

*ESPACE* (εspas)

M A N U A L

V E R S I O N 2.9.0β

Event Scanning Program for Hall A Collaboration Experiments  
The Hall A Collaboration  
August 12, 2002



# Contents

<b>1</b>	<b>General Information</b>	<b>1</b>
1.1	Program Concept . . . . .	1
1.2	How to Start? . . . . .	3
1.2.1	The ESPACE binary . . . . .	3
1.2.1.1	System installation . . . . .	3
1.2.1.2	Private installation . . . . .	4
1.2.2	Getting ready . . . . .	8
1.2.2.1	Runtime Errors . . . . .	11
<b>2</b>	<b>Variables</b>	<b>13</b>
2.1	Units of the Variables . . . . .	13
2.2	Overview of Pre-Defined Variables . . . . .	13
2.3	Coding New Variables . . . . .	27
2.4	Defining New Variables . . . . .	29
<b>3</b>	<b>Defining Histograms, Logicals and Filters</b>	<b>31</b>
3.1	Histograms . . . . .	31
3.2	Logicals . . . . .	32
3.3	Filters . . . . .	34
3.4	Global Conditions . . . . .	35
3.5	When is What Stored? . . . . .	36
<b>4</b>	<b>Coordinate Systems</b>	<b>37</b>
4.1	Definition of Coordinate Systems . . . . .	37
4.1.1	Hall A Laboratory Coordinate System . . . . .	37
4.1.2	Spectrometer Reconstructed Coordinate System . . . . .	37
4.1.3	Spectrometer Detector Coordinate System . . . . .	39
4.1.4	Spectrometer Focal-Plane Coordinate System . . . . .	39
4.2	Calculation of Coordinates . . . . .	39
4.2.1	Detector Vertex . . . . .	40
4.2.2	Focal-Plane Vertex . . . . .	43
4.2.3	Hall A Laboratory Vertex . . . . .	44
4.3	Calibration of Coordinates . . . . .	45

<b>5</b>	<b>Tracking</b>	<b>47</b>
5.1	Program Flow for Tracking . . . . .	47
5.2	Minimum Spanning Tree Algorithm . . . . .	47
5.3	Definition of Wire Chamber Hit Type HPCH . . . . .	47
5.4	Tracking Efficiency . . . . .	48
<b>6</b>	<b>Calibration</b>	<b>49</b>
6.1	ADC Gains and Pedestals . . . . .	50
6.2	Čerenkov ADC . . . . .	54
6.3	Shower and Pre-Shower Calibration . . . . .	54
6.4	Scintillator Timing . . . . .	55
6.5	VDC Time Offsets . . . . .	55
6.6	VDC Time to Distance . . . . .	55
6.7	Vertex Reconstruction . . . . .	55
6.7.1	Non-Dispersive Position $\mathbf{y}_{tg}$ . . . . .	58
6.7.2	Angles . . . . .	58
6.7.3	Momentum Aberrations . . . . .	58
6.8	Coincidence Time . . . . .	61
6.9	Missing Energy . . . . .	61
6.10	Adding a new Optimization . . . . .	61
6.11	Beam Position Determination . . . . .	61
6.11.1	The Beam Position Monitors . . . . .	61
6.11.2	Beam Position Monitor Calibration . . . . .	63
6.11.3	Phase Shift . . . . .	65
6.11.4	Struck 7510 “Burst Mode” ADC . . . . .	65
<b>A</b>	<b>Header File Description</b>	<b>69</b>
<b>B</b>	<b>Database File Description</b>	<b>75</b>
B.1	Raster and Beam Position Information . . . . .	75
B.2	The other Stuff . . . . .	76
<b>C</b>	<b>Transformation between Different Coordinate Systems</b>	<b>79</b>
<b>D</b>	<b>ESPACE Reference Manual</b>	<b>81</b>
D.1	Espace/Debug . . . . .	81
D.2	Espace/Set . . . . .	82
D.2.1	Espace/Set/file . . . . .	83
D.2.2	Espace/Set/timing . . . . .	83
D.2.3	Espace/Set/type . . . . .	84
D.2.4	Espace/Set/fit . . . . .	85
D.2.4.1	Espace/Set/fit/Adc . . . . .	85
D.2.4.2	Espace/Set/fit/time . . . . .	85

D.2.4.3	Espace/Set/fit/momentum . . . . .	86
D.2.5	Espace/Set/spectrum . . . . .	86
D.2.5.1	Espace/Set/spectrum/window . . . . .	86
D.2.5.2	Espace/Set/spectrum/bins . . . . .	87
D.2.6	Espace/Set/vdc . . . . .	87
D.3	Espace/Define . . . . .	88
D.3.1	Espace/Define/cut . . . . .	89
D.4	Espace/Spectra . . . . .	90
D.5	Espace/File . . . . .	91
D.6	Espace/Calibrate . . . . .	93
<b>E</b>	<b>Flow Diagram</b>	<b>95</b>
<b>F</b>	<b>COOLHANDS Programming Guide</b>	<b>127</b>
F.1	Concept . . . . .	127
F.2	FORTTRAN Components . . . . .	127
F.3	FORTTRAN Unit Numbers and Common Blocks . . . . .	128
F.4	User-Supplied Subroutines . . . . .	129
F.4.1	input . . . . .	130
F.4.2	variable . . . . .	133
F.4.3	user_init . . . . .	137
F.4.4	user_exit . . . . .	138
F.4.5	filter_open, filter_event and filter_close . . . . .	138
F.4.6	option_do . . . . .	139



# Chapter 1

## General Information

The program `ESPACE` can analyze experiments performed with one or two high-resolution spectrometers. In this manual we will discuss the implementation specific to the setup in hall A at TJNAF.

`ESPACE` can filter, histogram and/or calibrate variables of the experimental setup while applying conditions on the incoming data. The variables range from raw detector signals, like the value of a scintillator TDC channel to much more elaborate ones like the coordinates of the reaction point in the target.

The program interface is supplied by the `KUIP CERNLIB` and, therefore, the program has a PAW-like look. Details about this interface can be found in the CERN documentation of [1] and [2].

### 1.1 Program Concept

Let's try to explain the history and the concepts behind the analysis program `ESPACE`<sup>1</sup>. The program is build up in three layers where each layer on top of the lower one adds more information specific to the experiments.

The first layer performs operations on variables defined in higher layers, for each event input. These operations can be histogramming, filtering, evaluating conditions and/or accumulating data for an optimization cycle. The routines are collected in the `COOLHANDS`<sup>2</sup>-library. The first version of this package was written by Joe Mandeville [3]. In `COOLHANDS` it is possible to define 'on the fly' new variables as a function of existing ones with a FORTRAN expression evaluator contained in the `FCALC`-library (also written by Joe).

Figure 1.1 shows how the data flow from a `COOLHANDS` perspective. After initialization in `COOLHANDS` and possibly in a routine supplied by the upper `ESPACE`-layer, commands supplied through the `KUIP` interpreter are executed. The most important ones are:

- definition of an ntuple or histogram through the `save` command which triggers a call to the `MBOOK4D` library, an interface to the CERN `HBOOK`-library.

---

<sup>1</sup>`ESPACE`: Experiment Scanning Program for hall A Collaboration Experiments

<sup>2</sup>`COOLHANDS`: Command Oriented Off-Line Histogramming AND Scanning program

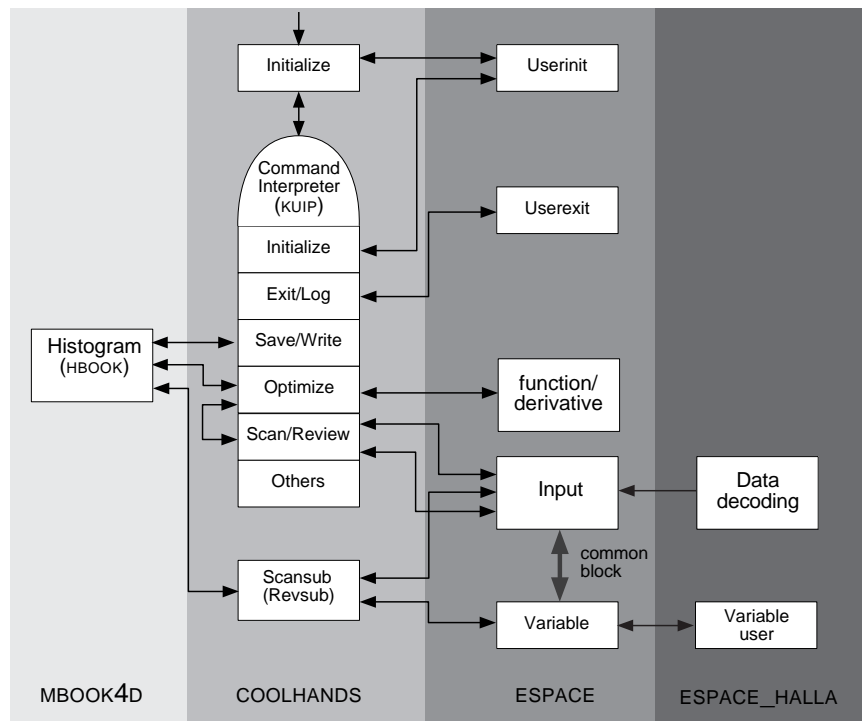


Figure 1.1: Diagram showing the flow of data around the COOLHANDS library.

- the `scan` command which calls in the ESPACE library the `input` routine. It opens all the necessary files and calls on its turn the raw data decoding routine in the top layer, which is experiment specific (Here at TJNAF it is a set of CODA routines [4]). After receiving data, `input` will call the routine `scansub`, the central routine of COOLHANDS. This routine first evaluates all the logical definitions, and then stores the data through MBOOK for histogramming and optimization, applying all the cuts defined. To analyze the logical definitions and to store the appropriate values, `scansub` has to call the ESPACE-routine `variable` where the different variables have been defined.

The story looks a bit different from an ESPACE-perspective (Fig. 1.2). After the `scan` command has been issued, COOLHANDS calls the ESPACE routine `input` which takes over till the data file has been analyzed. The data file and some other necessary files are opened and in `process_data` a loop is started over all data events. This is the center piece of the two-spectrometer analysis in ESPACE.

`process_data` will call for each spectrometer the routine `process_spectrometer` which digests all the detector information and for instance, evaluates all the tracks through the wire chamber package. The track that made the trigger signal is determined and traced back through the magnet to the spectrometer entrance. Next, the reaction point in the target is determined with the vertex information of both spectrometers and the information of the beam position monitors for the incoming beam (`reaction_z`). Having the reaction

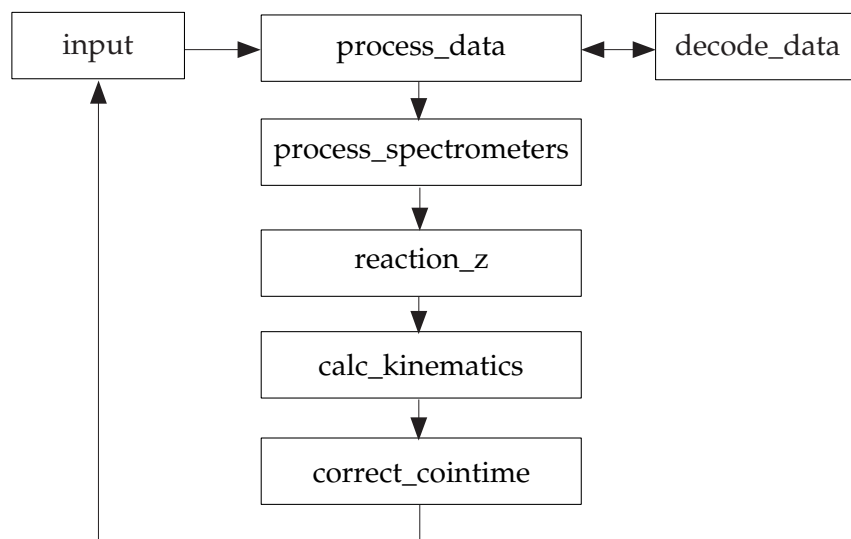


Figure 1.2: Diagram showing the flow of data in the ESPACE library.

point and the target geometry, corrections can be made for the energy losses of incoming and outgoing particles in the target. This is coordinated by the routine `calc_kinematics`, together with calculation of numerous kinematical variables (`kinematics` library). After correcting the coincidence time between the two spectrometers for flight-path differences wrt. the central ray (`correct_cointime`), control is given back to the routine `input` which will call the COOLHANDS-routine `scansub` to evaluate all the logicals, histograms, ...

## 1.2 How to Start?

The ESPACE Makefile supports the operating systems HP-UX, SunOS, OSF1 and Linux. Nowadays, most development is done on Linux. Before submitting a new version, the code is also checked on HP-UX and SunOS (both onsite) and OSF1 machines.

### 1.2.1 The ESPACE binary

#### 1.2.1.1 System installation

Onsite, ESPACE binaries are stored in the `/work/halla/software/bin/$OSNAME` directories. Here `$OSNAME` stands for either HP-UX, SunOS or Linux. To execute `espace`, you have to modify your shell initialization (`rc`) file<sup>3</sup>. For `bash` and `sh`, you should add/modify the

---

<sup>3</sup>If you choose to edit your login file (either `~/.bash_profile`, `~/.bash_login` or `~/.profile` for `bash` and `sh`, `~/.login` for `tcsh` and `csh`) you get the definitions for an interactive shell but not for a non-interactive (batch) shell, where only the `rc` file is executed. For `bash` and `sh` there is the additional problem, that for an interactive shell only the login file is executed. However, in a standard installation the `~/.bashrc` should be called by the login file. Check your login file for that.

following lines in `~/bashrc`:

```
export HALLA_DIR=/work/halla/software
export OSNAME='uname -s'
export PATH=$HALLA_DIR/bin/$OSNAME/bin:PATH
```

For `tcsh` and `csch`, you should add/modify the following lines in `~/tcshrc` or `~/cshrc` (you should have exactly one of those):

```
setenv HALLA_DIR /work/halla/software
setenv OSNAME 'uname -s'
setenv PATH $HALLA_DIR/bin/$OSNAME/bin:PATH
```

The definitions of `HALLA_DIR` and `OSNAME`<sup>4</sup> are needed<sup>5</sup> to run `espace`. Appending the `espace` binary directory to the `PATH` is not really necessary. Calling `ESPACE` by typing `$HALLA_DIR/bin/$OSNAME/espace` (either exactly that way or with `HALLA_DIR` and `OSNAME` expanded) does the job too. Though it works it is not recommended to copy the binary to the directory where you (think you) need it. In a multi-OS environment like the CUE you will lose track for which OS that particular file was compiled. To conclude, the only two definitions you really need are `HALLA_DIR` and `OSNAME`, `PATH` is for convenience.

### 1.2.1.2 Private installation

If you want to have your own version of `ESPACE`, you will find online help by clicking on `ESPACE` on the Hall A web page. Currently, this will lead to

[http://www.jlab.org/Hall-A/data\\_reduc/calibrations/software.html](http://www.jlab.org/Hall-A/data_reduc/calibrations/software.html)

**Downloading the source distribution** You have to decide where you want to store your `ESPACE` sources. Generally, users create an `espace` directory in their `$HOME`, but this is not necessary. Whatever you do, define this directory as your `HALLA_DIR`. On the `ESPACE` web page you will find a link to download the latest `ESPACE` version<sup>6</sup> as a GNU-zipped tar file. The default filename is `distribution-2.6.1-src.tar.gz` (I will use 2.6.1 in this example), save it into your `$HALLA_DIR`. To unpack, you can either use first GNU `unzip` (type `gunzip distribution-2.6.1-src.tar.gz`), and then extract the source files with `tar -xvf distribution-2.6.1-src.tar`, or directly using GNU-`tar` (type `tar -zxvf distribution-2.6.1-src.tar.gz`). On Linux systems, the binary `tar` is actually GNU-`tar`. You can check by typing `tar --version`. GNU-`tar` will give you the version

---

<sup>4</sup>`OSNAME` may be already defined by the system-wide login or initialization file. Check with `echo $OSNAME`.

<sup>5</sup>`ESPACE` reads the masses of particles from binary files. `ESPACE` searches for files called `$HALLA_DIR/db/masses_$OSNAME.dat`, with the shell environment variables `HALLA_DIR` and `OSNAME` expanded into their predefined values. The origin of those binary files is unclear, they were probably obtained from the “Particle Data Group”. Things would be much easier if this file would be a text file.

<sup>6</sup>At date of print, 2 is the major, 6 the minor version number and 1 the patch level.

number, the normal tar program does not understand `--version` and will produce a warning message, continued by a `Usage` line.

The espace sources you will find in `src/r2-6`. Additionally, you need to define `OSNAME`<sup>7</sup>.

**Compilation and installation** To compile, go into the directory `$HALLA_DIR/src/r2-6`. To actually compile ESPACE, you need several things:

- GNU-make. The standard make cannot parse the ESPACE makefile. However, the binary make on most systems is actually GNU-make. You can check with `make -v`. If make is not GNU-make you can try if you find `gnumake` or `gmake`. Compilation works with version 3.74 (installed at JLAB), but probably older versions too.
- CERNLIB. You may have it installed anyway if you use PAW.
- A VAX-style FORTRAN compiler which understands structures and unions. It has to be installed as `f77`. On SunOS, HP-UX and OSF1 such a compiler is ususally part of the installation. If not, bug your system manager. For Linux, there are two commercial compilers that may work. The Absoft FORTRAN F77 v4.4 compiler does work. The Portland Group `f77` compiler may work but was not tested.

Also, for Linux you have to take care which CERNLIB you are using and which Linux you actually have installed. See for reference the “Linux installation notes” you find on the ESPACE web page. Depending on your system you have to modify the master makefile (`$HALLA_DIR/src/r2-6/Makefile`). Simple modifications, and you will find instructions there. Different Linux distributions and versions come with different `f2c` libraries. Also, if you have a PENTIUM processor or less, you need to disable the PENTIUMPRO specific optimizations. If this is done, type `make espace` or simply `make`.

While you drink a coffee<sup>8</sup> ESPACE compiles everything, and finally you should have an output like

```
f77 -X -defsym -X MAIN_=main -X -defsym -X open_database_=open_database__
-X -defsym -X close_database_=close_database__ block_data_decode.o calc_char
ge.o cbuffchk.o cfp.o coda.o config_parse.o corrupt_synch.o detmap_datastrea
m.o det_sub_user.o espace.o filter.o filter_til_eof.o get_bcm.o get_epics_bp
m.o get_fpd_vertex.o get_pos.o get_wires.o index_range_user.o ini_rawdata.o
interpret_bpm_rast.o interpret_event_coin.o interpret_event_e.o interpret_ev
ent_h.o open_database.o option.o pulser_fired.o rawdata_til_eof.o read_argli
st.o read_cher_e.o read_cher_h.o read_fpp_h.o read_helicity_h.o read_scint_e
.o read_scint_h.o read_show_e.o read_spare.o read_vdc_e.o read_vdc_h.o scale
r_history.o scaler_update.o shcal.o superperiod_check.o tot_shower.o variabl
```

---

<sup>7</sup>For sunos, previous manuals requested to add the path to the MOTIF libraries to the shared library path, e.g. with `export LD_LIBRARY_PATH=/usr/dt/lib` for `bash`. This is not necessary anymore since the routines of ESPACE which require X-libraries are disarmed.

<sup>8</sup>Depending on your system, you should drink fast (few minutes) or take the time ( $\infty$ ).

```
e_user.o variable_user_init.o kludge.o ../espace_lib/block_data_const.o ../e
space_lib/block_data_var.o /home/kees/espace/lib/Linux/coolhands.o -L/home/k
ees/espace/lib/Linux -lespace -lkinematics -lspectra -lfcalc -lmbook4d -lplo
t -lstat -lutil -lvms -levio -L/app/cernlib/98egcs/lib -lgraflib -lgrafX11 -
lpacklib -L/usr/X11R6/lib -lXt -lX11 -lm -lU77 -lg77 -o espace
make[1]: Leaving directory '/home/kees/espace/src/r2-6/espace_halla'
```

and `ls -l espace_lib/espace` should show something like

```
-rwxrwxr-x  1 kees      kees      3697023 Mar 22 20:49 espace_halla/espace
```

If there is no `espace` binary with your actual system date, and the last lines of your compilation session does not look like the above, they even show some error messages, then something went wrong. Reasons can be:

**wrong or missing HALLA\_DIR:** The space sources are grouped (the term **organized** would be flattering) into several source directories. The master makefile loops over all those directories and calls local makefiles which actually do the compilation. Include files needed by the FORTRAN files are searched first locally, and if not found there in the directory path defined in the Makefile by `$FINCLUDE` (path includes currently only `$HALLA_DIR/src/r2-6`)<sup>9</sup>. Actually, the most lucky case is that `HALLA_DIR` is undefined. You will quickly get an error message like

```
f77 -I/src/r2-6 -N22 -N90 -B108 -f -o dynaparams dynaparams.f
FORTRAN 77 Compiler 4.4, Copyright (c) 1987-1997, Absoft Corp.
  error on line 13 of parameter.h: invalid INCLUDE
make[1]: *** [dynaparams.h] Error 1
make[1]: Leaving directory '/home/kees/espace/src/r2-6/espace_lib'
make: *** [subdirectories] Error 2
```

In this example, it indicates that the file to be included in line 13 of `parameter.h` (included by `dynaparams.f`) could not be found. As an experienced human compiler you will also notice that the directory after the `-I` flag is not correct, it should be `$HALLA_DIR/src/r2-6`, i.e. `/home/kees/espace/src/r2-6` and not `/src/r2-6`.

If `$HALLA_DIR` is defined but wrong the results are unpredictable. If there is a source distribution at that place, some of your include files will be included locally and some via the `$FINCLUDE` path. Also, the same include file may be included either locally or via the path, for different FORTRAN source files. Most likely something will be incompatible, and you will get at least some errors for some files. In worst case, compilation may be successful, you even get no runtime errors, but your results will be strange.

---

<sup>9</sup>Let's distinguish two cases. Files included via `#include <filename>` are searched locally (in `.`), those with `#include <directory>/<filename>` in the `$FINCLUDE` path.).

**wrong or missing PERL:** ESPACE uses the script `util_notsource/sfmakedepend` which parses through the entire FORTRAN sources and creates in each subdirectory the dependency files `.depend`. It does not matter if you don't know what this is all about. Important is that this script is written in the script language perl. Depending on your system manager it is installed in `/usr/bin`, `/usr/local/bin`, elsewhere or not at all. If you have the wrong path you will find an error message like that:

```
make[1]: /home/kees/espace/src/r2-6/util_notsource/sfmakedepend: Command
not found
make[1]: *** [.depend] Error 127
make[1]: Leaving directory '/home/kees/espace/src/r2-6/espace_lib'
make: *** [subdirectories] Error 2
```

Important is the first message ...`Command not found`. It may mean that the command `util_notsource/sfmakedepend` could not be found, but more probable the first line in that file specifies a wrong path to PERL. Check with `type perl` if and where PERL is installed, correct the first line of `util_notsource/sfmakedepend` and try again. If you don't have perl you should install it. As a workaround you can replace `util_notsource/sfmakedepend` by a simple file which contains two lines only:

```
#!/bin/sh
echo>.depend
```

This file creates a phony `.depend` file. Compilation will then work, and if you edit a FORTRAN source file make will still recognizes this (because those dependencies are declared in the Makefiles) and recompiles the file. However, if you edit an include file make has no track which FORTRAN source files need that and thus have to be recompiled, and thus does nothing. If you really decide for some dirty hack like that you should remove all non-source files with `make realclean` and recompile everything again. Get PERL!

Besides compiling the espace binary more files were created. You will find several libraries in `$HALLA_DIR/lib/$OSNAME`, each reflecting one of the source directories. Really useful is only one of them, `libevio.a`. It contains the routines needed to open and close a data file and extract the event headers and contents.

