. Uses 2D-histograms.

SADC_AnH KUMAC and SADC_PnH KUMAC
make (Slice, ADC value) plots for the Active resp. Passive channels on ADC card #n using 1D-Histograms. Note that the possibility of out-commenting the "ievent=[event]" cut is especially useful here.

SADC_AnM KUMAC and SADC_PnM KUMAC
work as above, but plot 3 different events, i.e. plot Multiple events separately.

SADC_53 KUMAC and SADC_53H KUMAC
are named after the same system as above: the first one makes a 2D (Slice, ADC value) plot for all events in channel (5,3), while the second one makes a 1D-histogram for a single event or sum of all events. Channel (5,3) is the beam cell of run 607.

SHIT_H KUMAC and SHIT3_H KUMAC
make (Slice, Hit bit) plots for all 3×3 ADCs, respectively only ADC #3. They use 1D-histograms.

SESUM_H KUMAC
plots (Slice, Esum) 1D-histograms for all 3×3 ADCs.

SESUME_H KUMAC using UWESUME FORTRAN
works as above, but also plots the Expected energy sums. The UWfunc takes an ASIC number as argument, and returns the expected energy sum for that ASIC for the current slice.

SAD53PED KUMAC using UWSLI53 FORTRAN
makes a 2D (Slice, ADC PEDestal difference) plot for channel (5,3), for slice 1-9 of the first 100 events. The y-value plotted for a given slice and event is the difference between the ADC value in that slice and the ADC value in a selected pedestal slice, currently number 1. This gives an impression of how little the pedestal values fluctuate within a single event. However, this method overestimates the fluctuations: If all events fluctuated randomly between just two ADC values each, they would still spread over three ADC differences in the plot, namely −1, 0 and +1. It would thus be better to subtract for example the mean of slice 1–9, but that would make the programming much more difficult with the current Ntuple structure.

ADC53PED KUMAC using UWSLI53 FORTRAN
effectively plots the projection of the previous plot onto the y-axis, except that it only uses slice 1–8 currently. The warning given for the previous macro applies, so that the rms reported by PAW is considerably larger than the rms expectation value for the ADC values in slice 1–8 of a randomly selected event.

UWCLUSP FORTRAN
can return the (maximum) Cluster Sum for ASIC #3 for the previous slice, as defined by the wrongly internally connected ASICs of the April/May '93 run. This is useful when checking hit flags, whether with NTUPLE/SCAN or NTUPLE/PLOT.

UWPESUME FORTRAN using UWESUME FORTRAN
prints expected and actual energy sums and the difference between them, when used with NTUPLE/LOOP as shown in App. A.5.

A.5 Expected and actual energy sums for the 2 unexplained events

The expected and actual energy sums for the two events of run 607 with only one *bad* slice each (Sec. 3.2) are shown in the following PAW session output:

```
PAW > nt/loop 2 uwpesume.fortran 20 2461
Event slice Esum(expect) Hex(exp) Hex(actual) Esum(act) Diff
  715     1          384      180         180       384    0
  715     2          378      17A         17A       378    0
  715     3          383      17F         17F       383    0
  715     4          384      180         130       304  -80  <--- BAD!
  715     5          380      17C         17C       380    0
  715     6          380      17C         17C       380    0
  715     7          384      180         180       384    0
  715     8          380      17C         17C       380    0
  715     9          383      17F         17F       383    0
  715    10          532      214         214       532    0
  715    11          647      287         287       647    0
  715    12          531      213         213       531    0
  715    13          409      199         199       409    0
  715    14          358      166         166       358    0
  715    15          361      169         169       361    0
  715    16          360      168         168       360    0
  715    17          359      167         167       359    0
  715    18          357      165         165       357    0
  715    19          358      166         166       358    0
  715    20          357      165         165       357    0
PAW > nt/loop 2 uwpesume.fortran 20 2501
Event slice Esum(expect) Hex(exp) Hex(actual) Esum(act) Diff
  795     1          429      1AD         1AD       429    0
  795     2          425      1A9         1A9       425    0
  795     3          426      1AA         1AA       426    0
  795     4          424      1A8         1A8       424    0
  795     5          425      1A9         1A9       425    0
  795     6          427      1AB         1AB       427    0
  795     7          425      1A9         1A9       425    0
  795     8          427      1AB         1AB       427    0
  795     9          428      1AC         1AC       428    0
  795    10          452      1C4         1C4       452    0
  795    11          543      21F         21F       543    0
  795    12          511      1FF         1FF       511    0
  795    13          448      1C0         1E0       480   32 <--- BAD!
  795    14          417      1A1         1A1       417    0
  795    15          410      19A         19A       410    0
  795    16          405      195         195       405    0
  795    17          400      190         190       400    0
  795    18          407      197         197       407    0
  795    19          400      190         190       400    0
  795    20          403      193         193       403    0
```

(The "BAD" text was added afterwards). The energy sums shown concern ASIC #2 for both events. The COMIS function UWPESUME FORTRAN was written specifically with these two events in mind, so the ASIC number was hard-coded in it. It gets the expected energy sums from UWESUME FORTRAN (App. A.4). The *PAW event* numbers “2461” and “2501” were found with the PAW command NTUPLE/SCAN.

It is seen from the output that the “late upper bits”-explanation [10] doesn’t work in these two cases.

References

[1] RD27: *First-level Trigger Systems for LHC Experiments*, RD27 Status Report, 11 August 1993, CERN/DRDC 93-32.

[2] I.P. Brawn et. al.: *Beam Tests of a Prototype Level-1 Calorimeter Trigger for LHC Experiments*, 1994, RD27 note 22. (To be submitted to Nuclear Instruments and Methods).

[3] ATLAS: *ATLAS Letter of Intent for a General-Purpose pp Experiment at the Large Hadron Collider at CERN*, 1 October 1992, CERN/LHCC 92-4.

[4] René Brun: *PAW, Physics Analysis Workstation, The Complete Reference, Version 2.03*, October 1993, CERN Program Library Long Writeup Q121. (All work was done with version 2.02, the difference mainly being the bugs).

[5] René Brun: *HBOOK Reference Manual, Version 4.20*, October 1993, CERN Program Library Long Writeup Y250. (All work was done with version 4.17).

[6] Richard Carney: *Level-1 Bit-Parallel E.M. Trigger Processor Beam Test’s*, copy of transparencies from RD27 Meeting CERN 07 June 1993. They contain useful drawings of the *trigger cell* $\leftrightarrow$ *calo cell* correspondence, with beam spots plotted in.

[7] Ulrik Dickow: *Features in plots of pedestal mean & rms versus run number in the 6⋆6 RD27 1st level e.m. trigger channels for the April/May 1993 test beam runs (run 172–618)*, 26 July 1993. A 1 1/2 page memo communicated by E-mail to a selection of RD27-members. Saved as PEDFEATU MEMO on DICKOW 191 at CERNVM.

[8] Richard Carney: E-mail reply to the above memo to 15 other persons, 03 August 1993. Saved as PEDCARNE MAILFROM on DICKOW 191 at CERNVM.

[9] Richard Carney: *RD27 Logbook Findings, April/May 1993 Test Beam Run*, currently unnumbered RD27 note, October 1993. Contains day-by-day information from the paper logbook, extracted in response to (the draft of) the present note, and intended to be read in conjunction with it. It also contains detailed figures of the cabling scheme.

[10] Alan Watson: Private communication (E-mail reply on 11 Oct 1993 to the draft of the present note).