If the compilation was successful, you may want to install ESPACE. Type `make install`<sup>10</sup>. It will copy the ESPACE binary into `$HALLA_DIR/bin/$OSNAME` and the `masses_$OSNAME.dat` into `$HALLA_DIR/db`. Thus, after installation you will have four subdirectories in your `$HALLA_DIR`, `bin`, `db`, `lib` and `src`. To invoke ESPACE by simply typing `espace` you should add `$HALLA_DIR/bin/$OSNAME` to your environment path.

---

<sup>10</sup>Actually, `make espace` is a dependency of `make install`, thus it would have execute `make espace` if needed.

## 1.2.2 Getting ready

Before starting, you need a few additional files.

**detmap file:** You won't want to create this file by yourself. Usually, it is inherited from the preceding experiment. In the detmap file, usually called `detmap.config`, the physical detectors are described. It is specified by which "Read Out Controller" (ROC), crate, slot and channel a particular detector information can be accessed. The detmap file is analyzed by `ESPACE`, and the information is used by the event decoder to fill internal variables with the raw detector information. It is not really needed, because it is stored in every data file. If no detmap file is specified by the user, `ESPACE` will read this information and writes it to the current working directory as `detmap_temp.config`. However, if the information in that detmap file is not accurate, it has to be modified and `ESPACE` has to be instructed to use the correct file. The header of the detmap file gives enough information to modify it.

**database:** You don't want to create this file by yourself. Usually, it is inherited from the preceding experiment. The database contains two sets of information. At first, it contains the location of your detectors, going down to the position of each wire in the VDCs, for example. Additionally, it contains all the parameters which are assumed to stay constant for a certain hardware setup. Those parameters are, e.g.

- matrix elements which are used to transform from the focal plane to the target coordinate system.
- pedestals and gains of ADCs, coefficients for walk correction.
- offsets for TDCs.
- ranges of ADCs and TDCs, TDC resolutions.
- drift speeds in the VDCs, light propagation speeds and attenuation lengths in the scintillators.
- coefficients to convert drift time into distance.

See App. B for a complete reference.

**header file:** The header file contains information which may vary run by run, but are specific to that run. They are either created separately for each data file or, more clever, for one particular setting. See App. A for a full description of the header file. Parameters it contains are, e.g.

- spectrometer angle, momentum (dipole  $\vec{B}$ -field) and pointing offset.
- parameters specifying the target, describing its geometry, position, composition and thickness.
- beam energy.

- reaction to analyze.

Some parameters, like the reaction, are not compliant to the above mentioned. In a future ESPACE version, all parameters which are related to a specific analysis should be specified in kumac file. Examples are the reaction (one may want to analyze  $(e, e')$ ,  $(e, p)$  and  $(e, e'p)$  for the same run) or the question whether or not use the beam position monitors for beam reconstruction. Essentially, those parameters are flags specified by the user but no parameters describing the hardware.

**ESPACE kumac file:** Actually, now you would be able to start an ESPACE session and use all the commands described in App. D to analyze your data file. However, this is not very convenient. It is more practical to group all necessary commands in a script file which is then executed from within the ESPACE session. To parse this script file ESPACE is using the KUIP[1] interpreter. Thus, the files look like a kumac file and the extension is `.kumac`, but users should keep in mind that ESPACE and PAW kumac files are two different things.

For files taken before Dec 1, 1999 you need additionally:

**rastconsts.dat file:** The `rastconsts.dat` file is a weird way to implement the calibration of the fast raster and the beam position monitors (BPM). The BPM information is not synchronized with the event but shows a (fortunately constant) delay (phase slippage of the BPM versus raster). The `rastconsts.dat` file provides the tools to correct for that. Data files taken after Dec 1, 1999, contain the Struck 7510 ADC

Because the ESPACE kumac file is the heart of what the ordinary user has to program in order to analyze his data file, an example should be given.

```
***** define FILES *****
*
set/file/output carbon_1434.hbook
set/file/database db_845_e
set/file/header hdr_1434
set/file/detmap detmap.config
*
***** define CUTS *****
*
define/cut/1d spec_e.th_tg-1 -0.03 0.03
define/cut/1d spec_e.ph_tg-1 -0.06 0.6
define/cut/1d spec_e.p_kin-1 825.0 850.0
define/cut/1d spec_e.gas.adc-1 0.0 150.0
define/logical accept_e spec_e.th_tg-1&&spec_e.ph_tg-1
*
***** spectrometer E *****
```

```

*
set/hitbits (+spec_e.gas)
set/auto_window on
set/spectrum/bins/x1 1
spectra/save spec_e.s1.hp_r,spec_e.s2.hp_r
spectra/save spec_e.s1.adc_r,spec_e.s1.adc_l, _
                spec_e.s2.adc_r,spec_e.s2.adc_l accept_e
set/ntuple on
spectra/save spec_e.s1.adc_r/spec_e.s2.adc_r/spec_e.s1.adc_l/ _
                spec_e.s2.adc_l accept_e
set/ntuple off
set/auto_window off
set/spectrum/bins/x1 300
set/spectrum/window/x1 -0.15 0.15
spectra/save spec_e.th_tg-1,spec_e.ph_tg-1 accept_e&&spec_e.p_kin-1
set/spectrum/bins/x1 100
set/spectrum/window/x1 -0.025 0.025
spectra/save spec_e.y_tg-1
set/spectrum/bins/x1 250
set/spectrum/window/x1 825.0 850.0
spectra/save spec_e.p_kin-1 accept_e&&spec_e.gas.adc
*
***** scanning and output *****
*
file/scan carbon_1434.dat FIRST=1 LAST=10000
spectra/write
quit

```

In the first few lines the names of the hbook, database and header file are given. File names can be referenced either with an absolute path or a relative (descending from the current working directory) path. Here the detmap file is specified explicitly, overriding the detmap contained in the data file.

The next lines contain the definition of cuts on ESPACE variables. A complete list of valid ESPACE variables can be found in Sec. 2. Here are used variables like the ADC value of the gas Čerenkov in the electron spectrometer (`spec_e.gas.adc`) or the reconstructed angles of the scattered electron,  $\theta_{tg}$  (`spec_e.th_tg`) and  $\phi_{tg}$  (`spec_e.ph_tg`). Logical definitions can be defined by combining cuts and logical conditions defined previously. Here, the two cuts on the angles are combined to a cut on solid acceptance.

In the next part histograms and ntuples are defined through `spectra/save` commands of ESPACE variables. The first parameter is a list of ESPACE variables, the second (optional) parameter a logical or a logical condition. Histograms and ntuples in the list are separated by ','s, two-dimensional histograms ( $y/x$ ) and ntuples can be defined by separating ESPACE variables by '/'s. To store variables into a ntuple you actually have to set a flag with

`set/ntuple on` and return to histogram mode with `set/ntuple off`.

To start a scanning session type the command `espace`. The prompt `espace>` has appeared and commands can be typed in, or the `ESPACE` kumac can be executed by typing `exec <filename>`.

### 1.2.2.1 Runtime Errors

**unit 98** is the FORTRAN unit assigned to the masses-database file. Such an error is issued if this file cannot be found. The particular error message depends on the compiler, e.g. on SunOS you get a clear statement, from Absoft only something with a 98 in it. There are three steps necessary for the masses db file:

- `HALLA_DIR` has to be defined
- `OSNAME` has to be defined
- the masses db file was installed previously. By default, `ESPACE` installs it in `$HALLA_DIR/db`.

If everything is correct, the file `$HALLA_DIR/db/masses_${OSNAME}.dat` should exist and is readable. Check with `ls -l $HALLA_DIR/db/masses_${OSNAME}.dat`.



# Chapter 2

## Variables

As was pointed out in Sect. 1.1, the COOLHANDS subroutine `scansub` evaluates logical expressions and increments histograms by obtaining the variable values through the routines `variable` in the `ESPACE` library or `variable_user` in the experiment-specific `espace_halla` part. The different locations where the variable can be defined is not important for the usage of the program, however, for maintenance/changes of the program care should be taken where one installs them. If the variable is of interest to many users, it should be installed in the `ESPACE` library. However, if it is only useful for testing purposes, installation should be done in `variable_user`.

In the first section an overview is given of the variables that have already been installed in the `ESPACE` library, followed by a section giving instructions for implementing new ones in the code. Finally, it is shown how new variables can be defined on-the-fly in a command file.

### 2.1 Units of the Variables

Before discussing which variables are installed in `ESPACE` and how to install new ones, a few words about the unit definition in `ESPACE`. We have tried to express variables that are going in and out of the program in one consistent set of units, as shown in Tab. 2.1.

However, from Tab. 2.1 it is clear that for the angle variables, besides degrees, another unit is also used. The angles of the particle vertex as measured at the target or in the detector hut are actually tangents of angles, so dimensionless. Since the angles are fairly small, the tangents of an angle is close to the angle when expressed in radians.

### 2.2 Overview of Pre-Defined Variables

In this section we want to give an overview of the variables that are defined in the program and, therefore, can be histogrammed and used in the definition of logicals. Most of the variable names are preceded by a string indicating to which spectrometer and/or detector they belong. Figure 2.1 gives an overview of the variables and a tree-like structure guiding

Table 2.1: Set of units used for ESPACE variables

QUANTITY	PURPOSE	UNITS
length	target parameters	meters
angles	vertex angles at target and focal plane wrt. spectrometer central axis	degrees dimensionless
momenta		MeV/c
energy		MeV
field		kG
time		seconds

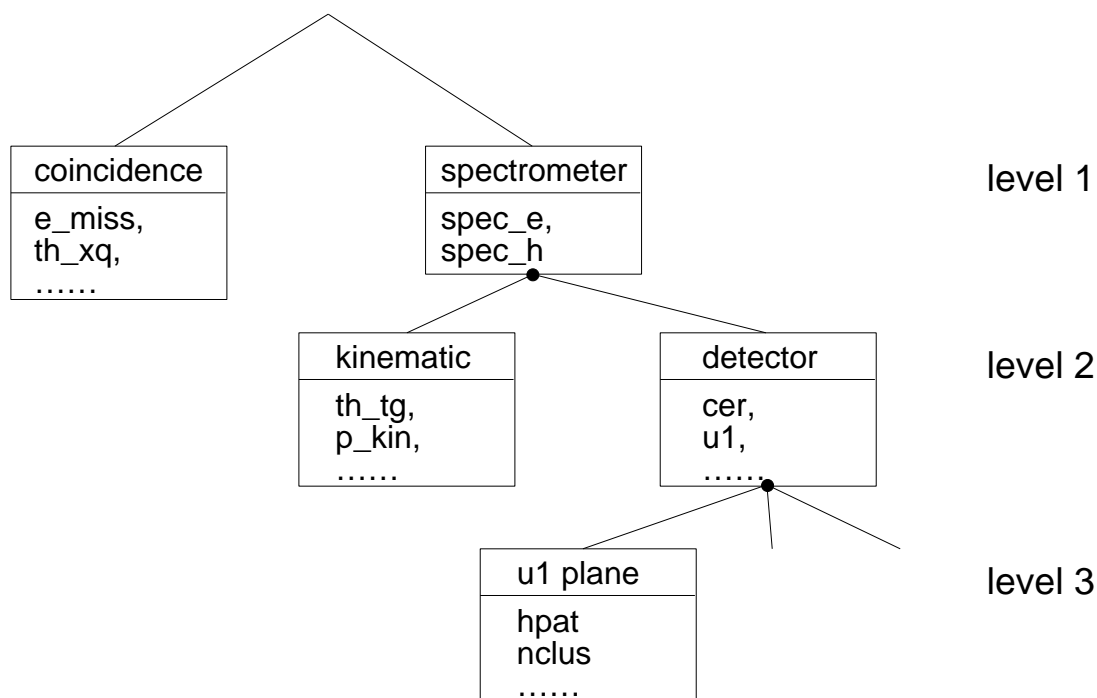


Figure 2.1: Structure guiding the user how to compose the variable string.

the user how to compose the variable string.

Starting at the top of the tree, the user has two options: choose a variable which is a function of the output of both spectrometers or select a variable linked to one of the spectrometers. In the first case there is no string, so the missing energy is simply referenced by `e_miss`. However, in the second case the string will start with the spectrometer name: `spec_e` or `spec_h`. The black dot at the bottom of the box indicates that the user should go at least one level deeper to have a valid variable name. The next choice is between

a kinematic variable or a variable connected to one of the detectors, e.g. `spec_e.p` (the momentum of the particle detected in the electron spectrometer) or `spec_e.s1`. In the last case the user has to go one level deeper, indicating the variable of that detector.

Below we will discuss four tables in which the variables are grouped according to the number of levels in the variable string. In Tab. 2.2, we start with the variables giving

Table 2.2: Level one variables, giving CODA and EPICS information.

variable	class	type	sub	dim	description
<code>clock_e*</code>	i	s	no	s	time since CODA start
<code>clock_h*</code>	i	s	no	s	time since CODA start
<code>run_number*</code>	i	s	no		CODA run number
<code>event_type*</code>	i	s	no		CODA event type
<code>event_nr*</code>	i	s	no		CODA event number
<code>cur_cav_e*</code>	r	s	no		current (unscaled) in the upstream (?) beam cavity
<code>cur_cav_h*</code>	r	s	no		current (unscaled) in the downstream beam cavity
<code>cur_cav_cal*</code>	r	s	no		current (calibrated) in Unser monitor???
<code>det_mon_e*</code>	r	s	no		electron spectrometer detector package rate (???)
<code>det_mon_h*</code>	r	s	no		hadron spectrometer detector package rate (???)

information about the CODA properties of the event. For each variable the parameters **class**, **type**, **size** and **sub** have been defined. The meaning of the parameters is the following:

**class** : The class of a variable is either real or integer. Its value will determine the result of the command `set/spectrum/bins`. In case of a real-variable, it means the number of bins of the histogram, otherwise the number of channels that will be grouped together.

**type** : The type of a variable is either scalar or array. If a variable has a type value of scalar, a histogram of this variable can only be incremented once for each CODA event. However, an array variable can be incremented multiple times. An example of a scalar variable is `event_type.tdc` (TDC values of drift times in a VDC wire plane) is an array variable and will contain the drift time of each wire exited by the tracks passing through this particular wire plane.

**sub** : Finally one can add to some variables a sub-detector number in order to be more specific when defining a cut or a histogram of a variable. For example, one could histogram the `tdc` times of wires 100 up to 200 of the `u1` plane in spectrometer `spec_e` with the command `spectra/save spec_e.u1.tdc[100;200]`. The command `spectra/save spec_e.u1.tdc` would have saved the `tdc` spectrum of the whole `u1` plane. Table 2.3 gives a summary of what the meaning of the sub detector number is for the different detectors.

Table 2.3: Meaning of sub-detector numbers for the variables of different detectors.

detector	sub-detector
wire plane	wire number
scintillator	paddle number
Čerenkov	mirror number
shower	block number

Tables 2.4, 2.5 and 2.6 list the level-one variables. Contained are in Tab. 2.4 the variables which are independent of the spectrometers, in Tab. 2.5 which are reconstructed with the electron spectrometer only, and in Tab. 2.6 which are reconstructed with both spectrometers.

Table 2.4: Level one variables, involving no spectrometer.

variable	class	type	sub	dim	description
pb	r	s	no	MeV	beam energy, after energy loss correction
pb_loss	r	s	no	MeV	beam energy-loss
rastadcr	i	a	yes		current in the raster deflection magnets, read using the burst mode Struck 7510 ADC. The field has a dimension of 32, the storage is equivalent to a fortran (4,8) field, i.e. the first eight field elements contain the readings for x, the next y, and finally the signs (high/low) of the derivatives of x and y. E.g., rastadcr[20:20] is the fourth reading of the derivative of x. Please be aware that in the current configuration per trigger only six values are read.
bpmaadcr bpmbadcr	i	a	yes		readings of the BPM antennas, read using the burst mode Struck 7510 ADC. The fields have a dimension of 32, the storage is equivalent to a fortran (4,8) field. The first eight field elements contain the readings for $X_p$ , the following $X_m$ , $Y_p$ , and $Y_m$ . Please be aware that in the current configuration per trigger only six values are read.
rastadcc bpmaadcc bpmbadcc	i	a	yes		rastadcr, bpmaadcr, and bpmbadcr, corrected for pedestals

continued on next page ...

Table 2.4: ... continued from previous page ...

variable	class	type	sub	dim	description
bpmxrot bpmyrot bpmbxrot bpmbrot	r	a	yes	m	fields of length 6, containing the beam positions in the BPMS, in the rotated coordinate system
bpmxraw bpmyraw bpmbxraw bpmbraw	r	a	yes	m	fields of length 6, containing the beam positions in the BPMS
afitxpos afitypos bfitxpos bfitypos	r	i	no	m	position from the fit to the burst mode readings
afitxamp afityamp bfitxamp bfityamp	r	i	no	m	amplitude from the fit to the burst mode readings
afitxphs afityphs bfitxphs bfityphs	r	i	no	m	phase from the fit to the burst mode readings
bpma_x bpma_y bpmb_x bpmb_y	r	s	no	m	beam position in the BPMS, corrected for the phase shift
beam_x beam_y	r	s	no	m	beam position at the Hall center (practically the target), derived from the BPM positions
beam_ph beam_th	r	s	no	rad	tangents of the beam angles in the HLCS (see Sec. 4.1.1 on p. 37)
rast_x rast_y	r	s	no	m	beam position, derived from the raster current
rast1adc rast2adc	i	s	no		current in the raster deflection magnets, in x (1) and y (2) direction, read with VMIC3123 ADC
rast3adc rast4adc	i	s	no		derivative of the rast1adc and rast2adc signals, same definition as for rastadc derivatives
rast5adc rast6adc	i	s	no		current in the raster deflection magnets, in x (1) and y (2) direction, read with LeCroy1182 ADC
continued on next page ...					

Table 2.4: ... continued from previous page ...

variable	class	type	sub	dim	description
rast_adc	i	a	yes		raster adc values, with the same meaning as <i>rast_adc</i>
rast2_x rast2_y	r	s	no	m	raster amplitude, derived from the LeCroy1182 read-out
bpma1adc bpma2adc bpma3adc bpma4adc	i	s	no		beam position monitor cavity IPM1H03A read-out
bpmb1adc bpmb2adc bpmb3adc bpmb4adc	i	s	no		beam position monitor cavity IPM1H03B read-out
rastxphs rastyphs	r	s	no	rad	absolute phase of the horizontal/vertical raster
bpmaxphs bpmayphs	r	s	no	rad	absolute phase of the read-out in IPM1H03A, horizontal/vertical
bpmbxphs bpmbypshs	r	s	no	rad	absolute phase of the read-out in IPM1H03B, horizontal/vertical

Table 2.5: Level one variables, involving one spectrometer.

variable	class	type	sub	dim	description
omega	r	s	no	MeV	energy transfer
q	r	s	no	MeV/c	momentum transfer
q2	r	s	no	$(\text{MeV}/c)^2$	four-momentum transfer squared
w2	r	s	no	$(\text{MeV}/c^2)^2$	Mandelstam variable s
mandelt	r	s	no	$(\text{MeV}/c^2)^2$	Mandelstam t
mandelu	r	s	no	$(\text{MeV}/c^2)^2$	Mandelstam u
scattang	r	s	no	deg	electron scattering angle
epsilon	r	s	no		virtual photon polarization

Table 2.6: Level one variables, involving both spectrometers.

variable	class	type	sub	dim	description
e_miss	r	s	no	MeV	missing energy (in the definition of the old electron scatterer)
continued on next page ...					

Table 2.6: ... continued from previous page ...

variable	class	type	sub	dim	description
emiss	r	s	no	MeV	missing energy (energy of the missing particle)
p_miss	r	s	no	MeV/c	missing momentum
p_missx	r	s	no	MeV/c	
p_missy	r	s	no	MeV/c	
p_missz	r	s	no	MeV/c	
m_miss	r	s	no	MeV/c <sup>2</sup>	missing mass
m_miss2	r	s	no	(MeV/c <sup>2</sup> ) <sup>2</sup>	missing mass square
tc	r	s	no	TDC chan	coincidence time
tc_cor	r	s	no	TDC chan	coincidence time corrected for path length difference wrt. the central rays
ts_tdc	i	a	yes	TDC chan	TDC time of the first hit in the multi-hit TDC
ts_patt	i	s	no		hitpattern of the multi-hit TDC
ts_mult	i	a	yes		multiplicity of channel $i$ in the multi-hit TDC
tstdcall	i	a	yes	TDC chan	all tdc times of all hits in the multi-hit TDC. The $j^{\text{th}}$ hit in the $i^{\text{th}}$ channel is at position $6*(i-1)+j$ .
th_xq ph_xq	r	s	no	deg	spherical angles between momentum of particle x and the momentum transfer in the reaction $A(e, e'x)B$ . The z-axis is defined by the momentum transfer and the x-z plane by the electron scattering plane.
th_xq_cm ph_xq_cm	r	s	no	deg	same angles, but in the c.m. system
th_bq ph_bq	r	s	no	deg	spherical angles between momentum of particle B and the momentum transfer in $A(e, e'x)B$
th_bq_cm ph_bq_cm	r	s	no	deg	same angles, but in the c.m. system
react_x	r	s	no	m	x-coordinate of reaction point in the target given in the laboratory frame
react_y	r	s	no	m	y-coordinate
react_z	r	s	no	m	z-coordinate
continued on next page ...					

Table 2.6: ... continued from previous page ...

variable	class	type	sub	dim	description
mlu_in1* mlu_in2* mlu_in3* mlu_in4* mlu_in5*	r	s	no		coincidence event rate into the different channels of the memory lookup unit MLU. For coincidences, only channel 5 is meaningful.
mlu_out*	r	s	no		coincidence event rate coming out of the MLU
dead_time*	r	s	no		dead time in coincidence events

A '\*' behind the variable name of the Tabs. 2.2, 2.5, 2.6, 2.7 and 2.8 indicates that the histogram of this particular variable is updated even when the necessary detectors did not fire. For instance, the number of hits in a scintillator would be updated even when this detector did not fire. For level-three variables which detector had to fire is obvious from the variable string. Level-two variables are a bit more complicated variables which are composed out of the response of several detectors. Here, the histogram is updated when the trigger detector of the spectrometer fired. For level-one variables either the trigger detectors of both spectrometers should have fired in case of coincidence variables or no detector is necessary at all like for the variable `event_type`.

Table 2.7: Level two variables, involving one spectrometer.

variable	class	type	sub	dim	description
omega	r	s	no	MeV	energy transfer
q	r	s	no	MeV/c	momentum transfer
q2	r	s	no	$(\text{MeV}/c)^2$	four-momentum transfer squared
w2	r	s	no	$(\text{MeV}/c^2)^2$	Mandelstam s
reactx	r	s	no	m	x-coordinate of the crossing of the beam and one spectrometer vertex
reacty	r	s	no	m	y-coordinate
reactz	r	s	no	m	z-coordinate
twoarm_x	r	s	no	m	x-coordinate of the crossing of the two spectrometer vertices
twoarm_z	r	s	no	m	z-coordinate
xeb_xhb	r	s	no	m	spec_e.reactx-spec_h.reactx
xeb_xeh	r	s	no	m	spec_e.reactx-twoarm_x
xhb_xeh	r	s	no	m	spec_h.reactx-twoarm_x
zeb_zhb	r	s	no	m	spec_e.reactz-spec_h.reactz
zeb_zeh	r	s	no	m	spec_e.reactz-twoarm_z

continued on next page ...

Table 2.7: ... continued from previous page ...

variable	class	type	sub	dim	description
z_hb_zeh	r	s	no	m	spec_h.reactz-twoarm_z
x_rot	r	s	no	m	dispersive position x of the “golden track” in the u1 plane in the rotated c.s. <sup>1</sup>
th_rot	r	s	no		tangent of angle $\theta$ of the particle track in the focal plane in the dispersive direction wrt. the local central ray
y_rot	r	s	no	m	non-dispersive position y in the u1 plane in the rotated c.s.
ph_rot	r	s	no		tangent of angle $\phi$ of the particle track in the non-dispersive plane at the focal plane
xr_abr	r	s	no	m	dispersive position <b>x_rot</b> corrected for spectrometer aberrations
xr_kin	r	s	no	m	dispersive position <b>x_rot</b> corrected for spectrometer aberrations and for deviations of the nuclear recoil momentum <sup>2</sup>
x_tra th_tra y_tra ph_tra	r	s	no		coordinates of the golden track defined as above but now in the transport c.s.
beta	r	s	no		$\beta$ of the particle, as calculated by the time-of-flight between s1 and s2.
beta_ratio	r	s	no		beta, divided by the $\beta$ as derived from the particle momentum.
p	r	s	no	MeV/c	momentum of the particle after exiting the target corrected for spectrometer aberrations
px	r	s	no	MeV	momentum of the particle, after energy loss correction
px_loss	r	s	no	MeV	particle momentum-loss
p_kin	r	s	no	MeV/c	momentum of the particle after exiting the target corrected for spectrometer aberrations and for deviations of the nuclear recoil momentum
dp	r	s	no		like p but relative to the central ray
continued on next page ...					

<sup>1</sup>The golden track is defined as the track having a `tzero_av` closest to zero, therefore, most likely being the particle track that made the trigger.

<sup>2</sup>This correction is different for electrons and protons. It makes only sense to use it for the hadron arm in case of  $H(e, e'p)$ .

Table 2.7: ... continued from previous page ...

variable	class	type	sub	dim	description
dp_kin	r	s	no		like <code>p_kin</code> but relative to the central ray
dp_cor	r	s	no		momentum relative to the central ray corrected only for spectrometer aberrations
th_tg	r	s	no		tangent of angle $\theta_{tg}$ of the particle track in the dispersive plane at the scattering point at the target
ph_tg	r	s	no		tangent of angle $\phi_{tg}$ of the particle track in the non-dispersive plane at the scattering point at the target
y_tg	r	s	no		the non-dispersive position <code>y_tg</code> at the scattering point at the target
x_sieve	r	s	no	m	dispersive position at the position of the collimator
y_sieve	r	s	no	m	non-dispersive position at the position of the collimator
path_diff	r	s	no	m	difference in path length between this track and the central ray through the spectrometer
track*	i	s	no		number of reconstructed particle tracks
tzero_av*	i	s	no	TDC chan	average time offset of TDC times in the clusters of the different planes that are in the track
mlu_in1* mlu_in2* mlu_in3* mlu_in4* mlu_in5*	r	s	no		single event rate into the different channels of the memory lookup unit MLU; only channels 1 and 2 are meaningful for <code>spec_e</code> and 3 and 4 for <code>spec_h</code> .
mlu_out*	r	s	no		pre-scaled single event rate coming out of the MLU
dead_time*	r	s	no		dead time in pre-scaled single

Table 2.8: Level three variables, involving detectors in the spectrometers. `spec_s` can be either `spec_e` or `spec_h`.

variable	class	type	sub	detector	dim	description
hit_type*	i	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		hitpattern of wire plane, see description in section ??
clus	i	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		number of clusters in the wire plane
tdc	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.aero spec_s.gas	TDC chan	TDC values of drift times in wire plane or of photo multipliers in Čerenkov
tdc_c	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	TDC values of drift times in wire plane, corrected for COMMON STOP shifts due to path length differences to the time defining scintillator and for drift to path length conversion
tdc_cor	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	correction to TDC values of drift times in wire plane
tdc_diff	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	drift time difference determined by comparing measured drift time to the distance between wire plane and global fit divided through the drift velocity obtained from the database
wire	i	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		wire numbers that fired in wire plane

continued on next page ...

Table 2.8: ... continued from previous page ...

variable	class	type	sub	detector	dim	description
coars	i	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		wire number with the shortest drift time in each cluster
slope	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		slope of track through wire plane, determined for each cluster
pos	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	m	position where track passed through the wire plane given in the local wire plane coordinate system, determined for each cluster
tzero	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	time offset of TDC times in cluster
chisq	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		reduced $\chi^2$ of fit to the drift times in the wire plane for each cluster
drift	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	m/s	????reduced $\chi^2$ of fit to the drift times in the wire plane for each cluster
angle_diff	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		difference in slope as determined from the drift times in one plane and two planes

continued on next page ...

Table 2.8: ... continued from previous page ...

variable	class	type	sub	detector	dim	description
mult*	i	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.s1 spec_s.s2 spec_s.aero spec_s.gas spec_e.preshow spec_e.show spec_h.s3		number of hits on detector
hp_l* hp_r*	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3		paddle hit pattern for photo multipliers on the left/right side
adc	i	a	yes	spec_s.aero spec_s.gas spec_e.preshow spec_e.show	ADC chan	ADC channel of Čerenkov mirrors or (pre)shower blocks
adc_c	i	a	yes	spec_s.aero spec_s.gas spec_e.preshow spec_e.show	ADC chan	ADC channel of Čerenkov mirrors or (pre)shower blocks corrected for pedestal and gain,
adc_l adc_r	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	ADC chan	ADC channel of scintillator paddles on the left/right side
adc_l_c adc_r_c	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	ADC chan	ADC channel of scintillator paddles on the left/right side corrected for pedestal and gain
adcsun_t	r	s	no	spec_s.gas spec_s.aero		sum of all ADC channels, corrected for pedestal and gain
tdc_l tdc_r	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	TDC chan	TDC channel of scintillator paddles on the left/right side
continued on next page ...						

Table 2.8: ... continued from previous page ...

variable	class	type	sub	detector	dim	description
tdc_l_c tdc_r_c	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	TDC chan	TDC channel of scintillator paddles on the left/right side, corrected for timing offset
x_det y_det	r	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.s1 spec_s.s2 spec_s.aero spec_s.gas spec_e.preshow spec_e.show spec_h.s3	m	coordinates of crossing point of VDC track with detector plane, given in the detector's local coordinate system and manipulated if the particle track exceeds the detector dimensions specified in detmap.config
xdetgeom ydetgeom	r	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.s1 spec_s.s2 spec_s.aero spec_s.gas spec_e.preshow spec_e.show spec_h.s3	m	coordinates of crossing point of VDC track with detector plane, given in the detector's local coordinate system
rate	r	a	yes	spec_s.aero spec_s.gas spec_e.preshow spec_e.show		count rate in mirrors and blocks, resp.
rate_l rate_r	r	a	yes	spec_s.s1 spec_s.s2 spec_h.s3		count rate in paddles on the left or right side, resp.

A word of caution is necessary for the usage of variables that give the rates of the scalers in the experimental setup. Realize that the whole histogramming procedure in `ESPACE` is event-interrupt driven. So if you would save a one-dimensional histogram of, for instance, the coincidence rate in paddle 3 of scintillator s1 and its value is always 100 Hz and 1000 events are scanned, you will end up with a sharp spike at 100 containing 1000 entries. This

is probably not very useful. If you are interested in the time evaluation of this rate, its better to save a 2-dim ntuple of rate versus time.

## 2.3 Coding New Variables

In this section we discuss how to install in the code a new variable. It is assumed that the new variable is installed in the `ESPACE` library. Installation in the `espace_halla` part would proceed along the same lines. The routines that should then be modified will have the same names as the ones shown below, however, the extension `user` has to be added.

In our example, we want to install the ADC value of the left side of the scintillator paddle in both spectrometer `spec_e` and `spec_h`. Realize that per interrupt the scintillator's left side could have more than one entry and that one would like to specify the paddle number of the scintillator. In other words, this variable will be an array and could be specified with a sub-detector number. The consequence is that the last two routines below (`index_range.f`, for array variables and `det_sub.f`, for variables with sub-detector numbers) need to be modified.

The following routines have to be changed in the `ESPACE` library:

`block_data_var.f` : In this file the string array `var_txt` is initialized. It contains all the necessary parameter input for each variable. Each variable should have an entry of one line in the data statement:

```
data (var txt(i),i=1,199)
.....
> 'adc_1 5-6,21-22,49 none none 0.0 4095.0 i a y',! 49
.....
```

The description of these entries (possible values between square brackets):

name [*alphanumeric characters including '\_' and '-'*] Variable name.

detector [*numeric characters or all or none*] String of numbers indicating which detectors have access to this variable. A '-' in the string indicates a 'from-to', so 5-7 means 5,6,7. The correspondance between number and detector string is given by the array `det_txt`, also initialized in `block_data_var.f`. As you can see in Tab. 2.8 with the `adc_1` entry, the string 5-6,21-22,49 corresponds to detectors `spec_e.s1`, `spec_e.s2`, `spec_h.s1`, `spec_h.s2` and `spec_h.s3`.

optimize [*numeric characters or all or none*]

fire string [*numeric characters or all or none*] String of numbers pointing at detectors that should have fired in addition to the one whose variable is asked for. This is a useful feature if a variable is referenced that has a detector in both spectrometers, like all variables of Tab. 2.6.

range [numeric characters] Default range of the variable which is used when the command `set/auto-window on` was given.

class [i,r] integer- or real-variable

type [s,a] scalar or array variable

sub [y,n] sub-detector number allowed, y(es) or n(o)

variable.f: The function `variable.f` should return the value of the variable pointed at by `ispec` and `det txt`. `ispec` corresponds to the entry number in the array `det_txt`, see routine `block_data_var.f`.

`indx` is the hit number on the detector and its value is given by the routine `det_sub.f`, see below.

```

.....
else if (ivar.eq.49) then
  if (ip_sc.ne.0) then
    ip_pad = spdetector.sp(ispectro).trigger.
>          sc(ip_sc).padnr(indx).adc.left
    ip_hit = spdetector.sp(ispectro).trigger.
>          sc(ip_sc).hitnr(indx).adc.left
    dvar = dble(spdetector.sp(ispectro).trigger.
>          sc(ip_sc).pad(ip_pad).val(ip_hit).adc.left)
  endif
else
.....

```

In the example given above, first the pointers to the correct scintillator paddle and position in the array containing the different hits (`ip_pad` and `ip_hit`) have to be determined. Then the variable `dvar` is filled.

`index_range.f` Changing this routine is only necessary for array variables. This subroutine should return through `imax` the number of entries for the variable pointed at by `ivar`.

```

.....
else if (ivar.eq.49) then
  if (ip_sc.ne.0) then
    imax = spdetector.sp(ispectro).trigger.sc(ip_sc).nhits.adc.left
  endif
  imax = max(1,imax)
else
.....

```

The example above shows how `imax` is set if `ivar` equals 49.

det\_sub.f This routine has to be changed only for variables with subdetector indices. It should return the sub-detector number pointed at by `ivar` and `indx`.

```

.....
else if (ivar.eq.49) then
  if (ip_sc.ne.0) then
    det_sub = spdetector.sp(ispectro).trigger.
>          sc(ip_sc).padnr(indx).adc.left
  endif
else
.....

```

## 2.4 Defining New Variables

Instead of coding a variable into `ESPACE`, a new one can also be defined on the fly through the command interface. For example, suppose the missing mass has to be defined for the reaction  $H(e, e'p)$ . This would be done through the following steps:

```

espace > m_proton=938
espace > e0 = 845
espace > omega:e0-spec_e.p
espace > tx:sqrt(spec_e.p**2+m_proton**2)-m_proton
espace > m_miss:sqrt((omega-tx)**2-p_miss**2)

```

Notice that constants are assigned through an equal sign: `m_proton=938`. Variables are assigned through a colon: `omega:e0-spec_e.p`. After this assignment the new variable can be used, like the coded ones, in logical statements, histograms and ntuples.

Before summarizing the syntax for these assignments, notice the following restriction:

In the definition of a new variable, only scalar variables can be used.

New variables are defined by combining existing variables through the following operations:

( ) + - \* / \*\*

and intrinsic functions

```

abs acos acosd asin asind atan atand cos
cosd cosh deg exp log log10 rad sin
sind sinh sqrt tan tand tanh

```

The definition string should look like, `NEW : operation(OLD1,OLD2,...)`. Notice the assignment with `'.'`. Constants are defined by a string like, `PI=3.142`. Notice the assignment with `'='`.



# Chapter 3

## Defining Histograms, Logicals and Filters

In the previous section was described how to access a particular variable. What still has to be explained is how to histogram that variable and how to use it in the defining a condition.

### 3.1 Histograms

A histogram is defined with the command:

```
spectra/save variable_string condition_string
```

The `variable_string` can be any variable or combination of variables (for multi-dimensional histos and ntuples) as defined in Tabs. 2.2, 2.5, 2.6, 2.7 and 2.8 The condition string string is discussed in the next section.

A multi-dimensional histogram or ntuple (give command `set/ntuple on`) is created by making a variable string like `var2/var1`. For a histogram this has to be limited to two variables, an ntuple can contain upto 12 variables. The first variable mentioned in the `variable_string` is the last one in the histogram, so `var2/var1` will result in a 2-dim histogram with `x=var1` and `y=var2`.

A group of variables can be saved several times with different logical conditions. In this case the user has to supply at the end of the variable string either a counting index or leave it up to the program ESPACE to increment it internally. For instance,

<pre>spectra/save spec_e.dp condition1</pre>	or	<pre>spectra/save spec_e.dp-1 condition1</pre>
<pre>spectra/save spec_e.dp condition2</pre>		<pre>spectra/save spec_e.dp-2 condition2</pre>

will have the same result, however in the second case the user has explicitly given the counter numbers and has, therefore, its values under control. As shown in the previous chapter, a variable has several parameters which can influence the `spectra/save` command. Before histogramming a variable one should check whether its `class` is real or integer. In the

example below the command `set/spectrum/bins/x1 10` makes a histogram of the real-variable `spec_e.th_tg` containing 10 bins between 100.0 and 200.0 while for the integer-variable `spec_e.u1.tdc` it means that a histogram running from 100 to 200 should be made with a bin size of 10.

```
set/spectrum/window/x1 100 200
set/spectrum/bins/x1 10
spectra/save spec_e.th_tg
```

or

```
set/spectrum/window/x1 100 200
set/spectrum/bins/x1 10
spectra/save spec_e.u1.tdc
```

The `sub-detector` parameter for the variable allows to specify a subset of the detector for which the variable should be stored (see Sec. 2.2 for the meaning of `sub-detector` for the different detectors). The syntax for the `sub-detector` range is

```
spectra/save var[start;stop] condition
```

This can be used to create an ADC spectrum for the read-out of every paddle on the right side of scintillator `s1` (notice the usage of the do-loop as implemented in the KUIP command interpreter):

```
do i=1,6
spectra/save spec_e.s1.adc_r[[i];[i]] condition
enddo
```

## 3.2 Logicals

The definition of logicals can be done in two different ways:

- Define logicals by making cuts on the variables, either one- or two-dimensional.
- Define new logicals by combining the previously defined logicals through logical operators.

The syntax of the cut command in one dimension is

```
define/cut/1d variable string start_value stop_value
```

The syntax of the `variable_string` is exactly along the lines of defining a variable string in the `spectra/save` command. The logical value of the cut is `.TRUE.` if `start_value ≥ variable_string ≥ stop_value`.

In two dimensions a cut can be defined in two ways:

- Define a curve with a 2<sup>nd</sup>-order polynomial, and demand that a point is below or above it. The syntax is

```
define/cut/2d variable_string direction p0,p1,p2
```

Here the `variable_string` has the same syntax as for a 2-dim histogram, so `var2/var1`. The parameters `p0`, `p1` and `p2` are polynomial coefficients describing the line  $\text{var2} = p_0 + p_1 \cdot \text{var1} + p_2 \cdot \text{var1}^2$ . The parameter `direction` should have the value `above` ( $\text{var2} \leq p_0 + p_1 \cdot \text{var1} + p_2 \cdot \text{var1}^2$ ) or `below` ( $\text{var2} \geq p_0 + p_1 \cdot \text{var1} + p_2 \cdot \text{var1}^2$ ).

- Define a region with a polygon and demanding that the data point is inside the polygon. Syntax for this command is:

```
define/cut/polygon variable_string file_name
```

`file_name` is the name of a file containing the coordinates of the polygon corners. The numbers of points is limited to 20!

A way to make such a file is by plotting the appropriate 2-dim histogram or ntuple in PAW and then execute the following lines in PAW:

```
vector/delete x,y
vlocate x y -LPS
write x,y file_name '(g13.7,2x,g13.7)' '0C'
```

This command sequence will generate a file called `file name` in which the `x`- and `y`-coordinates are written on each left-button mouse-click. It stops after a right-button mouse click.

In both 1-dim and 2-dim `cut` commands one is allowed to use variables of type `array`. This can make cuts more specific, e.g. `define/cut/1d spec_e.s1.adc_r[2;2] 100 200` is only `.TRUE.` when paddle 2 of `s1` has a value between 100 and 200. A statement like `define/cut/1d spec_e.s1.adc_r 100 200` is `.TRUE.` if any of the paddles had a value between 100 and 200. New logicals can be defined by combining existing ones through logical operands. A list of the implemented operands is given in Tab. 3.1. The order of

Table 3.1: Logical operands allowed in ESPACE.

symbol	operation
!	negation
&&	AND
	OR

operations can be guided by brackets. For the moment, a logical definition containing an `OR` should be surrounded by quotes because the KUIP command interpreter will otherwise stop reading at the appearance of a `—` character. It is important to realize that logical definitions are expanded only upon execution, so during scanning an event file. For example,

```
define/logical new 'old1 && old2'
spectra/save spec_e.th_tg-1 new
```

followed by

```
define/logical new 'old1 || old2'
spectra/save spec_e.th_tg-2 new
```

will result in two `th_tg` spectra stored under identical conditions, namely the logical OR of `old1` and `old2`.

### 3.3 Filters

A filter is defined with the command:

```
file/filter file_name condition_string ,
```

where file name is the file into which the data will be stored if it fulfills `condition_string`. Currently the filtered data is stored again in CODA format so that it can be replayed with the same input routines as the raw data. However, the user can change that; all it takes is a change of the routines in the file `filter.f` in the `espace_halla` package. The current version is shown below:

```
      subroutine filter_event(ihandle)
c
c-----
c
      implicit none

      include 'coda.h'

      integer ihandle
      character filename*(*)
      integer evopen
      integer istatus

c Write CODA event

      call evwrite(ihandle,evbuffer)
      return

      entry filter_open(ihandle,filename)
```

c Open CODA file

```

    istatus = evopen(filename,'w',ihandle)

    if (istatus.ne.0) then
        call derror('FILTER_OPEN: error opening CODA output file')
    endif

    return

    entry filter_close(ihandle)

```

c Close CODA file

```

    call evclose(ihandle)

    return
end

```

The file should contain the routines

`filter_open` This routine opens the filtered data file.

`filter_event` This routine is called for each event that fulfills the filter condition. In it is defined what data is stored and in which format.

`filter_close` The closing of the data file is performed here.

If the user prefers a format different from CODA and wants to be able to replay it later again with `ESPACE`, the input routines, controlled by `filter_til_eof.f` (`espace_halla`) should be changed too. When scanning the file, one should not forget to give the `f` option with the `scan` command so that `filter_til_eof.f` controls the input instead of `rawdata_til_eof.f`.

## 3.4 Global Conditions

In the previous sections was discussed how to provide selection criteria when saving histograms or defining filters through the `command_string`. It is also possible to define more global conditions, meaning that it is applied to all following commands. This global condition is defined with the command:

```
set/hitbits (+detector1,-detector2,...) ,
```

where one can fill in the list any `detector` as mentioned in Tab. 2.8. Each `detector` should be preceded by a `+` or `-`. A `+` means that the condition is only `.TRUE.` if the `detector` has fired while a `-` means just the opposite, so `.TRUE.` if the `detector` did not fire. The total condition is the obtained by AND'ing the result of each detector in the list.

### 3.5 When is What Stored?

For calculating absolute numbers, it is important to understand what is stored in a histogram or the filter file when or a condition is not fulfilled or a detector does not fire. If the condition for a particular operation in `ESPACE` is not fulfilled, then it will not be executed for that event. So if the operation was histogramming a variable, saving data in a filter file, accumulating data for an optimization, etc., it will not be performed for this event. After a global condition all further operations are skipped up to the point where a new global condition was defined.

If a detector did not fire either of the following can happen:

- The variable of this detector is well defined even when the detector did not fire. For example, the multiplicity of the detector event, being zero. These variables have an `'*` in Tabs. 2.2, 2.5, 2.6, 2.7 and 2.8 behind their name.
- The variable of the detector is undefined, `ESPACE` puts it on a very large number ( $10^9$ ).

In either case, the number of entries in the histogram/ntuple should reflect the number of times the condition was fulfilled. For variables not directly linked to one particular detector (so level-one and level-two) it is not immediately clear which detectors should be checked. For the moment the following convention has been adapted. Level-two variables are linked to the trigger detectors (`s1` and `s2`) in the appropriate spectrometer (see Tab. 2.7 while part of the level-one variables are linked to the trigger packages in both spectrometers, see Tab. 2.6, some depend on the electron spectrometer only (Tab. 2.5), and the rest is completely detector independent, see Tab. 2.2.

# Chapter 4

## Coordinate Systems

### 4.1 Definition of Coordinate Systems

In this section five coordinate systems are discussed. One coordinate system is connected to the hall while a second one is a coordinate system in the hall that is connected to each of the magnetic spectrometers. Coordinate systems three, four and five are linked to the detector packages in the focal planes of the spectrometers.

#### 4.1.1 Hall A Laboratory Coordinate System

The origin of the **hall A laboratory coordinate system (HLCS)** is defined by the intersection (or at least the point of closest approach) of the non-rastered electron beam and the axis of rotation of the solid-target system. The non-rastered electron beam direction is defined by the last two beam-position monitors. The incident unscattered beam leaves the target along the positive  $z$  direction. The  $y$ -axis is pointing upward as defined by gravity.

Angles are defined with respect to this origin and a ray pointing along the positive  $z$ -axis. Convenient is the use of geographical angles. Here, the angle  $\theta$  is 0 along  $z$  and covers a 0 to 180 deg range in the  $x$ - $z$  plane.  $\phi$  is the out-of-plane angle formed between the vector under consideration and its projection on the  $x$ - $z$  plane. In contrast to the spherical angle system, geographical angles can be added and subtracted, see Sec. 4.2.3. In App. C an overview is given of available algorithms that translate between the different systems.

#### 4.1.2 Spectrometer Reconstructed Coordinate System

The origin of the spectrometer reconstructed coordinate system is defined as a point at a distance of 1.25 m from center of the central sieve-slit hole. The line going from the origin to the center of this hole is perpendicular to the sieve-slit surface and defines the  $z$ -axis. The  $x$ -axis is the line going through the center sieve-slit holes and points downward. Under optimal circumstances this origin should coincide with the origin of the hall A laboratory coordinate system and the center of rotation of the spectrometer. The  $x$ - $z$  plane and the  $y$  axis of the hall system should be parallel.

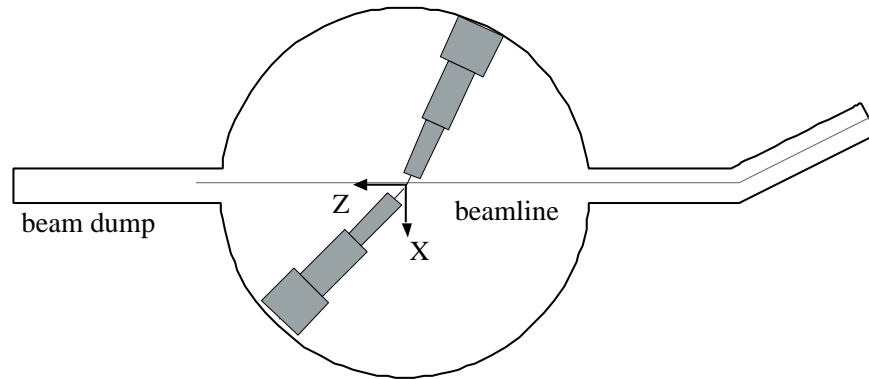


Figure 4.1: Hall A laboratory coordinate system.

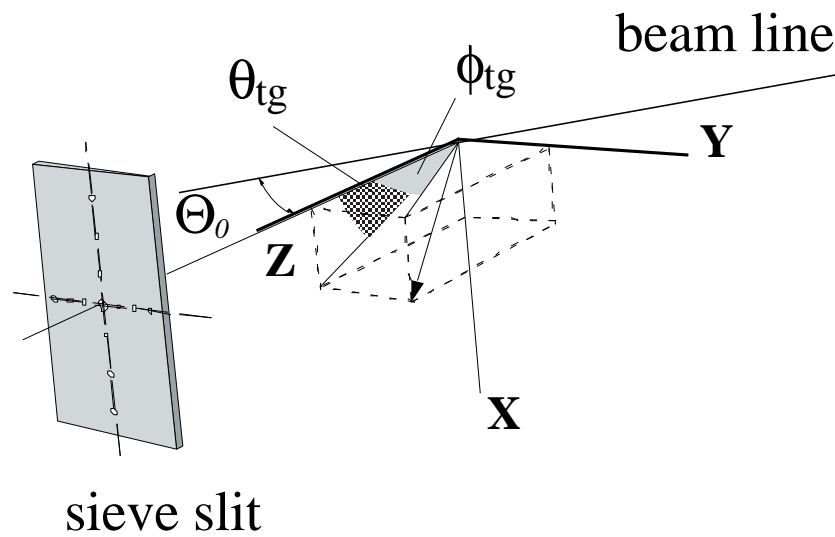


Figure 4.2: Spectrometer reconstructed coordinate system

### 4.1.3 Spectrometer Detector Coordinate System

The origin of the Spectrometer Detector Coordinate System (SDCS) is defined by the intersection of wire 184 in the first wire plane and the projection on the first wire plane of wire 184 in the second plane (assuming 368 wires in each plane). The z axis is perpendicular to the wire planes and its direction is fixed by demanding that its product with the central spectrometer ray (see Sec. 4.1.4) is larger than zero. The x axis is defined as the projection on the first wire plane of the vector difference between the the spectrometer central ray and a ray for which the momentum has been increased by an infinitesimal amount. Its direction is fixed by requiring an increase in momentum. It would be optimal if the x-z plane would coincide with the spectrometer symmetry plane. In Sec. 4.2 we will discuss how these coordinates can be calculated with the wire chamber information and how corrections can be performed for misalignments of the detector package.

### 4.1.4 Spectrometer Focal-Plane Coordinate System

The Spectrometer Focal-Plane Coordinate System SFPCS shares its origin with the detector system and the x-z planes coincide. However, its z-axis (and therefore also its x-axis) has a different orientation. The z axis is defined as the projection of the local central ray on the x-z plane. The consequence is that x and z axis are a function of the fractional particle momentum  $\Delta p/p$ .

## 4.2 Calculation of Coordinates

In this section is discussed how the coordinates are calculated in the different coordinate systems. We will start in the detector system and will finish with the particle vertex in the hall A system. Along the way offsets are introduced to correct for misalignments. Of course, there are many ways in which parameters can be introduced to make these corrections, however, it is our goal to choose that set that fulfills the following two important requirements:

1. The parameter set that corrects for the misalignments should be as small as possible.
2. There should be a clear procedure how to determine each of them in a unique way.

After this section the experimental procedures will be discussed with which the parameters will be calibrated.

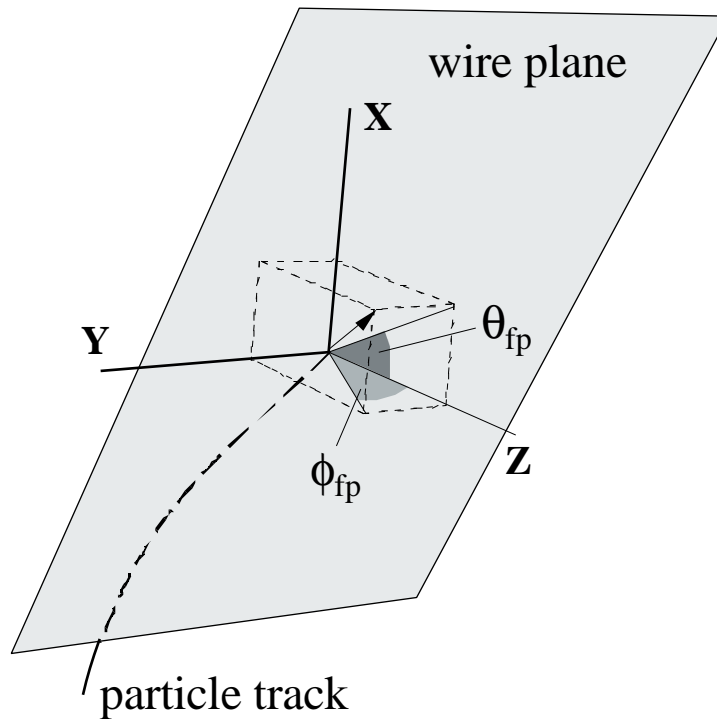


Figure 4.3: Spectrometer focal plane coordinate system.

### 4.2.1 Detector Vertex

A particle track  $p$  in the detector package of a magnetic spectrometer can be written in the detector coordinate system as

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} x_{ds} \\ y_{ds} \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} \sin \theta_{ds} \cos \phi_{ds} \\ \sin \phi_{ds} \\ \cos \theta_{ds} \cos \phi_{ds} \end{pmatrix}, \quad (4.1)$$

where  $\theta_{ds}$  and  $\phi_{ds}$  are geographic angles, see Sec. 4.1.1. The four unknowns ( $x_{ds}$ ,  $y_{ds}$ ,  $\theta_{ds}$  and  $\phi_{ds}$ ) are determined with four measurements in the drift-chamber package.

In Fig. 4.4 the wire direction in the first plane is shown making an angle of  $-\xi$  with the y-axis of the detector coordinate system. Indicated also is a particle track through the plane. The drift chamber can measure the projection of this track on a plane perpendicular to its wire plane. The projection of an arbitrary vector  $v = (v_x, v_y, v_z)$  is

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \xrightarrow{\text{projection}} \begin{pmatrix} v_x \cos^2 \xi - v_y \cos \xi \sin \xi \\ v_y \sin^2 \xi - v_x \cos \xi \sin \xi \\ v_z \end{pmatrix}. \quad (4.2)$$

The projection is characterized by two quantities: the angle  $\eta$  (see Fig. 4.4 and the position  $p_{vdc}$  where the projected track crosses the wire plane relative to the projection of

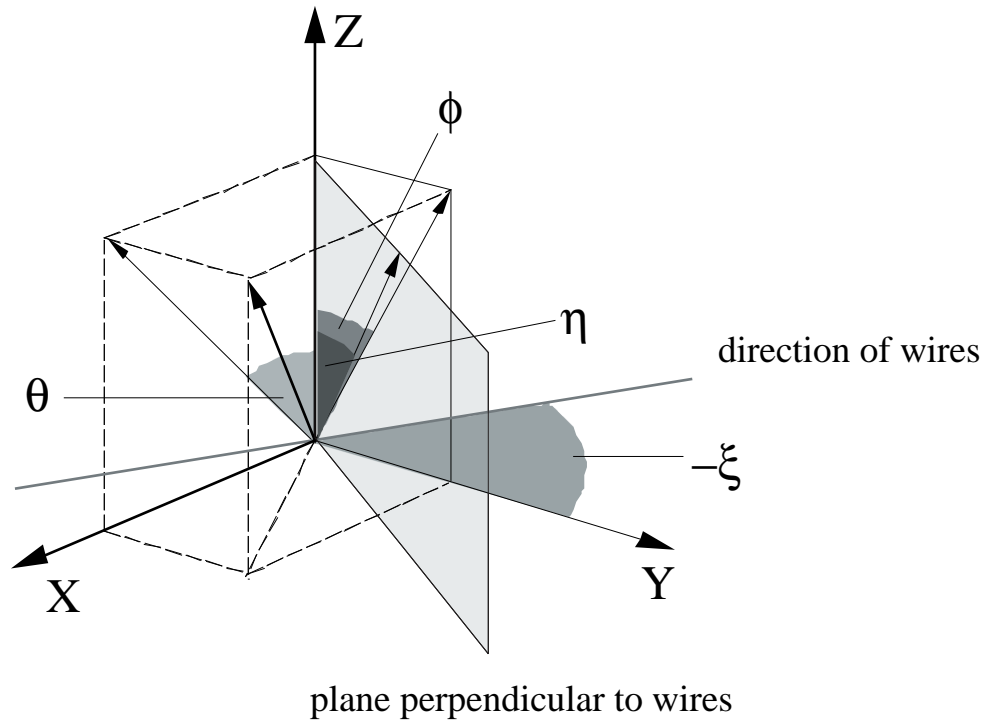


Figure 4.4: Spectrometer detector coordinate system. Two track angles  $\theta_{ds}$  and  $\phi_{ds}$  are indicated together with the angle  $\eta$  that is measured with one or two wire plane(s).



Figure 4.5: Configuration of the wire chamber package.

the origin of the detector coordinate system on this wire plane. Using Eq.(4.2), the expression for  $\eta$  and  $p_{\text{vdc}}$  in terms of the unknowns of Eq.(4.1) are:

$$\begin{aligned}\tan \eta_n &= \tan \phi_{\text{ds}} \sin \xi_n + \tan \phi_{\text{ds}} \cos \xi_n \\ p_{\text{vdc},n} &= p_x \cos \xi_n - p_y \sin \xi_n\end{aligned}\quad (4.3)$$

where the index  $n$  refers to the wire-plane number.

Equation 4.3 shows that two independent measurements of  $\eta$  will give  $\phi_{\text{ds}}$  and  $\theta_{\text{ds}}$ . This can be done with two drift chamber planes with different wire angles. However, in this configuration the  $\eta$ -resolution is limited to typically 8 mrad (FWHM). This can be improved by adding two more drift chamber planes, resulting in a configuration with two wire lanes with wire angle  $\xi_1$  and two with wire angle  $\xi_2$ . In Fig. 4.5 the spatial configuration of the wire planes is sketched.

Using only the position information  $p_{\text{vdc},n}$ ,  $n=1 \dots 4$ , we can calculate the vertex parameters in the SDCS as follows:

$$\begin{aligned}\tan \eta_1 &= \frac{p_{\text{vdc},3} - p_{\text{vdc},1}}{d_2} \\ \tan \eta_2 &= \frac{p_{\text{vdc},4} - p_{\text{vdc},1}}{d_2} \\ \tan \theta_{\text{ds}} &= \frac{\tan \eta_1 \sin \xi_2 + \tan \eta_2 \sin \xi_1}{\sin(\xi_2 - \xi_1)} \\ \tan \phi_{\text{ds}} &= \frac{\tan \eta_2 \cos \xi_1 + \tan \eta_1 \cos \xi_2}{\sin(\xi_2 - \xi_1)} \\ \tan x_{\text{ds}} &= \frac{p_{\text{vdc},1} \sin \xi_2 - (p_{\text{vdc},2} - d_1 \tan \eta_2) \sin \xi_1}{\sin(\xi_2 - \xi_1)} \\ \tan y_{\text{ds}} &= \frac{p_{\text{vdc},1} \cos \xi_2 - (p_{\text{vdc},2} - d_1 \tan \eta_2) \cos \xi_1}{\sin(\xi_2 - \xi_1)}\end{aligned}$$

Note that the coordinates of the point where the particle crosses the first plane are calculated with information mainly coming from the first two planes. This minimizes the effects on  $x_{\text{ds}}$  of multiple scattering in the drift chambers. A correction has been applied to

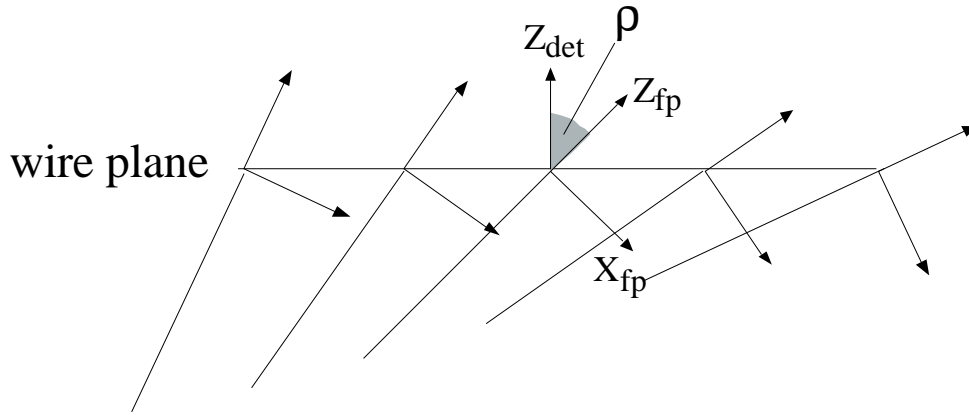


Figure 4.6: Spectrometer focal plane coordinate system which is rotated over an angle  $\rho$  along the  $y$ -axis wrt the SDCS.  $\rho$  is a function of the relative momentum.

the position  $p_{\text{vdc},2}$  for the fact that the first two planes are separated by a distance  $d_1$ . Some assumptions have been made in this derivation:

- Wire angles are known and constant.
- Wires are positioned in planes.
- Planes are parallel and at known distances.
- Location of the center of the wire planes wrt. the SDCS origin is known.

Deviations from the nominal values will lead to offsets in the SDCS coordinates. In order to correct for these deviations at least four offsets have to be introduced. In principle one could introduce offsets to correct for each of the items mentioned above, however, this would result in a set of highly correlated parameters. Therefore, we prefer to have one offset for each of the coordinates,  $x_{\text{ds}}$ ,  $\theta_{\text{ds}}$ ,  $x_{\text{ds}}$  and  $\phi_{\text{ds}}$ . The offsets will be introduced in this two sections describing the transformation of the vertex to the spectrometer focal plane coordinate system and the spectrometer reconstructed coordinate system (SRCS). Some of the offsets can be combined with parameters describing these transformations.

### 4.2.2 Focal-Plane Vertex

Before reconstructing the particle vertex at the spectrometer entrance the particle vertex in the detector system is transformed to the spectrometer focal-plane coordinate system (a name slightly misleading as one will see further on). This coordinate system is obtained by rotating SDCS along its  $y$ -axis over an angle  $\rho$ , where  $\rho$  is the angle between the projection of the local central ray ( $\theta_{\text{tg}} = \phi_{\text{tg}} = 0$ ), see Sec. 4.1.4 on the  $x$ - $z$  plane and the  $z$ -axis of the

SDCS<sup>1</sup>. So the new z-axis rotates as a function of the relative particle momentum  $\Delta p/p$ . The main advantage of expressing the vertex in this system is the fact that the dispersive angle  $\theta$  will be small making the expressions for the reconstructed vertex converge faster.

Writing down the expressions for the new coordinates, the following points should be noted:

1. Offset correction and rotation can be combined in one parameter.
2. Since the particle momentum is the variable with the highest resolution (and therefore most sensitive) we do not want to apply a rotation to this coordinate that is not constant. Then, a new calibration of the momentum dependence of  $\rho$  would immediately have consequences for the dispersion constants, see Eq.(6.4) on p. 57.

The offset for  $x_{ds}$  will be introduced when calculating the particle momentum and the  $\theta_{ds}$  offset will be absorbed in the transformation to the spectrometer focal-pane coordinate system. So up to here we have:

$$\begin{aligned}
 x_{fp} &= \cos \rho_0 (x_{ds} - x_{offset}) \\
 y_{fp} &= y_{ds} - y_{offset} \\
 \tan \theta_{fp} &= \frac{\tan \theta_{ds} - \tan \rho}{1 - \tan \theta_{ds} \tan \rho} \\
 \tan \phi_{fp} &= \frac{\tan \phi'_{ds}}{\cos \rho - \sin \rho \tan \theta_{ds}} \\
 \tan \phi'_{ds} &= \frac{\tan \phi_{ds} - \tan \phi_{offset}}{1 + \tan \phi_{ds} \tan \phi_{offset}}
 \end{aligned}$$

### 4.2.3 Hall A Laboratory Vertex

Suppose the normal to the sieve slit surface that goes through the central hole is given by the geographical angles  $\Theta_0$  and  $\Phi_0$  in the hall A laboratory coordinate system<sup>2</sup>. The geographical angles of the particle vertex in the HLCS is given by:

$$\begin{aligned}
 \theta_{HLCS} &= \theta_0 \pm \phi_{tg} \\
 \phi_{HLCS} &= \phi_0 + \theta_{tg}
 \end{aligned}$$

The question whether to add or subtract  $\phi_{tg}$  depends on the position of the spectrometer with respect of the beam line. In order to calculate the reaction point  $(x, y, z)$  additional

---

<sup>1</sup>Under ideal conditions, such as spectrometer mid-plane symmetry, the local central ray is in the x-z plane.

<sup>2</sup>I guess some of you will argue that to define the orientation of the spectrometer uniquely in space also the rotation around this normal should be specified. I guess you are right but usually spectrometer can not (or better formulated should not) perform this motion.

information is needed from another spectrometer, detecting a particle in coincidence, and/or the beam position from the rastering system.

### 4.3 Calibration of Coordinates

Up to now we have defined the different coordinate systems and discussed how the particle vertex as determined in the spectrometer detector package leads to the vertex of the particle emerging from the reaction point in the target. However, the usefulness of all this stands or falls with the existence of procedures to calibrate these coordinates.

In the following discussion it is assumed that the orientation of the sieve slit with respect to the HLCS is aligned carefully through optical procedures. At this point we assume also that the HLCS itself can be checked through a straightforward procedure. So starting with a well calibrated system of sieve slit and magnet positions wrt. the HLCS, elastic electron scattering measurements should be performed so that a sharp peak in momentum is detected. This measurement should be repeated after changing the magnetic field, thereby after several changes covering the whole momentum acceptance. The central hole of the sieve slit provides us with data to determine the offsets introduced in the SDCS for  $\theta$ ,  $y$  and  $\phi$ . The offset in  $x$  is absorbed in the dispersion coefficient  $d_0$  of Eq.(6.4) on p. 57. Determination of this coefficient needs different type of measurements as discussed in [5]



# Chapter 5

## Tracking

In this chapter we will discuss the track reconstruction with the spectrometer VDC package. This is the most important part in the data analysis with a high-resolution spectrometer, justifying a whole chapter about this subject. First, the program flow for tracking will be discussed, followed by an explanation of the minimum spanning tree algorithm, used in disentangling multiple tracks. We finish with showing the tracking efficiency of the algorithm by analyzing simulated  $^{16}\text{O}(e, e'p)$  data.

### 5.1 Program Flow for Tracking

### 5.2 Minimum Spanning Tree Algorithm

### 5.3 Definition of Wire Chamber Hit Type HPCH

Each wire chamber hit is assigned a hit type. At the moment the following seven different conditions are being distinguished:

Table 5.1: Hit types for a wire chamber hit.

HPCH value		Description
dec	bin	
2	00000010	a nohit or all clusters are of ERR1 type.
4	00000100	number of wires hit is $\geq 64$
8	00001000	some drift times resulting in drift distances larger than 1.25 times the distance between wire plane and cathode
16	00010000	some drift times are correcting for the stop offset negative

For an error condition HPCH has a value  $\geq 2$ . In this case the hit type can be a combination of several error conditions.

## 5.4 Tracking Efficiency

# Chapter 6

## Calibration

The resolution and position of several variables can be optimized. To start an optimization give the command

```
optimize/calibrate variable_string condition_string
```

The `condition_string` can be any valid logical statement. The variable combinations that can be optimized are given in Tab. 6.1.

Table 6.1: Variable combinations that can be optimized. `spec_s` can be either `spec_e` or `spec_h`, `snum` either `s1` or `s2`, and `side` can be either `l` (left) or `r`(right).

variable	comments
<code>spec_s.th_tg/spec_s.ph_tg</code>	Reconstructed $\theta$ and $\phi$ target coordinates. These angles can only be optimized for a measurement with a sieve collimator. Both of them are optimized at the same time, see Ref. [6].
<code>spec_s.y_tg</code>	Reconstructed $y_{tg}$ coordinate. The non-dispersive position can be optimized after a series of measurements have been performed in which the beam spot has been put at different positions.
<code>tc_cor</code>	Coincidence time between HRSE and HRSB after correction for path length differences wrt. the central rays and time offsets between the scintillators.
<code>e_miss</code>	Missing energy for $(e, e'x)$ .
<code>e_miss/p_miss</code>	Missing energy and momentum for $H(e, e'p)$ .
<code>spec_s.snum.adc_side_c</code>	Scintillator ADC values corrected for pedestals and attenuation.
<code>spec_s.beta</code>	particle $\beta$ as calculated by the time-of-flight between <code>s1</code> and <code>s2</code> .

There, `spec_s` can be either `spec_e` or `spec_h`, `snum` either `s1` or `s2`, and `side` can be either `l` (left) or `r` (right). Except for the optimization of `tc_cor`, an additional input file has to

be supplied. Since the selection of events used in the optimization procedure is usually too complicated to be accomplished through a series of cuts defined in `ESPACE`, this file contains information for the event selection. The file can be assigned through the command `set/file/fit input`, see Sec. D (pp. 81).

For the different optimization options this file has a different format. First we discuss the case that the vertex reconstruction of a spectrometer is optimized, so `dp_kin`, `th_tg/ph_tg` and `y_tg`. Then this file should contain in the first line the number of positions along the focal plane where one wants to optimize, followed by a line for each position containing the center of this region, its new position (only useful when `dp_kin` is being optimized) and the half-width of this region.

If one wants to optimize `th_tg/ph_tg` or `y_tg` additional data have to be supplied. In case of `th_tg/ph_tg` this should be the number of holes in the sieve collimator. It has to be followed by a line for each hole giving the center of the hole in `th_tg` before optimization, `th_tg` hole position after optimization and half the width of the ellipse in the  $\theta$  direction which will be put around the initial coordinates of the hole center. The same should be given for `ph_tg`. Only those events are used which have their initial values inside one of the ellipses. The beginning of a fit-file for optimizing `th_tg/ph_tg` with a 77-hole sieve-collimator would look like

```
77 (theta_ini, theta_final, dtheta, phi_ini, phi_final, dphi)
    -0.04500    -0.07024  0.0075 -0.08600    -0.09757  0.0050
```

The input for optimizing `y_tg` would consist out of first a line with the number of beam-spot positions which is being used in the optimization. It is followed by a line for each beam-spot position with the initial position, final position and half the width of the region around the initial position containing the events that are accepted for the optimization.

An important parameter in the optimization procedure is the variable that controls the number of events per peak per file that is being used (command `set/fit/npts`). With this variable one can prevent that the data used in the fit is coming from just the first few files being scanned.

If `e_miss` is optimized the file should contain in the first line the number of peaks in the missing energy spectrum that one wants to optimize. This is followed by a line for each peak containing the center of this region, its new position and the half width of this region. All in units of MeV.

## 6.1 ADC Gains and Pedestals

The first calibration that has to be done is the calibration of the pedestals and gains for the scintillator ADCs. This has to be done whenever any replacement of phototubes took place or at the beginning of an experiment.

Let us consider what determines the ADC value for a given event. An ionizing particle passes a scintillator paddle. The amount of light produced is proportional to the energy loss times the pathlength in the paddle. For good events, the pathlength is somewhat a constant,

because they intersect the paddle planes almost perpendicular. The light propagates through the scintillator to the PMTs. Because of non-100% transparence and reflection at the paddle surfaces the signal is attenuated. This is usually, though not exactly correct, described by a exponential. This attenuation constant is somehow empirical, one finds different coefficients for paddles made of the same material, but, e.g., different thicknesses. Furthermore, you get a signal loss coupling the scintillator paddle to the light guide and to the PMT window. The quality of the coupling may vary and may cause different losses for different paddles. The remaining photons knock electrons off a photo cathode which are multiplied by the PMT. This resulting signal is converted by an ADC into a digital signal and fed into the data stream. All these effects cause an per PMT individual reduction of the deposited signal, additionally to a statistical broadening. The goal of the optimization is to correct for the pedestals and all these losses to reconstruct the differential energy loss of the particle, e.g. for particle discrimination purposes. Currently, ESPACE corrects for pedestals and gain only and calls this new variable `adc_<side>_c`<sup>1</sup>. Calibrating pedestals and gain may be done separately using the `/Espace/SEt/Fit/Adc/Pedestal` (see description on p. 85) and `/Espace/SEt/Fit/Adc/Gain` (p. 85) commands. The attenuation is not corrected.

For normal data acquisition mode the pedestals are suppressed. Consequently to do scintillator ADC optimization, one has to take data with pedestal unsupressed. This can be done by setting the pedestal position at zero in a file that is read by CODA. It might be useful to point out that this is not what is usually called a pedestal run. The pedestal run purpose is to find the position of the actual pedestal to write it in the files that CODA uses for pedestal suppression. Data used for the purpose of fitting the pedestal and the gain can be either cosemics or beam data, the only condition is to cover the whole plane.

A typical phototube ADC spectra consists from a peak for the pedestal and a physical signal zone; the purpose of finding the pedestal position and set a gain is to apply a correction to the physical signal so that the maximum of the physical signal zone to correspond to same ADC value (compare Figs. 6.1 and 6.2). By representing `adc_<side>_c` we can check if the calibration/optimization was successful or not. The optimization can be done simultaneously for the left and right phototubes in electron or hadron arm.

Choosing an appropriate data set for optimization one has to distinguish two conditions:

pedestals: If one is interested in the pedestals only (which are important for the time walk correction) and does not care about particle identification, any data (without pedestal suppression) are good. If you have production run data only, most of the pedestals are suppressed, and you will find only the tail extending to higher energies plotting the ADC spectra. Identify the threshold. The pedestal position should be 10 channels below that threshold. Put this number into the database, by hand.

gain: Best suited for gain calibration are real data illuminating the whole focal/detector

---

<sup>1</sup>The ADC value is not used in ESPACE except in calculating the time walk correction for the timing. For that one should use the raw ADC value, subtracted by the pedestal. This is the “real” ADC value that was fed into the discriminator and ADC. Thus, in future the definition of `adc_<side>_c` should be changed to reflect `adc_<side>-pedestal`, and the full correction should be stored in a variable called `dEdx_<side>`.

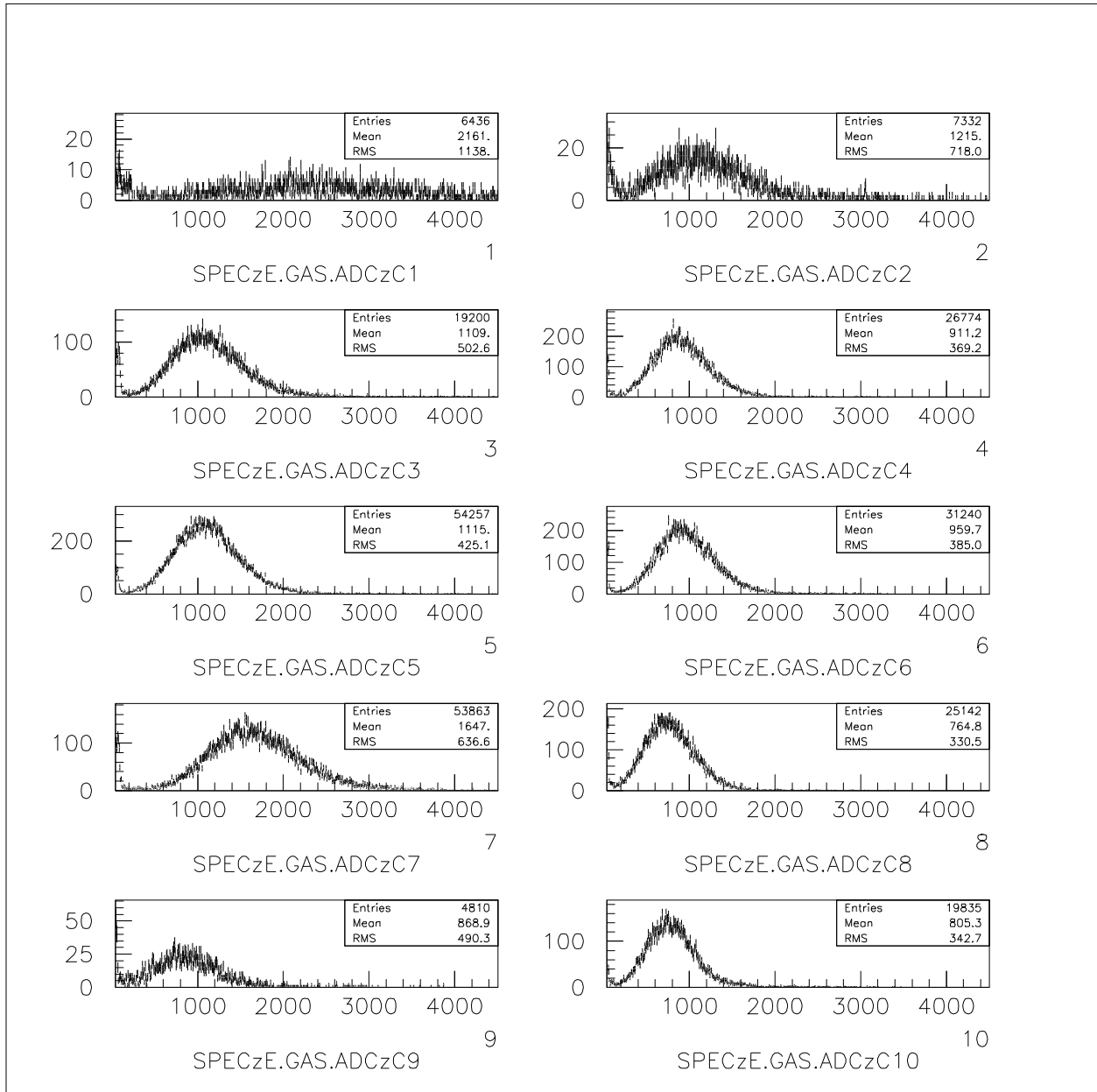


Figure 6.1: Čerenkov ADC signal before optimization.

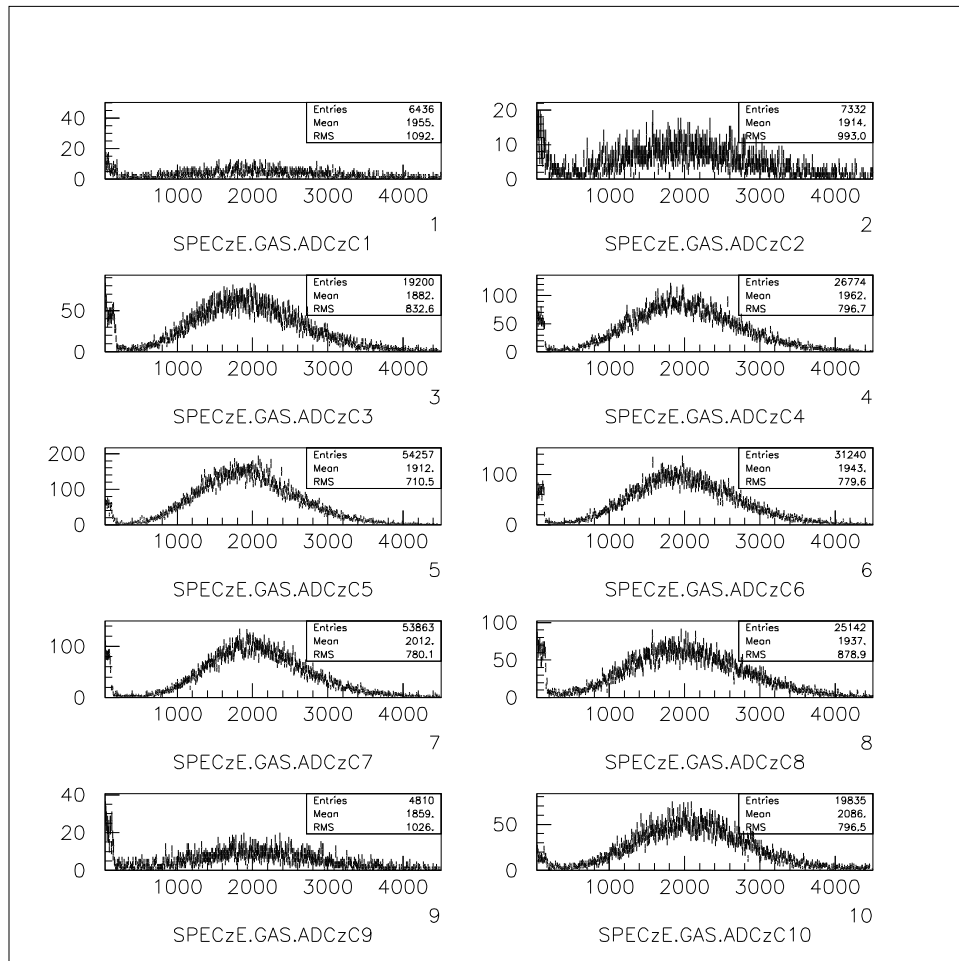


Figure 6.2: Čerenkov ADC signal after optimization.

plane. Good are singles, but not from elastic scattering, and quasi-elastic data. Elastic scattering data show a correlation between momentum and scattering angle, i.e. between paddle number and `ydetgeom` where the paddle is hit. Thus, the attenuation correction is folded into the gain, and the resulting data base may not be applicable to any other kinematical setting. Cosmics may be good to, especially for uniform illumination, but depending on which other conditions may be applied, the intersection with the paddle may not be perpendicular nor the mean uniform in paddle number. Thus, different individual path lengths may be fitted into the gain.

To clean up the events one should consider a few cuts:

- No multi hits in ADC, e.g. `define/cut/1d spec_e.s1.mult 0.5 1.5`
- coincidence of `s1` and `s2`, i.e.  
`define/logical one_sc_hit_e spec_e.s1.mult&&spec_e.s2.mult`
- adding the VDCs, cutting on one cluster `def/cut/1d spec_e.u1.clus 0.5 1.5`  
and good multiplicity `define/cut/1d spec_e.u1.mult 1.5 7.5`
- Using the VDC reconstructed track a cut on the intersection angle with the scintillator plane can be applied.

The more cuts you apply, the more reliable your calibration will be, but consider e.g. using cosmics you may need a lot of events getting a small sample of good events.

Thus, an ESPACE kumac file for ADC calibration could look like:

```
set/fit/adc/pedestal off
define/cut/1d spec_e.s1.mult      0.5 1.5
define/cut/1d spec_e.s2.mult      0.5 1.5
define/logical one_sc_hit_e spec_e.s1.mult&&spec_e.s2.mult

calibrate/optimize spec_e.s1.adc_l_c one_sc_hit_e
calibrate/optimize spec_e.s2.adc_l_c one_sc_hit_e
calibrate/optimize spec_e.s1.adc_r_c one_sc_hit_e
calibrate/optimize spec_e.s2.adc_r_c one_sc_hit_e
```

After the optimization, of the database the lines containing the offsets (pedestal positions) and amplification factors (gains) for the paddles in the respective HRS should have been altered.

## 6.2 Čerenkov ADC

## 6.3 Shower and Pre-Shower Calibration

Please see Ref. [7].

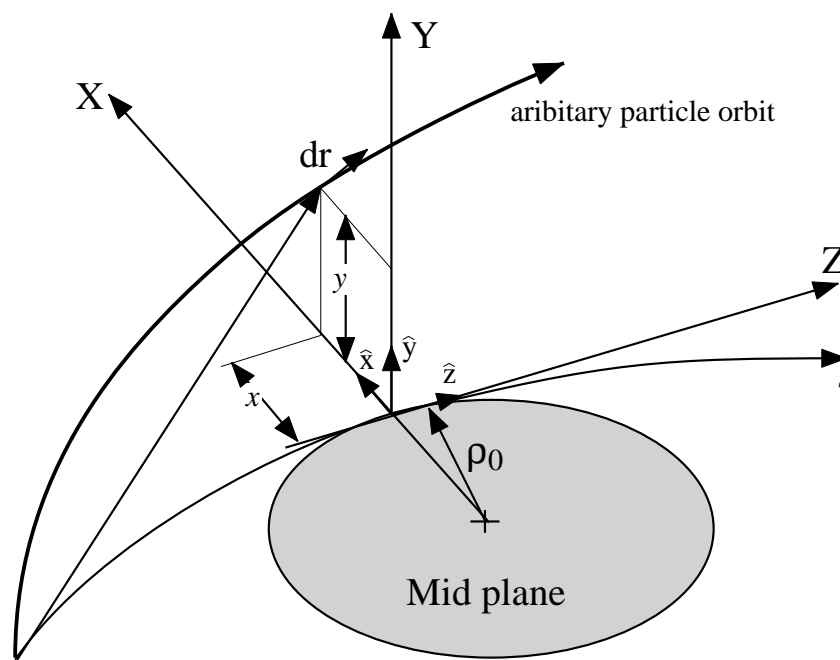


Figure 6.3: Spectrometer transport coordinate system.

## 6.4 Scintillator Timing

## 6.5 VDC Time Offsets

## 6.6 VDC Time to Distance

## 6.7 Vertex Reconstruction

Well, one could devote a whole chapter to this topic, actually a whole book. But let's not get carried away and just explain the basics to get you prepared to do your spectrometer calibration.

The basis for magnetic design and discussion is a six dimensional vector and the corresponding transfer tensor calculated by such programs as TRANSPORT [8]. The components of this vector include position and tangent of angle in the dispersive plane (usually designated  $x$  and  $\theta$ ), the nondispersive coordinates ( $y$  and  $\phi$ ), and the fractional deviation of the momentum from the central design momentum ( $\delta$ ). A sixth dimension describing ray path length variations is included, but it is ignored in our discussion for simplicity. Note that all tensor elements in this document with subscript '5' correspond to the TRANSPORT element with subscript '6'.

The positions  $x$  and  $y$  are defined in a right-handed rectangular coordinate system with  $\hat{z}$  along the particle direction and  $\hat{x}$  directed away from the center of curvature in the bending

plane. Thus, for upward bending spectrometer like HRS, at the target  $\hat{x}$  points downward. The angles  $\theta$  and  $\phi$  are measured from the  $z$ -axis to the projection of the particle vector onto the  $x$ - $z$  and  $y$ - $z$  planes, respectively. At any position through the spectrometer, these coordinates are measured with respect to the central ray (i.e.,  $\hat{z}$  points along the central ray), where the central ray is defined to be that ray with all initial coordinates equal to zero at the target.

The form of the first-order forward transfer tensor is shown in Eq.(6.1).  $\text{tg}$  designates a vector at the target, and  $\text{fp}$  designates a vector measured in the focal plane region.

$$\begin{pmatrix} x \\ \tan \theta \\ y \\ \tan \phi \\ \delta \end{pmatrix}_{\text{fp}} = \begin{pmatrix} \langle x|x \rangle & \langle x|\theta \rangle & 0 & 0 & \langle x|\delta \rangle \\ \langle \theta|x \rangle & \langle \theta|\theta \rangle & 0 & 0 & \langle \theta|\delta \rangle \\ 0 & 0 & \langle y|y \rangle & \langle y|\phi \rangle & 0 \\ 0 & 0 & \langle \phi|y \rangle & \langle \phi|\phi \rangle & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ \tan \theta \\ y \\ \tan \phi \\ \delta \end{pmatrix}_{\text{tg}} \quad (6.1)$$

Notice that symmetry in the  $x$ - $z$  plane is enforced by making several tensor elements zero. We can easily find the inverse matrix which is shown in Eq.(6.2). Again, those elements which are explicitly zero result from midplane symmetry. In the notation below,  $\langle a|b \rangle^{-1}$  means  $(\langle a|b \rangle^{-1})_{ij}$  and not  $(\langle a|b \rangle_{ij})^{-1}$ .

$$\begin{pmatrix} x \\ \tan \theta \\ y \\ \tan \phi \\ \delta \end{pmatrix}_{\text{tg}} = \begin{pmatrix} \langle x|x \rangle^{-1} & \langle x|\theta \rangle^{-1} & 0 & 0 & \langle x|\delta \rangle^{-1} \\ \langle \theta|x \rangle^{-1} & \langle \theta|\theta \rangle^{-1} & 0 & 0 & \langle \theta|\delta \rangle^{-1} \\ 0 & 0 & \langle y|y \rangle^{-1} & \langle y|\phi \rangle^{-1} & 0 \\ 0 & 0 & \langle \phi|y \rangle^{-1} & \langle \phi|\phi \rangle^{-1} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ \tan \theta \\ y \\ \tan \phi \\ \delta \end{pmatrix}_{\text{fp}} \quad (6.2)$$

The representation above is not adequate, however, to describe the reconstruction of target variables from measured ones. The full five-dimensional vector is not detected by the focal plane instrumentation, which measures only positions and angles; the momentum information  $\delta$ , is absent. Thus, only two ( $\delta$  and  $y_{\text{tg}}$ ) of the three dispersive coordinates can be reconstructed. The third dispersive target coordinate must in principle be fixed to allow a solution for the problem. The logical choice is to constrain, of course,  $x_{\text{tg}}$ .

A matrix representation appropriate to the reconstruction procedure is trivially related to the above matrix by using

$$\delta_{\text{fp}} = \frac{x_{\text{tg}} - \langle x|x \rangle^{-1} x_{\text{fp}} - \langle x|\theta \rangle^{-1} \tan \theta}{\langle x|\delta \rangle^{-1}}$$

which results in the relation

$$\begin{pmatrix} x \\ \tan \theta \\ y \\ \tan \phi \\ \delta \end{pmatrix}_{\text{tg}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ \langle \theta|x \rangle^{-1} - \frac{\langle \theta|\delta \rangle^{-1}}{\langle x|\delta \rangle^{-1}} \langle x|x \rangle^{-1} & \langle \theta|\theta \rangle^{-1} - \frac{\langle \theta|\delta \rangle^{-1}}{\langle x|\delta \rangle^{-1}} \langle x|\theta \rangle^{-1} & 0 & 0 & \frac{\langle \theta|\delta \rangle^{-1}}{\langle x|\delta \rangle^{-1}} \\ 0 & 0 & \langle y|y \rangle^{-1} & \langle y|\phi \rangle^{-1} & 0 \\ 0 & 0 & \langle \phi|y \rangle^{-1} & \langle \phi|\phi \rangle^{-1} & 0 \\ -\frac{\langle x|x \rangle^{-1}}{\langle x|\delta \rangle^{-1}} & -\frac{\langle x|\theta \rangle^{-1}}{\langle x|\delta \rangle^{-1}} & 0 & 0 & \frac{1}{\langle x|\delta \rangle^{-1}} \end{pmatrix} \begin{pmatrix} x_{\text{fp}} \\ \tan \theta_{\text{fp}} \\ y_{\text{fp}} \\ \tan \phi_{\text{fp}} \\ x_{\text{tg}} \end{pmatrix} \quad (6.3)$$

This will hereafter be referred to as the reverse tensor (as opposed to the inverse tensor above). The dependence of  $\delta$  and  $\theta_{tg}$  on  $x_{tg}$  is explicit.

In practice, ESPACE proceeds a bit different. Only the the lower left  $4 \times 4$  part of Eq.(6.3) can be calibrated for a known  $x_{tg}$  position (hopefully zero). In the analysis the  $x_{tg}$  position is determined with the information of the beam position on target and the reconstructed  $\theta$ ,  $y$  and  $\phi$  position in the target of one or two spectrometers. This value of  $x_{tg}$  is used together with Eq.(6.3) to correct the focal plane coordinates to values corresponding to  $x_{tg} = 0^2$ . Realize that one has to obtain part of the tensor coefficients in Eq.(6.3) from a program like RAYTRACE [9] or COSY [10]. The new focal-plane vertex is used in the calculation of the reconstructed target vertex.

The transformation from the focal plane to the spectrometer entrance is described with a set of tensors,  $D_{ijkl}$ ,  $T_{ijkl}$ ,  $Y_{ijkl}$  and  $P_{ijkl}$ :

$$\begin{aligned} \frac{\Delta p}{p} &= \sum_{ijkl} D_{ijkl} x_{fp}^i \tan^j \theta_{fp} y_{fp}^k \tan^l \phi_{fp} \\ \tan \theta_{tg} &= \sum_{ijkl} D_{ijkl} x_{fp}^i \tan^j \theta_{fp} y_{fp}^k \tan^l \phi_{fp} \\ y_{tg} &= \sum_{ijkl} D_{ijkl} x_{fp}^i \tan^j \theta_{fp} y_{fp}^k \tan^l \phi_{fp} \\ \tan \phi_{tg} &= \sum_{ijkl} D_{ijkl} x_{fp}^i \tan^j \theta_{fp} y_{fp}^k \tan^l \phi_{fp} \end{aligned}$$

If the spectrometer has a symmetry plane that coincides with the  $x$ - $z$  plane of the SRCS, the following restrictions should be imposed on the tensor coefficients: only combinations of powers of  $y_{fp}$  and  $\tan \phi_{fp}$  with even values for the power sum  $k + l$  occur in the calculation of  $\Delta p/p$  and  $\tan \theta_{tg}$ , while only odd values for  $k + l$  are in the calculation of  $y_{tg}$  and  $\tan \phi_{tg}$ .

It is important to notice that  $\Delta p/p$  for particles entering the spectrometer with  $\theta_{tg} = \phi_{tg} = 0$  only depends on  $x_{fp}$ . The expression for the absolute momentum of these particles is given by:

$$p_f = p_{\text{centr}} \left( 1 + \sum_{j=1}^n d_j x_{fp}^j \right) \quad , \quad (6.4)$$

where  $p_{\text{centr}}$  is the central momentum setting of the spectrometer and  $d_j$  the spectrometer dispersion coefficients. Under ideal circumstances its dependence on the spectrometer magnetic field is linear. However, saturation effects will introduce non-linear terms:

$$p_{\text{centr}} = \sum_{j=0}^m \Gamma_j B^j$$

This simple relation is due to the choice of our focal-plane coordinate system. In the next section where the calibration of the different coordinates is discussed we will see that as a result of this choice a special calibration procedure will give information about the important spectrometer dispersion coefficients.

---

<sup>2</sup>So, this  $x_{tg}$  correction is only correct upto first order.

## Spherical aberrations

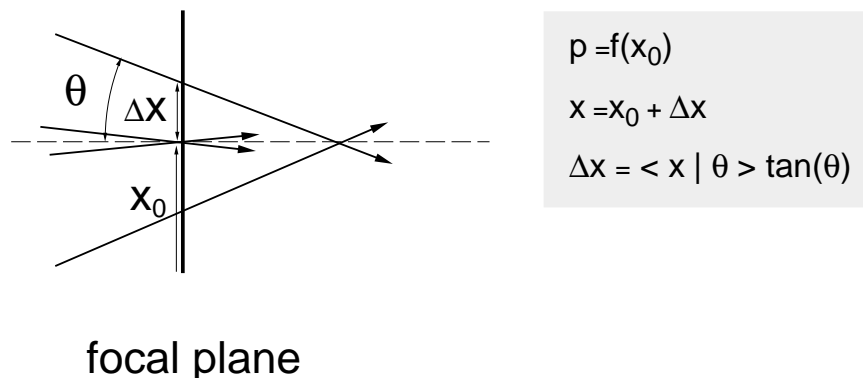


Figure 6.4: Explanation of spherical momentum aberrations in the focal plane of a magnetic spectrometer. The dashed line is made by a ray with momentum  $p$  and  $\theta_{tg} = 0$ .

### 6.7.1 Non-Dispersive Position $y_{tg}$

### 6.7.2 Angles

### 6.7.3 Momentum Aberrations

To give you some idea how a scanning session with optimization will look like, an example is discussed in which the transfer tensor for calculation of momentum aberrations is being optimized. In Fig. 6.4 the dispersive projection of the focal plane is drawn. Imagine that the spectrometer momentum setting is such that we observe the elastic peak of  $^{12}\text{C}(e, e)$ . The ray with momentum  $p$  and  $\theta_{tg} = 0$  is indicated by a dashed line. The position where it crosses the first wire plane is indicated by  $x = 0$ . Rays with the same momentum and an infinitesimal small  $\theta_{tg}$  cross this wire plane at the same spot and a relation can be established between momentum and  $x$ -position in that wire plane. However, rays entering the spectrometer collimator with the same momentum but a larger  $\theta_{tg}$  will typically cross the wire plane at a different spot. In other words, the particle momentum can not be deduced solely from the dispersive position  $x$ , unfortunately all the focal plane vertex parameters ( $\theta_{fp}$ ,  $y_{fp}$ ,  $\phi_{fp}$ ) are involved. It is now our task to describe this deviation  $\Delta x$  as a function of the focal plane vertex, the so-called transfer tensor for the momentum aberrations. Most spectrometer designs are such that the dependence on  $\theta_{fp}$  is the most important one.

Below an example of a KUMAC-file is given in which the transfer tensor for the momentum aberrations is being optimized. Seven measurements of an elastic electron scattering peak have been made at different values of  $\Delta p/p$ . After the assignment of the different input files and definitions of cut's, histograms and scatter plots are defined. Especially the  $\text{th\_tra}/dp_{kin}$  ( $\theta_{fp}/dp_{kin}$ ) plots around the seven elastic peak positions are of interest since they will reveal the momentum aberration as a function of  $\theta_{fp}$ . This is followed by

a command indicating that at the end of the event-by-event scan an optimization of the momentum transfer tensor has to be performed.

```

define/batch on
set/file/output c12_full conv1.hbook
set/file/database db_hrs_7x7
set/file/opt database db_hrs_7x7_opt
set/file/detmap detmap.config
set/file/fit_input fit_full_conv1.inp

***** Define cuts *****

define/cut/1d spec_e.y_tra-1 -0.144 0.144
define/cut/1d spec_e.x_tra-1 -0.749 0.749
define/cut/1d spec_e.th_tg-1 -0.07 0.07
define/cut/1d spec_e.ph_tg-1 -0.030 0.030
define/logical vdc_hole spec_e.y_tra-1&&spec_e.x_tra-1
define/logical accept_e spec_e.th_tg-1&&spec_e.ph_tg-1

***** spectrometer HRS *****

set/hitbits (+spec_e.s1)
set/auto_window off
set/spectrum/bins/x1 560
set/spectrum/window/x1 -0.07 0.07
spectra/save spec_e.th_tg
set/spectrum/bins/x1 320
set/spectrum/window/x1 -0.04 0.04
spectra/save spec_e.ph_tg
set/spectrum/bins/x1 500
set/spectrum/window/x1 -0.065 0.065
spectra/save spec_e.dp-1
spectra/save spec_e.dp_kin-1
set/spectrum/window/x1 -0.7 0.7
spectra/save spec_e.x_tra-1
set/ntuple 5000

**** dp-aberration plots

set/spectrum/window/x2 -0.5 0.5
set/spectrum/window/x1 -0.06 -0.04
spectra/save spec_e.th_tra/spec_e.dp_kin-1
set/spectrum/window/x1 -0.035 -0.0275

```

```

spectra/save spec_e.th_tra/spec_e.dp_kin-2 vdc_hole
set/spectrum/window/x1 -0.025 -0.0175
spectra/save spec_e.th_tra/spec_e.dp_kin-3 vdc_hole
set/spectrum/window/x1 -0.015 0.015
spectra/save spec_e.th_tra/spec_e.dp_kin-4 vdc_hole
set/spectrum/window/x1 0.015 0.0225
spectra/save spec_e.th_tra/spec_e.dp_kin-5 vdc_hole
set/spectrum/window/x1 0.0225 0.030
spectra/save spec_e.th_tra/spec_e.dp_kin-6 vdc_hole
set/spectrum/window/x1 0.040 0.060
spectra/save spec_e.th_tra/spec_e.dp_kin-7 vdc_hole

***** optimize momentum resolution in electron spectrometer

calibrate/optimize spec_e.dp_kin

***** scanning and output *****

set/file/header hdr_c12_dpm5.prm
file/scan ../monte_car/data_full_conv1_dpm5 LAST=1000 FIRST=1
set/file/header hdr_c12_dpm4.prm
file/scan ../monte_car/data_full_conv1_dpm4 LAST=1000 FIRST=1
set/file/header hdr_c12_dpm3.prm
file/scan ../monte_car/data_full_conv1_dpm3 LAST=1000 FIRST=1
set/file/header hdr_c12.prm
file/scan ../monte_car/data_full_conv1 LAST=1000 FIRST=1
set/file/header hdr_c12 dpp3.prm
file/scan ../monte_car/data_full_conv1_dpp3 LAST=1000 FIRST=1
set/file/header hdr_c12 dpp4.prm
file/scan ../monte_car/data_full_conv1_dpp4 LAST=1000 FIRST=1
set/file/header hdr_c12 dpp5.prm
file/scan ../monte_car/data_full_conv1_dpp5 LAST=1000 FIRST=1 CHOPT=-o

spectra/write
quit

```

Which parts of the tensor should be optimized is indicated in the database, see App. B. The first number in each line of the tensor elements indicates whether it is kept fixed or is being fitted. A summary of the fitting is listed in a file which name is deduced from the command `fit_input`, `fi.fit_full_conv1.inp` produces `fit_full_conv1.res`.

In Fig. 6.5 the result of such an optimization is shown. On the left side the scatter plot of `th_rot/dp_kin` is shown for elastic  $^{12}\text{C}(e, e)$  scattering without any aberration corrections. On the right is shown the same data but now after correction for momentum aberrations.

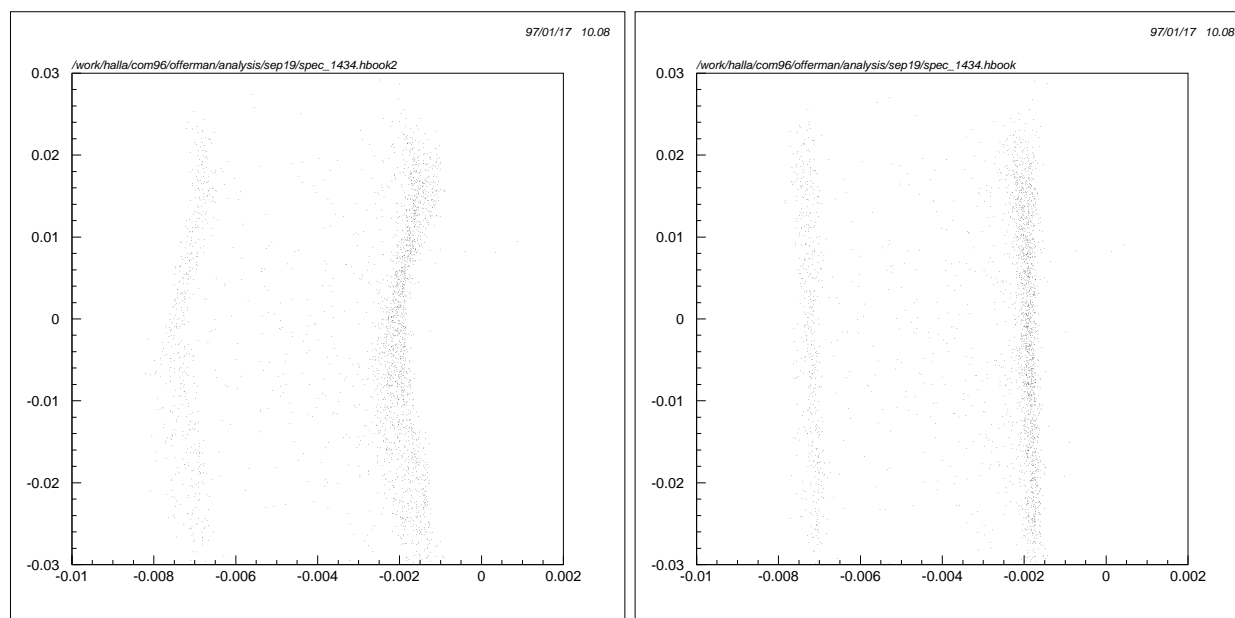


Figure 6.5: Two scatter plots showing  $\theta_{\text{rot}}$  versus  $dp/p$  for elastic  $^{12}\text{C}(e, e)$  scattering. In the one on the left side, no aberration corrections have been performed on the calculated momentum. The plot on the right is the result of an optimization like the example shown in this section.

## 6.8 Coincidence Time

## 6.9 Missing Energy

## 6.10 Adding a new Optimization

## 6.11 Beam Position Determination

This section will describe two aspects. The first is the calibration of the beam position monitors (BPM) versus the harps, the second describes the correction of the “phase slip” of the BPM readout. Major parts of this section were taken from ref. [11].

### 6.11.1 The Beam Position Monitors

A sketch of the beam position monitor and of the read-out electronics is shown in Fig. 6.6.

Each BPM has four antennas which are labeled  $X_p$  and  $X_m$ , and  $Y_p$  and  $Y_m$ . They pick up the signal of the beam, which is converted by ADCs. In December 1999, Struck 7510 ADC’s were installed. Since, the beam position is not only read out once per trigger but six times, separated by  $4\mu\text{s}$ . The importance of this will become clear in Sec. 6.11.4.

The beam position is computed in the routine `espace_halla/interpret_bpm_rast.f`. Depending on the parameter `beam_vertex` in the header file (see App. A), either the BPM or raster information is used, or no correction is done at all.

Here will be described only how the calculation of a single beam position for a BPM is done. Table 2.4 on p. 16 lists the new additional ESPACE variables. The beam position in the rotated coordinate system of the BPM is given by:

$$x_{\text{rot}} = \kappa \frac{X_{\text{p,cor}} - \alpha_X X_{\text{m,cor}}}{X_{\text{p,cor}} + \alpha_X X_{\text{m,cor}}} \quad (6.5)$$

The equation for  $y_{\text{rot}}$  is computed in a similar way. The ADC values are corrected for pedestals, e.g.  $X_{\text{p,cor}}$  is given by

$$X_{\text{p,cor}} = X_{\text{p}} - X_{\text{p,ped}} \quad (6.6)$$

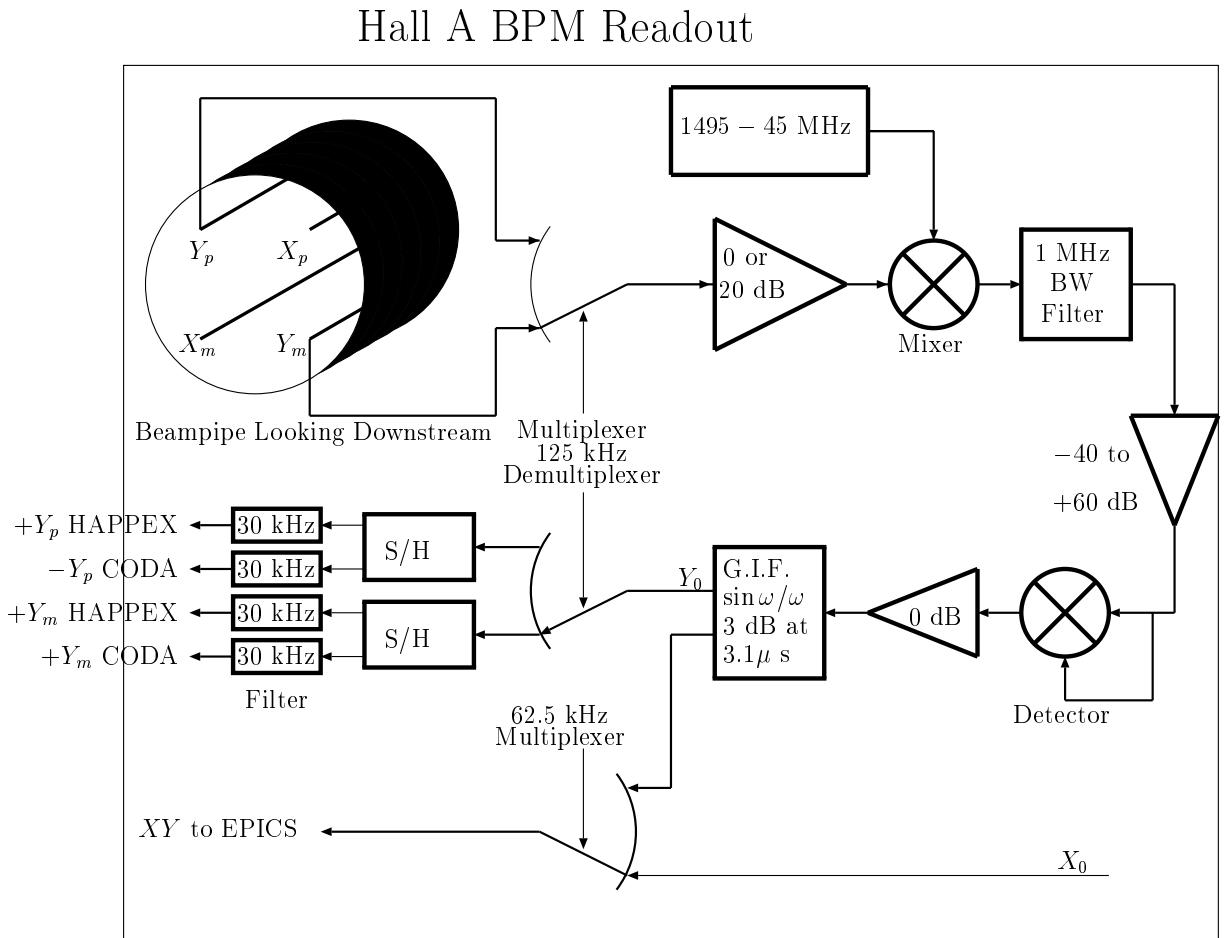


Figure 6.6: Sketch of the Hall A beam position monitors. The figure was taken from Ref. [11].

$\alpha_X$  corrects for different gains in the antennas, and  $\kappa$  converts the ADC value into a length, here meters.

The conversion from the rotated, left handed into the right handed Hall A coordinate system is done via

$$x_{\text{raw}} = x_{\text{pos}} + \frac{\kappa_X}{\sqrt{2}}(y_{\text{rot}} - x_{\text{rot}}) \quad (6.7)$$

$$y_{\text{raw}} = y_{\text{pos}} + \frac{\kappa_Y}{\sqrt{2}}(y_{\text{rot}} + x_{\text{rot}}) \quad (6.8)$$

$$(6.9)$$

All the calibration coefficients have to be specified in the data base, see App. B.

### 6.11.2 Beam Position Monitor Calibration

The beam position monitors can be calibrated by a so-called “bull-eye” scan. For un rastered beam, the beam position is measured using the harps, and at the same time a coda run is done. Typically, this is done for various horizontal and vertical displacements of the beam. A favorite scheme, e.g., is to displace the beam by  $\pm 2$  mm horizontally and vertically, but also other schemes can be considered. Important is, that both horizontally and vertically an as wide as possible range is covered, at least to the extents of the running experiment. The job is now to modify  $\alpha_X$ ,  $\alpha_Y$ ,  $\kappa_X$ , and  $\kappa_Y$  so that the beam position calculated by ESPACE agrees with the harps analysis. For the other parameters,  $\kappa$  is canonical and should not be changed (it is redundant anyway, and could be multiplied into  $\kappa_X$  and  $\kappa_Y$ ), the pedestals have to be obtained from a special pedestal calibration run, and the positions are taken from a survey.

An optimization routine does not exist, it is more like trial and error. However, it is not necessary to rerun ESPACE over and over again, instead an external script can be used, like the one for the analysis of the 3 Feb 2000 bulleye scan:

```
#!/bin/sh
#
cat antenna.dat | gawk '
BEGIN {
  getline
  AcalibRotXm = 1
  AcalibRotYm = 1
  BcalibRotXm = 1
  BcalibRotYm = 1
  calibRot    = 18.87
  calibX      = 1.1714
  calibY      = 1.2921
  AcalibX     = 1.12
```

```

AcalibY      = 1.12
BcalibX      = 1.02
BcalibY      = 1.02
AposX        = -0.9
AposY        = +0.35
BposX        = -0.15
BposY        = -0.3
print "coda  bpma x      y      bpmb x      y"
}
{
  Arotx = calibRot * ( $2 - AcalibRotXm * $3 ) / ( $2 + AcalibRotXm * $3 )
  Aroty = calibRot * ( $4 - AcalibRotYm * $5 ) / ( $4 + AcalibRotYm * $5 )
  Brotx = calibRot * ( $6 - BcalibRotXm * $7 ) / ( $6 + BcalibRotXm * $7 )
  Broty = calibRot * ( $8 - BcalibRotYm * $9 ) / ( $8 + BcalibRotYm * $9 )
  Ax = AposX + AcalibX / sqrt(2) * ( -Arotx + Aroty )
  Ay = AposY + AcalibY / sqrt(2) * (  Arotx + Aroty )
  Bx = BposX + BcalibX / sqrt(2) * ( -Brotx + Broty )
  By = BposY + BcalibY / sqrt(2) * (  Brotx + Broty )
  printf "%4d  %+6.3f ; %+6.3f  %+6.3f ; %+6.3f\n", $1, Ax, Ay, Bx, By
}'

```

This script reads the centroids of the separate antenna readings from `antenna.dat`, which needs to be created only one time:

```

coda  ax+  ax-  ay+  ay-  bx+  bx-  by+  by-
1741  1188 1160 1177 1170  1184 1102 1191 1105
1744  1187 1132 1177 1145  1183 1076 1191 1080
1745  1188 1133 1178 1145  1184 1077 1192 1080
1746  1071 1183 1178 1140  1160 1152 1192 1147
1747  1061 1182 1178 1128  1002 1151 1192  979
1748  1065 1182 1177 1142   997 1151 1192  983
1749  1075 1182 1177 1132  1004 1151 1192  983
1750  1115 1183 1165 1179  1184 1026 1035 1185
1751  1187 1103 1177 1030  1183 1011 1191  992
1752   962 1182 1040 1178   992 1152 1049 1184

```

Modifying the calibration parameters in the script, after some trying, `ESPACE` and `harp` analyses should agree. It should be mentioned that also  $\alpha_X$ ,  $\alpha_Y$ ,  $\kappa_X$ , and  $\kappa_Y$  have physical meanings and should be obtained from a hardware calibration, in principle (check Ref. [11]). However, in reality one finds that a better description can be achieved with slightly altered numbers. Also, it may be necessary to modify the device's position to avoid too unreasonable  $\alpha$ s. This can be seen in analogy to the transfer matrix, which in principle, also can be obtained from a magnet optics simulation.

### 6.11.3 Phase Shift

Plotting the beam position obtained from the raster (or just the ADC value) versus the determined beam position (raw position), one would expect a linear correlation (having the correct calibrations, a 45° line). E.g., for maximum raster current maximum elongation in the BPMs should be expected. Instead, one obtains something elliptical, which indicates that both values are not read at the same time, but that the time difference is constant. Thinking in terms of the raster frequency, one can also speak of a phase shift. The general assumption is that the raster is read at the trigger, but that the BPM readings are delayed. This can be verified for the horizontal raster plotting raster or BPM versus `y_tg`, for pointlike targets. Indeed, within resolution, raster and `y_tg` show a linear correlation (take care of the signs), while `bpmx` and `y_tg` show an ellipse. For the vertical, in principle, several checks are possible, however they are not very sensitive. For vertical beam displacement, the vertical angular acceptance varies, and one also should obtain a correlation with `E_miss` (having the vertical correction `reconstruct r 1 2` switched off, see App. A). One can also here use a trial and error method to extract the phase shifts, however Ref. [11] describes a method to obtain them by fitting.

### 6.11.4 Struck 7510 “Burst Mode” ADC

A complicated, awkward and error prone procedure was used to correct the read beam position to the real beam position, using the determined phase shifts, the raster ADC values and their corresponding derivatives. The calibration constants had to be determined frequently and had to be stored in the notorious `rastconsts.dat` file. It is not the scope of this section to describe this method, this was before the Struck 7510 ADC.

The Struck 7510 ADC allows to read a given number of values for each event. This enables a burst mode for the raster and BPM readings, i.e. it was chosen to read the ADC values six times per trigger, separated by 4  $\mu$ s. The six readings allow to precisely track the beam motion due to raster for each event trigger, and correct the beam position for the phase shift. Essentially, instead to determine coefficients for a sample of events and successively read them from `rastconsts.dat`, they can be obtained for each single event.

The task cooks down to fit a trigonometric function into a given number of data points  $a_i$ , here 6, which are read at times  $t_i$ . The function is given by

$$a(t) = a_0 + A \cos(\omega t + \phi) \quad (6.10)$$

$\omega$  is  $2\pi\nu$ ,  $\nu$  the raster frequency. We are searching for  $a_0$ ,  $A$  and  $\phi$ , (offset of the beam, beam motion amplitude due to the raster, phase of the raster motion), which minimize

$$\sum_{i=1}^6 (a_i - a(t_i)) \quad (6.11)$$

This minimization has to be done eight times per event (eight antennas). It is clear that the (Eq.6.10) is not linear, and that a MINUIT optimization would take too much time.

Factorizing the cosine gives

$$a(t) = a_0 + A \cos \omega t \cos \phi - A \sin \omega t \sin \phi \quad (6.12)$$

or

$$a(t) = \sum_{j=1}^3 \alpha_j(t) z_j \quad (6.13)$$

which follows from (Eq.6.12) using the following substitutions:

$$\begin{array}{c|cc} j & \alpha_j & z_j \\ \hline 1 & 1 & a_0 \\ 2 & \cos \omega t & A \cos \phi \\ 3 & \sin \omega t & -A \sin \phi \end{array} \quad (6.14)$$

(Eq.6.13) is linear, and we can perform a  $\chi^2$  minimization ( $\alpha^i$  is short for  $\alpha(t_i)$ )

$$\chi^2 = \sum_i (a_i - a(t_i))^2 = \sum_i (a_i - \sum_j (\alpha_j^i z_j))^2 = \min \quad (6.15)$$

i.e. compute the derivatives

$$\frac{\partial}{\partial z_k} \chi^2 = 0 = \sum_i (a_i - \sum_j (\alpha_j^i z_j) \alpha_k^i) \quad (6.16)$$

for  $k = 1 \dots 3$ . This system of three equations can be displayed as a matrix equation

$$\begin{pmatrix} \sum_i (\alpha_1^i)^2 & \sum_i \alpha_2^i \alpha_1^i & \sum_i \alpha_3^i \alpha_1^i \\ \sum_i \alpha_1^i \alpha_2^i & \sum_i (\alpha_2^i)^2 & \sum_i \alpha_3^i \alpha_2^i \\ \sum_i \alpha_1^i \alpha_3^i & \sum_i \alpha_2^i \alpha_3^i & \sum_i (\alpha_3^i)^2 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \sum_i a^i \alpha_1^i \\ \sum_i a^i \alpha_2^i \\ \sum_i a^i \alpha_3^i \end{pmatrix} \quad (6.17)$$

Substituting the  $\alpha_j$  of (Eq.6.14) follows:

$$\begin{pmatrix} 6 & \sum_i \cos \omega t_i & \sum_i \sin \omega t_i \\ \sum_i \cos \omega t_i & \sum_i \cos^2 \omega t_i & \frac{1}{2} \sum_i \sin 2\omega t_i \\ \sum_i \sin \omega t_i & \frac{1}{2} \sum_i \sin 2\omega t_i & \sum_i \sin^2 \omega t_i \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \sum_i a^i \\ \sum_i a^i \cos \omega t_i \\ \sum_i a^i \sin \omega t_i \end{pmatrix} \quad (6.18)$$

The solutions for  $z_j$  are given by

$$z_j = \frac{\det(M_j)}{\det(M)} \quad (6.19)$$

$M$  here stands for the  $3 \times 3$  matrix in (Eq.6.18),  $M_j$  is constructed by replacing the  $j^{\text{th}}$  column of  $M$  by the vector on the right hand of (Eq.6.18). The rest is trivial, one gets:

$$a_0 = z_1 \quad (6.20)$$

$$A = \sqrt{z_2^2 + z_3^2} \quad (6.21)$$

$$\phi = \text{ATAN2}(-z_3, z_2) \quad (6.22)$$

and the beam position

$$a(t_{\text{trigger}}) = a_0 + A \cos(\phi + \phi_{\text{delay}}) \quad (6.23)$$

at the assumed trigger time follows.

Figure 6.7 shows an example of a fit to the burst mode BPM readings. A good fit should show no correlations in pos%phs and amp%phs. In this case, there is a correlation for x and a smaller one for y. This is a hint that the “timing” is not correct. What is this? Because also fitting the frequency of the cosine would make the system non-linear, the raster frequency is an input to the fit. Specifically, the product of raster frequency and  $\Delta t$  between the consecutive readings, goes into the fit. A 10% higher raster frequency has the same effect as a 10% shorter  $\Delta t$ . The solution here is empirically. The field for the raster frequency in the data base can be changed (see App. B). If the “correct” one is chosen, there should be no remaining correlation.

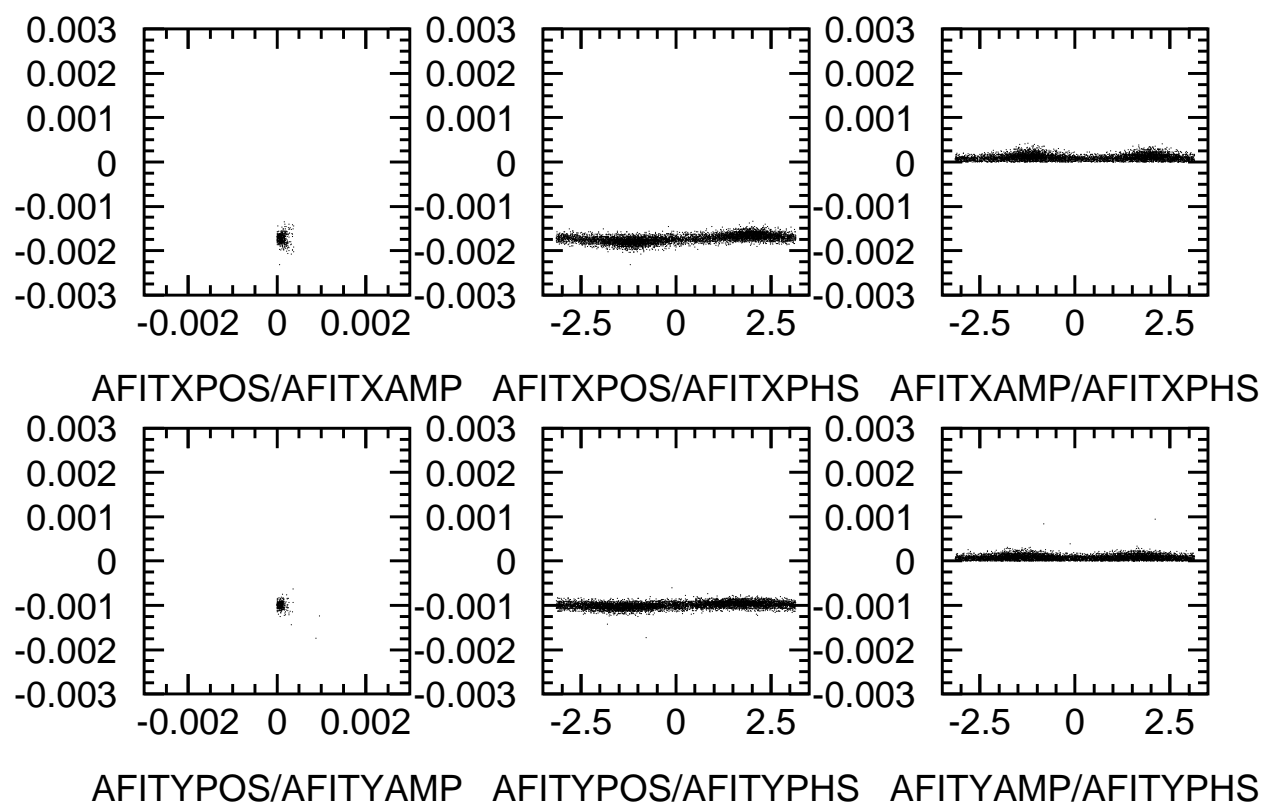


Figure 6.7: Example result for the fit to the burst mode BPM readings. A good fit should show no correlations in pos%phs and amp%phs. See further explanation in text.



# Appendix A

## Header File Description

The header file contains information which is needed during the scan to construct some of the variables. At the end of the scan this information is saved with each histogram. This way it is available to the next programs in the analysis chain. An example of a header file for analyzing  $^{12}\text{C}(e, e)$  at 845 MeV is given below:

```
reaction c (e,e)
e0 r 1 845.0
particle e c lepton
spece r 5 3.4730 +16.00 0.0 0.030 0.060
target r 5 0.05580 2.20 2 1 1
tar_angle r 1 +8.00
isotope1_name c 12C
isotope1 r 3 0.989 12.000 6
isotope2_name c 13C
isotope2 r 3 0.011 13.003 6
```

The order of the lines is not important. They are built up in the following way. The first string indicates the name of the item. It is followed by a character which should be either *c* which stands for character string or *r* which indicates that real-number will follow. The next number gives the number of fields if a real-number follows. Finally, the values of the item should be given. Table A.1

Table A.1: Description of header file items.

item	type	field	units	description
reaction*	c			reaction string. Possible entries are: g (photon), n (neutron), p (proton), d (deuteron), t (triton), h+ ( ${}^3\text{He}^+$ ), h++ ( ${}^3\text{He}^{++}$ ), a+ ( ${}^4\text{He}^+$ ), a++ ( ${}^4\text{He}^{++}$ , $\alpha$ ), pi+ ( $\pi^+$ ), pi0 ( $\pi^0$ ), pi- ( $\pi^-$ ), mu+ ( $\mu^+$ ), mu- ( $\mu^-$ ), e or e- (electron), e+ (positron), k+ ( $K^+$ ), k0 ( $K^0$ ), k- ( $K^-$ ).
e0*	r		MeV	electron beam energy
particle_e*	c			particle in HRSE
particle_h*	c			particle in HRSH
spece*	r	field angle  solid	kG deg deg  rad rad	parameters of the electron spectrometer magnetic induction of the dipole angular position of spectrometer in-plane angular position of spectrometer out-of-plane spectrometer collimator acceptances half horizontal acceptance half vertical acceptance
spece_offset	r	x y z	m m m	position of the electron spectrometer optical point in the hall coordinate system x component
spech*	r	field angle  solid	kG deg deg  rad rad	parameters of the hadron spectrometer magnetic induction of the dipole angular position of spectrometer in-plane angular position of spectrometer out-of-plane spectrometer collimator acceptances half horizontal acceptance half vertical acceptance
spech_offset	r	x y z	m m m	position of the hadron spectrometer optical point in the hall coordinate system x component
target*	r			target parameters

continued on next page ...

Table A.1: ... continued from previous page ...

item	type	field	units	description
		d	g/cm <sup>2</sup> (cm)	total target thickness (model 1), target radius (model 2), target length (model 3). Is ignored for model 1 if foils are specified.
		rho	g/cm <sup>3</sup>	target density
		niso		number of isotopes
		isop		isotope being investigated
		model		target model (1:foil; 2:circular; 3:cylindrical)
		diam	cm	diameter of cylinder (only model 3)
foil	r	d	g/cm <sup>2</sup>	target foil parameters
		width	cm	target thickness
		pos_x		foil width
		pos_y		
		pos_z	cm	distance of foil from target center
tar_angle*	r	tangle	deg	target angle parameters; only for model=1
		fangle	deg	angle between target coordinate system z-axis and HLCS z-axis
				angle between normal to target foil and target c.s. z-axis
tar_offset	r			position of target center in the hall coordinate system; it is assumed that this is also the center of rotation for the target.
		x	m	x component
		y	m	ignored for target model 1
		z	m	
isotope1_name*	c			name of isotope 1
isotope1*	r	f		abundance ( $\leq 1.0$ )
		A	amu	atomic mass
		Z		isotope charge
isotope2_name*	c			name of isotope 2
isotope2*	r	f		abundance ( $\leq 1.0$ )
		A	amu	atomic mass
		Z		isotope charge
reconstruct	r			parameter controlling extended target corrections

continued on next page ...

Table A.1: ... continued from previous page ...

item	type	field	units	description
	0			It is assumed that the reaction happened in the center of the target. No <code>x_tg</code> correction is applied.
	1			The position information of the beam and the vertex reconstructed with the spectrometers are used to calculate the reaction point in the target. This information is used to calculate the energy loss of the beam before and of the emitted particles after the reaction.
	2			In addition to option 1 the vertices of all spectrometers are corrected for the fact that the particle did not emerge from the optical point <sup>1</sup> .
beam_vertex	r			parameter controlling the beam position determination. Section 6.11 describes how the beam position is computed.
	0			It is assumed that the beam is on the ideal axis, i.e. the positions and angles are 0.
	1			The position information of the BPMs is used to compute the beam vector ( $x$ , $y$ , $\phi$ , and $\theta$ ) at the Hall A center ( $z=0$ ).
	2			The deflection information of the raster is used to compute the beam position ( $x$ , $y$ ) at the Hall A center. The angles are assumed to be 0. Please note that this method does not give the real beam position, but the deviation of the current beam position from the ideal one.
eloss	r			energy loss, additional to the energy loss in the target ...
		loss_b	MeV	... of the beam before the reaction
		loss_e	MeV	... of the particle detected in HRSE
continued on next page ...				

<sup>1</sup>This is realized in the following way: a) the matrix elements are used to transform the focal plane coordinates into target coordinates. b) `x_tg`, the vertical position at the target, is calculated and replaces `dp`. c) “forward matrix elements” (determined from a SNAKE simulation) are used to transform those coordinates back into the focal plane. d) those manipulated focal plane coordinates are transformed into the final target coordinates.

Table A.1: ... continued from previous page ...

item	type	field	units	description
		loss_h	MeV	... of the particle detected in HRSH If only three parameters are specified, ES- PACE does not distinguish between passing the cylinder wall or the end cap of the tar- get cell (model 3). Two optional parame- ters, useful for cylindrical targets. For the other target models they are ignored. Here the two latter parameters are used only if the particle exits trough the cylinder wall.
		loss_e	MeV	... of the particle detected in HRSE, if passing the end cap
		loss_h	MeV	... of the particle detected in HRSH, if passing the end cap

gives a description of each item and its entries. Note that items followed by an '\*' are mandatory, you will have to give it in the header file, otherwise an error message will appear.

All values are initialized on zero. Although `spece` and `spech` are mandatory items, entry of one will suffice, the same is true for the isotope information.

In Fig. A.1 the configuration of the three target models is shown. Indicated are the different parameters describing the target. Realize that if target model 1 is chosen and a foil card is not given, the number of foils is assumed to be 1 with a target thickness as given in the target card. When the foil card is given, the target thickness in the target card is ignored.

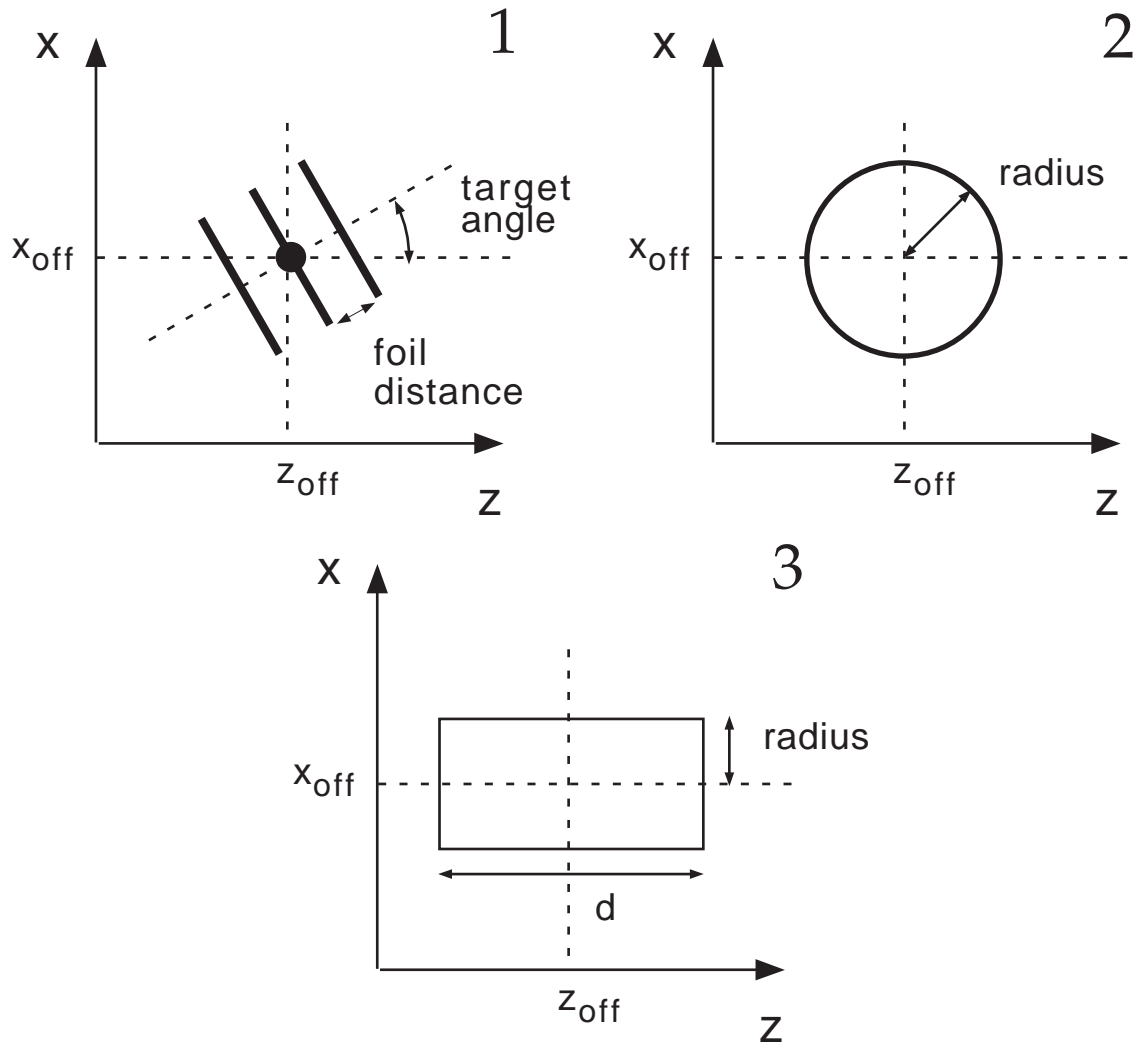


Figure A.1: Shown are the various target models and their parameters.

# Appendix B

## Database File Description

The database file contains constants of the spectrometers and their detectors.

The data base is separated into a section describing the HRSE, and into a section describing the HRSH. For the beam position reconstruction, it was necessary to include information about the raster magnet and the BPM too. Usually, this precedes the HRSE section, except for single HRSH data acquisition..

Each section contains technically two kinds of records, those which are preceded by a key, and those which are not. Records with a key are identified by this record, and the following fields are stored in specific variables. The order of these records can be arbitrary, in principle, though it wouldn't be very collegial. To actually read a record, ESPACE uses the CERN package `ffread`. One drawback of `ffread` is, that it reads only up to 80 characters per line. Thus, eventually a line has to be broken up into several lines, like in the example in Sec. B.1. Be aware that in case of an optimization the new data base is written in a specific format, so that the original line may suddenly exceed the 80 character limit and may get truncated. Following the subsection containing the "key" records, comes a long list of numbers. These are read sequentially, and adding or removing a line or even a single field messes things up. This feature of the data base makes modifications and compatibility of different ESPACE versions extremely difficult.

### B.1 Raster and Beam Position Information

I will describe the first three (five lines) in the header file, which contain the information ESPACE needs to determine the beam position.

For ESPACE version 2.9.0 or higher, the first three keys of the data base describe the raster and the two BPMs, e.g:

```
raster    0.0  0.0 -23.0  18.3  22.6  2065  2054  1.6E-6  1.1E-6
bpma      -0.0009 +0.00035 -7.607  2105  2104  2098  2098  1.0  1.0  0.01887  1.12  1.12
1.1  1.35
bpmb      -0.00015 -0.0003 -1.370  2090  2085  2085  2086  1.0  1.0  0.01887  1.02  1.02
1.1  1.35
```

For each key, the first three fields  $x_{\text{pos}}$ ,  $y_{\text{pos}}$ , and  $z_{\text{pos}}$  describe the location of the device in the Hall A coordinate system, in meters.

For the BPMS, fields 4-7  $X_{\text{p,ped}}$ ,  $X_{\text{m,ped}}$ ,  $Y_{\text{p,ped}}$ , and  $Y_{\text{m,ped}}$  are the pedestals of the ADCs for the individual antennas, 8  $\alpha_x$  and 9  $\alpha_y$  the relative gains of the opposing antennas, 10  $\kappa$  (m/ADC channel) the conversion of ADC value into a length, 11  $\kappa_x$  and 12  $\kappa_y$  the attenuation of the ADC signal due to the 30 kHz Filter, and 13  $\phi_x$  and 14  $\phi_y$  (rad) the phase difference of the BPM read-out versus the event trigger.

For the raster, fields 4  $\nu_x$  and 5  $\nu_y$  (kHz) are the raster frequencies, 6  $X_{\text{ped}}$  and 7  $Y_{\text{ped}}$  the pedestals, and 8  $\kappa_x$  and 9  $\kappa_y$  (m/ADC channel) the conversion of the ADC value into a length.

## B.2 The other Stuff

Unfortunately, the data base chapter was never updated since February 1997, and probably never will. From here, it needs a serious revision. Sorry<sup>1</sup>.

We describe here how the part for spectrometer HRSE looks like, the part for the other spectrometer has the same structure.

The first seven lines of the database file contain drift chamber information and should look as follows:

```
10.0 10.0 10.0 10.0
5.7e+04 5.7e+04 5.7e+04 5.7e+04
4096. 4096. 4096. 4096.
5.0e-10 5.0e-10 5.0e-10 5.0e-10
2.0 0.0 0.0 0.0 -4.479051e-5 1.588897e-4 3.861646e-4 -6.223269e-5
```

The meaning of these lines is:

line	Description
1	estimate of drift time resolution (chan)
2	electron drift velocity (m/s)
3	range of the VDC-TDC
4	VDC-TDC resolution (s/chan)
5	coefficients of a polynomial for drift length correction in the VDC(m)

Next lines contain for each wire plane and each wire, the wire number and the TDC offsets. Each line contains information for five wires, f.i.:

```
1 503.0 2 503.0 3 503.0 4 503.0 5 503.0
```

If the wire number is negative then this wire will not be used in the analysis. This should be followed by lines containing wire number and position:

---

<sup>1</sup>Michael Kuss, August 9, 2000

1 1.42057 2 1.41557 3 1.41057 4 1.40557 5 1.40057

Now information about the trigger detectors comes (first s1, followed by the same information for s2)

```
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
0.700
1.9e+8
1.0e-10
0.00 42.46 37.18 36.65 39.97 36.02
30.57 16.25 37.34 18.28 74.78 66.88
0.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
```

The meaning of these lines is:

line	Description
1-2	Parameters for correction of amplification of the ADC signal for each Čerenkov mirror. For each mirror first the offset is given and then an amplification factor.
3	Inverse of attenuation length of light in the scintillator ( $\text{m}^{-1}$ )
4	Speed of light in scintillator (m/s)
5	TDC resolution in scintillator plane (s/chan)
6-7	Time offset of each scintillator paddle on both sides (chan)
8	Parameter for correction timewalk effects in the scintillator timing with the ADC signal
9-10	Parameters for correction of amplification of the ADC signal for paddle on the right side, followed by the left side. For each paddle an offset and an amplification factor is given.

At the end of the database the transport information of the spectrometer is given. The first number is the radius of the central ray through the dipoles. Then the coefficients for polynome of the spectrometer constant  $\Gamma$  (in MeV/kG) are given and it is followed by the dispersion constants  $d_i$  which together link dispersive position  $x_{\text{fp}}$  and momentum  $p$  through the relation:

$$p = \left( \sum_{i=0}^2 \Gamma_i B^i \right) \left( \sum_{i=1}^6 d_i x_{\text{fp}}^i \right)$$

The number following the dispersion constants indicates the definition of the transport matrix:

#	Matrix definition
1	Transport elements using focal plane coordinates defined wrt. the local central ray
2	Transport elements using focal plane coordinates defined wrt. the detector system

1.400

41.972 0.0 0.0

1.2063E-01 1.1105E-02 -1.0599E-03 -1.1160E-04 -1.2408E-04 0.0

2

It is completed with a list of the transfer elements. Each line contains the following information. The first integer tells whether the transfer element will be fitted when an OPTIMIZE command has been given. A 0 means keep fixed, a 1 means fit it and -1 indicates that when reconstructing the vertex at the target the first time use the values in the database but keep them fixed on zero during the fit. This is followed by the  $x_{fp}$  dependence of the particular transfer coefficient and its name:

$$M(x_{fp}) = \sum_{i=1}^6 M_i x_{fp}^i$$

For example, the line for  $\langle\theta|\theta\rangle$  looks like

1 1 0 0 -5.748E-1 1.254E-3 -1.028E-5 6.655E-8 0.000E+0 -1.21E-12 'T100'

# Appendix C

## Transformation between Different Coordinate Systems

In this section we will use the following abbreviations for the different coordinate systems:

SDCS Spectrometer Detector Coordinate System

SFPCS Spectrometer Focal Plane Coordinate System

SRCS Spectrometer Reconstructed Coordinate System

HLCS Hall A Laboratory Coordinate System

In Tab. C.1 a set of FORTRAN routines is described that transforms a vector between the different coordinate systems. These routines are part of the kinematics package. Sources are located in the directory `$HALLA_DIR/src/kinematics` and the library is in the directory `$HALLA_DIR/lib/$OSNAME`.

Table C.1: FORTRAN routines used to transform vectors between the different ESPACE coordinate systems.

Routine	Description
<code>beam_to_transport</code>	This routine transforms a HLCS-vector in spherical coordinates to the SRCS in transport coordinates. This transformation is defined by a call to the routine <code>define_beam_to_transport</code> .
<code>define_beam_to_transport</code>	Initialization routine for the transformation used in beam to transport.
<code>transport_to_beam</code>	This routine converts a SRCS-vector in transport coordinates to the HLCS in cartesian coordinates. This transformation is defined by a call to the routine <code>define_trans</code> .
<code>define_transport_to_beam</code>	Initialization routine for the transformation used in transport to beam.
<code>ctrans</code>	Routine transforming geographical angles to spherical angles and vice versa.
<code>spherang</code>	Routine transforming geographical angles to spherical angles and vice versa. This routine gives also the option to switch HLCS and a variant of the HLCS with the $z$ -axis along the momentum transfer $q$ .
<code>newang</code>	

# Appendix D

## ESPACE Reference Manual

Event Scanning Program for hall A Collaboration Experiments

### D.1 Espace/Debug

This menu contains several switches controlling output for debugging purposes. At the moment the CODA event stream and the VDC tracks can be monitored.

```
vdc [vdc]    vdc  C  "vdc tracking"  D='off'
```

Possible **vdc** values are:

off

on

Switch the VDC debug mode. In the VDC debug mode, a graphics window will pop up and for each CODA event a track will be shown for each cluster in the U1,V1,U2 and V2 plane.

```
coda [coda]  coda  C  "coda"  D='off'
```

Possible **coda** values are:

off

on

Switch the CODA debug mode. In the CODA debug mode, part of the raw CODA data is displayed.

```
sync [sync]  sync  I  "synchronization"  D=0 R=0:3
```

Switch the synchronization debug mode. In the SYNC debug mode, CODA information is displayed together when synchronization problems appear.

**hbook** [**hbook**] **hbook** C “hbook” D=’off’

Possible **hbook** values are:

**off**

**on**

Switch the HBOOK debug mode. In the HBOOK debug mode, HBOOK memory headers are displayed.

## D.2 Espace/Set

Menu to assign parameters to ESPACE variables.

**test** [**test**] **test** C “test mode” D=’off’

Possible **test** values are:

**on**

**off**

Switch the test mode. If the test mode is on, the command syntax and the existence of data files is checked. No scanning or optimizing is performed.

**auto\_window** [**awindc**] **awindc** C “auto-windowing” D=’on’

Possible **awindc** values are:

**on**

**off**

Switch the automatic histogram windowing. If the auto-window mode is on, the **LOW\_X**, **HIGH\_X** and **NUM\_X** parameters are put on the default values for the particular variable to be histogrammed.

**hitbits** **hitbits** **hitbits** C “hitbits” D=’’

Choose the condition, which detector should have fired or not, that is imposed on all histograms defined after this point.

**ntuple** [**ntuple**] **ntuple** C “NTUPLE format” D=’off’

Possible **ntuple** values are:

**on**

**off**

If the NTUPLE switch is on, the data are store in NTUPLE format, otherwise it will be a multi-dimensional histogram.

**cosmic\_rays** [**cosmic\_rays**] **cosmic\_rays** C “cosmic corr.” D=’off’

Possible **cosmic\_rays** values are:

on

off

If **COSMIC\_RAYS** is switched on, the VDC TDC drift times are corrected with an opposite sign since the cosmic rays drift from timing scintillator to VDC.

### D.2.1 Espace/Set/file

Assign file names.

**header** **hdr\_file** **hdr\_file** C “header file” D=’\_’

Set the name of the header file.

**database** **db\_file** **db\_file** C “database file” D=’\_’

Set the name of the database file.

**opt\_database** **optdb\_file** **optdb\_file** C “opt database file” D=’datab\_opt’

Set the name of the optimized database file in which results are stored of an optimization triggered with the FIT command.

**detmap** **detmap\_file** **detmap\_file** C “detmap-config file” D=’tstand\_detmap.config’

Set the name of the detector configuration file. This file describes how the raw data of the data acquisition code CODA should be interpreted.

**fit\_input** **fit\_file** **fit\_file** C “fit input file” D=’fit.inp’

Set the name of the input file containing parameters for optimization. This file contains starting parameters for fitting **th\_tg**, **ph\_tg**, **y\_tg**, **t\_c**, **e\_miss** and **e\_miss/p\_miss**.

**output** **output\_file** **output\_file** C “output file” D=’espace.hbook’

Set the name of the HBOOK output file containing histograms and Ntuples.

### D.2.2 Espace/Set/timing

Assign trigger definition parameters.

```

e plane [ side ]   plane  C  "scintillator plane"  D='s2'
                    side   C  "paddle side"    D='right'

```

Possible **plane** values are:

**s1**

**s2**

Possible **side** values are:

**left**

**right**

Choose the number scintillator plane and the paddle side that will define the trigger timing for e-spectrometer.

```

h plane [ side ]   plane  C  "scintillator plane"  D='s1'
                    side   C  "paddle side"    D='left'

```

Possible **plane** values are:

**s1**

**s2**

Possible **side** values are:

**left**

**right**

Choose the number scintillator plane and the paddle side that will define the trigger timing for h-spectrometer.

### D.2.3 Espace/Set/type

Assign CODA event types

```

e type   type  C  "e-spec type"  D='1,2'

```

Choose the CODA event type for single e-spectrometer event.

```

h type   type  C  "h-spec type"  D='3,4'

```

Choose the CODA event type for single h-spectrometer event.

```

coin type  type  C  "coin type"  D='5'

```

Choose the CODA event type for coincidence event.

```

scal type  type  C  "scaler type"  D='140'

```

Choose the CODA event type for scaler event.

## D.2.4 Espace/Set/fit

Assign fit parameters.

```
npts npts_file  npts_file  I  "max points/file"  D=10000  R=0:5000000
Set the maximum number of data points accumulated in one file for optimization.
```

### D.2.4.1 Espace/Set/fit/Adc

Assign control parameters for the scintillator ADC value calibration.

```
Pedestal [fit_pedestal]  fit_pedestal  C  "fit adc offsets"  D='on'
```

Possible `fit_pedestal` values are:

```
on
off
```

Fit the adc offsets of the scintillators when `adc_{side}_c` is being optimized.

```
Gain [fit_gain]  fit_gain  C  "fit adc gains"  D='on'
```

Possible `fit_gain` values are:

```
on
off
```

Fit the adc gains of the scintillators when `adc_{side}_c` is being optimized.

### D.2.4.2 Espace/Set/fit/time

Assign control parameters for the spectrometer path length fit.

```
paddle_e [fit_paddle_e]  fit_paddle_e  C  "fit time-offset paddles spec e"  D='off'
```

Possible `fit_paddle_e` values are:

```
on
off
```

Fit the time offsets of the scintillator used in the trigger of the electron spectrometer when `tc_cor` is being optimized.

```
paddle_h [fit_paddle_h]  fit_paddle_h  C  "fit time-offset paddles spec h"  D='off'
```

Possible `fit_paddle_h` values are:

```
on
off
```

Fit the time offsets of the scintillator used in the trigger of the hadron spectrometer when `tc_cor` is being optimized.

```

time_walk [ fit_timewalk order ]   fit_timewalk  C  "fit time walk coefficients"  D='off'
                                         order          I  "order of polynome"    D=1 R=1,2,3

```

Possible `fit_timewalk` values are:

```

on
off

```

Fit the time walk of the timing scintillator signals. It is assumed that it is the same for all scintillators of the electron and hadron spectrometer when `tc_cor` is being optimized.

### D.2.4.3 Espace/Set/fit/momentum

Assign control parameters for the spectrometer momentum fit.

```

gamma_e [ fit_gamma_e ]   fit_gamma_e  C  "fit gamma spec E"  D='off'

```

Possible `fit_gamma_e` values are:

```

on
off

```

Fit the electron spectrometer constant defining the central momentum.

```

gamma_h [ fit_gamma_h ]   fit_gamma_h  C  "fit gamma spec H"  D='off'

```

Possible `fit_gamma_h` values are:

```

on
off

```

Fit the hadron spectrometer constant defining the central momentum.

## D.2.5 Espace/Set/spectrum

Set histogram parameters.

### D.2.5.1 Espace/Set/spectrum/window

Set range.

```

x1 low high   low  R  "x1 lower limit"
                high R  "x1 upper limit"
LOW and HIGH value of the x1 variable

```

```

x2 low high   low  R  "x2 lower limit"
                high R  "x2 upper limit"
LOW and HIGH value of the x2 variable

```

**x3 low high**    low    R    “x3 lower limit”  
                          high    R    “x3 upper limit”  
 LOW and HIGH value of the x3 variable

**x4 low high**    low    R    “x4 lower limit”  
                          high    R    “x4 upper limit”  
 LOW and HIGH value of the x4 variable

### D.2.5.2 Espace/Set/spectrum/bins

Set number of bins.

**x1 num**    num    I    “number of bins x1”    D=1  
 Number of X1 bins for REAL variable, binsize for INTEGER variable.

**x2 num**    num    I    “number of bins x2”    D=1  
 Number of X2 bins for REAL variable, binsize for INTEGER variable.

**x3 num**    num    I    “number of bins x3”    D=1  
 Number of X3 bins for REAL variable, binsize for INTEGER variable.

**x4 num**    num    I    “number of bins x4”    D=1  
 Number of X4 bins for REAL variable, binsize for INTEGER variable.

### D.2.6 Espace/Set/vdc

Assign VDC analysis parameters.

**iterate iterate\_vdc**    iterate\_vdc    I    “iterate VDC”    D=1 R=1:4  
 Choose the number of iterations in the determination of a track through a VDC plane. If ITERATE\_VDC > 1 the global angle coming from two wire planes is used in the calculation of several corrections.

**trig\_cor [ trig\_cor ]**    trig\_cor    C    “vdc COMMON STOP corr.”    D= 'on'

Possible **trig\_cor** values are:

on

off

If TRIG\_COR is switched on, the VDC TDC drift times are corrected for time of flight to the timing scintillator and signal propagation through this scintillator.

```

tx_cor icor [ spectro tx_table ]   icor      I  "vdc t-x switch"  D=1 R=0:3
                                   spectro    C  "spectrometer"   D='e'
                                   tx_table   C  "t-x HBOOK lookup table" D='tx_table.hbook'

```

Possible `spectro` values are:

e

h

The `TX_COR` parameter controls the drift time to position correction in the VDC's.

```

tdc_cut tdc_cut tdc_cut I "TDC cut option" D=0 R=0,1,2

```

With the `TDC_CUT` command the range of drift times that will be used in the VDC analysis can be controlled. The default setting is zero. The different cut options are (larger number means more restrictive condition):

0: For each wire only the largest TDC value is stored. In `COMMON STOP`

mode this corresponds (no noise !) to the shortest drift time.

1: 0 + TDC values larger than the TDC offset are not used (corresponding

to negative drift times)

2: 0 + 1 + TDC values corresponding to drift distances 50% larger than

than the wire plane / cathode distance are not used.

Notice that in options 1 and 2 the TDC offset for the corresponding wire is used. So an incorrect offset will result in loss of data !!

## D.3 Espace/Define

Menu to define logicals and variables.

```

init      Set all parameters back on their default values.

```

```

batch [ batch ]  batch C "batch mode" D='off'

```

Possible `batch` values are:

on

off

Stop execution if an error condition has occurred.

```
reset hist hist C "hist list" D='all'
```

Empty contents of histograms in list. Correct syntax for the list is e.g. all: empty all histograms, 1-3,10: histograms 1,2,3 and 10. The numbers refer to the histogram HBOOK identifier.

```
logical name str name C "logical name"
                str C "logical expr"
```

Define a new logical by combining existing logical through the following operations:

```
! negate
&& logical AND
|| logical OR
```

The order of operations can be guided by brackets.

```
variable str str C "expression"
```

With this command, constants and new variables can be defined.

New variables are defined by combining existing variables through the following operations:

```
( ) + - * / ^ **
```

and intrinsic functions

```
abs      acos      acosd     asin      asind     atan      atand     cos
cosd     cosh      deg       exp       log       log10     rad       sin
sind     sinh      sqrt     tan       tand      tanh
```

The definition string should look like, NEW : operation(OLD1,OLD2). Notice the assignment with ':':

Constants are defined by a string like, PI=3.142 . Notice the assignment with '='.

### D.3.1 Espace/Define/cut

Define cuts n variables calculated on a event-by-event base.

```

var_1d C "1d variable string" D='□'
1d var_1d xmin xmax xmin R "low range" D=-100.
xmax R "high range" D=100.

```

Define a window on a variable. This definition results in a logical which is TRUE if the variable value is inside the window and FALSE otherwise. The variable can be a scalar or an array. If it is an array the logical is TRUE if the contents of the whole array fulfill the condition.

Examples:

```

scalar cut: define/cut/1d spec_e.ph_tg -0.03 0.03
array cut: define/cut/1d spec_e.u1.tdc 2000 2500
define/cut/1d spec_e.u1.tdc[100;200] 2000 2500
The last cut is performed on the TDC values of wires 100 upto 200.

```

```

var_2d C "2d variable string" D='□'
2d var_2d direction par direction C "cut direction" D='above'
par C "polynome parameters" D='1,0,0'

```

Possible `direction` values are:

above

below

Define a region in 2 dimensions. This definition results in a logical which is TRUE if the combination of the values of the 2 variables is on the right side of the given polynomial, which is either ABOVE or BELOW. Again variables can be scalars or arrays.

Example:

```
define/cut/2d spec_e.ph_tg/spec_e.th_tg above 0.,1.,0.
```

```

polygon var_2d file var_2d C "2d variable string" D='□'
file C "cut data file" D='cut.dat'

```

Define a polygon in 2 dimensions. This definition results in a logical which is TRUE if the combination of the values of the 2 variables is inside this polygon.

Example:

```
define/cut/polygon spec_e.ph_tg/spec_e.th_tg acceptance.dat
```

## D.4 Espace/Spectra

Histograms commands.

```

save spec [ condition ]   spec      C  "spectrum list"  D='□'
                           condition C  "logical expr"  D='□'

```

Define a histogram of a particular variable. In this command several histograms can be defined by giving a list of variables separated by commas. After the the variable list a logical expression can be given which has to be fulfilled for incrementing the histograms.

Examples:

```

1-dim histogram: spectra/save spec_e.p_kin spec_e.ph_tg&&spec_e.th_tg
2-dim histogram: spectra/save spec_e.th_fp/spec_e.p_kin
spec_e.ph_tg&&spec_e.th_tg

```

a 'set/ntuple on' before the last save statement would result in a ntuple instead of a histogram.

1-dim histogram of an array variable:

```

spectra/save spec_e.u1.tdc[100;200]

```

This results in 1-dim histogram containing the TDC values of wires 100-200.

```

DO I = 1,368

```

```

  ISTA = [I]

```

```

  ISTO = [I]

```

```

  spectra/save spec_e.u1.tdc[[ISTA];[ISTO]]

```

```

ENDDO

```

This will give you a TDC histogram of each wire in the U1 VDC plane of the electron spectrometer. Notice that the KUIP macro language has been used.

```

rotate list rot   list  C  "list of spectra"  D='1-100'
                   rot   I  "rotation number"  D=1

```

Do not use this command.

```

write [ list ]   list  C  "list of spectra"  D='1-4999'

```

Save the defined spectra to a file. Correct syntax for the spectra list is e.g. all: save all histograms, 1-3,10: histograms 1,2,3 and 10. The numbers refer to the histogram HBOOK identifier.

## D.5 Espace/File

Menu containing commands related to the data file.

```

file      C  "file name"  D='_'
first     I  "first event" D=1
scan file [ first last chopt rvf ] last      I  "last event"  D=-1
chopt     C  "options"    D='_'
rvf       C  "review file" D='review.dat'
```

Possible `chopt` values are:

'\_'

- o fit : Start fitting after scanning file
- r review : review file
- w write review : write review report after this file
- f format : not rawdata input
- l loop : Do not open the file again but continue scan

Scan a file. Start a scanning session in which the previously defined histograms and logicals are updated event-by-event with the specified file. In the scan command parameters like the first interrupt, last interrupts and several options can be specified.

Among the options are for instance the possibility to incorporate the data from this file in an optimization (specified before through a `calibrate/fit` statement) or to specify the format of the incoming data.

Example:

```
file/scan file.dat 1 1000 -f
```

In this example the first 1000 interrupts of the file 'file.dat' are analyzed. The data is not in raw format but specified by the routine `FILTER_TIL_EOF`

```

file      C  "file name"  D='_'
first     I  "first event" D=1
review file [ first last chopt rvf ] last      I  "last event"  D=-1
chopt     C  "options"    D='_'
rvf       C  "review file" D='review.dat'
```

Possible `chopt` values are:

'\_'

- w write review : write review report after this file
- f format : input data format

Review the detector coincidences. Issuing this command results in a review file in which a summary is given of the hit pattern of the detectors: how many times did a detector fire and how many coincidences and between which detectors are there.



```
optimize spec [ condition ]   spec      C  "spectrum name"  D='□'
                              condition C  "logical expr"   D='□'
```

Start an optimization. Possible variables that can be optimized:

spec_e.dp_kin	Relative momentum after correction for   aberations and kinematic broadening.
spec_e.th_tg/spec_e.ph_tg	Reconstructed theta and phi target coordinate   of the electron spectrometer.
spec_h.th_tg/spec_h.ph_tg	Reconstructed theta and phi target coordinate   of the hadron spectrometer.
spec_e.y_tg	Reconstructed y target coordinate of the   the electron spectrometer.
spec_h.y_tg	Reconstructed y target coordinate of the   the hadron spectrometer.
tc_cor	Coincidence time between e and h spectrometer   after correction for time-of-flight and paddle   offsets.
e_miss	Missing energy.
e_miss/p_miss	Missing energy and momentum for H(e,e'p).
spec_e.s1.adc_l_c	
spec_e.s1.adc_r_c	
spec_e.s2.adc_l_c	
spec_e.s2.adc_r_c	
spec_h.s1.adc_l_c	
spec_h.s1.adc_r_c	
spec_h.s2.adc_l_c	
spec_h.s2.adc_r_c	
spec_h.s3.adc_l_c	
spec_h.s3.adc_r_c	
spec_e.s1	
spec_h.s2	

A LOGICAL condition can be specified for selection of the data to be used in the optimization.

# Appendix E

## Flow Diagram

The ESPACE sources were analyzed with the source code analyzer FORCHECK. From that output this flow diagram was created automatically. To date, there are several problems with that automatic generation.

- EXTERNAL statements and function declarations (like REAL Sqrt) in a calling routine are treated as a call to that function.
- Functions which are parameters in a call of another function are treated as if both are called sequentially, rather than the called is executed by the calling.
- The flow diagram may be misleading because due to the usage of the KUIP interpreter in ESPACE routines are not called sequentially. One can consider this as an IF ... ELSEIF ... ELSE ... ENDIF construct.
- Some functions should be skipped due to non-relevance, e.g. like NOSPAC (stripping blanks off a string).

The tree of a called function is shown only once. For subsequent calls is referenced the position where the complete tree can be found. The literals !"#%&'()\*+,- are used to indicate the call level and to guide the eye scrolling through the flow diagram.

```
ESPACE          1
  ! READ_ARGLIST      2
    " IARGC           3
      " GETARG        4
        ! ESPACE_REG  5
          " INIT_COOLHANDS 6
            # NR_ITEMS  7
              $ GETPIECE  8
                % NSP1   9
                  % NSP2 10
                    % NSP1 (see 9)
```

```

        % NSP2          (see 10)
# VARIABLE          11
    $ VARIABLE_USER    12
# MINIT            13
# INIT_REVIEW      14
# FF_INIT          15
    $ FF_DEFVARF      16
        % EXMATCH      17
            & NOSPAC    18
                ' NSP1    (see 9)
                ' NSP2    (see 10)
                ' NSP1    (see 9)
                ' NSP2    (see 10)
        % FF_LOCIND    19
            & EXMATCH    (see 17)
            & VOF_CALC    20
                ' CODE_VAR    21
                    ( CODE1    22
                        ) CONDENSE    23
                            * NOSPAC    (see 18)
                        ) EXMATCH    (see 17)
                        ) NON_ANUMERIC    24
                            * NOSPAC    (see 18)
                        ) NOSPAC    (see 18)
                        ) NON_ANUMERIC    (see 24)
                        ) CONDENSE    (see 23)
                        ) NOSPAC    (see 18)
                        ) EXMATCH    (see 17)
                    ( CODE2    25
                        ) CONDENSE    (see 23)
                        ) REVERSE    26
                        ) EXMATCH    (see 17)
                        ) NON_ANUMERIC    (see 24)
                        ) NOSPAC    (see 18)
                        ) NON_ANUMERIC    (see 24)
                        ) CONDENSE    (see 23)
                        ) REVERSE    (see 26)
                        ) NOSPAC    (see 18)
                        ) EXMATCH    (see 17)
                        ) NOSPAC    (see 18)
                        ) EXMATCH    (see 17)
        % FF_EVAL      27

```

& VAR_FCALC	28
' USERVAR	29
( FF_EVAL	(see 27)
( VARIABLE	(see 11)
( FF_EVAL	(see 27)
( VARIABLE	(see 11)
& RPOP	30
& VAR_FCALC	(see 28)
& RPUSH	31
% NOSPAC	(see 18)
% VOFCALC	(see 20)
% EXMATCH	(see 17)
% FF_LOCIND	(see 19)
% FF_INTCHAR	32
& EXMATCH	(see 17)
& FF_LOCIND	(see 19)
& NOSPAC	(see 18)
& EXMATCH	(see 17)
& FF_LOCIND	(see 19)
% FF_TESTRANK	33
& FF_RANK	34
% FF_RPN	35
& CPOP	36
& IPOP	37
& FF_IPF	38
& FF_RANK	(see 34)
& FF_SPF	39
& FF_IPF	(see 38)
& FF_SPF	(see 39)
& IPOP	(see 37)
& CPOP	(see 36)
& FF_IPF	(see 38)
& FF_SPF	(see 39)
& FF_RANK	(see 34)
& IPUSH	40
& CPUSH	41
% FF_ELFUN	42
% FF_EVAL	(see 27)
\$ DINFO	43
\$ FF_DEFVARF	(see 16)
\$ DINFO	(see 43)
# DEFINE_VAR	44

```

$ NR_ITEMS      (see 7)
$ READSET      45
  % CONDENSE    (see 23)
  % NON_ANUMERIC (see 24)
  % CONDENSE    (see 23)
  % NON_ANUMERIC (see 24)
  % DERROR      46
# NR_ITEMS      (see 7)
# USER_INIT    47
$ RAN2          48
$ CPUBGN        49
  % INIT_TIMER   50
  % STAT_TIMER   51
  % INIT_TIMER   (see 50)
  % STAT_TIMER   (see 51)
$ RESET_ALLSCALERS 52
  % RESET_SCALER 53
  & EXMATCH     (see 17)
$ RAN2          (see 48)
$ VARIABLE_USER_INIT 54
  % DEFINE_VAR   (see 44)
# NOSPACE      (see 18)
# DERROR       (see 46)
! NOSPACE      (see 18)
! KUWHAT       55
" DEBUG_PARAMETER 56
  # SETLOG      57
  $ COMPARE     58
  % NOSPACE    (see 18)
  $ NOSPACE    (see 18)
  $ COMPARE     (see 58)
  # DERROR     (see 46)
  # SETLOG     (see 57)
  # PLOT_INIT   59
  # SETLOG     (see 57)
" SET_PARAMETER 60
  # GETERR     61
  # SETLOG     (see 57)
  # CONDENSE   (see 23)
  # DERROR     (see 46)
  # SETLOG     (see 57)
  # SET_BITS   62

```

```

    $ CONDENSE          (see 23)
    $ EXMATCH           (see 17)
    $ NON_ANUMERIC      (see 24)
    $ CONDENSE          (see 23)
    $ DERROR            (see 46)
    $ NON_ANUMERIC      (see 24)
    $ DERROR            (see 46)
    $ NOSPAC            (see 18)
    $ EXMATCH           (see 17)
    $ DERROR            (see 46)
# SETLOG               (see 57)
# CONDENSE             (see 23)
# NOSPAC               (see 18)
# OPEN_FILE            63
    $ DERROR            (see 46)
# CLOSE_FILE           64
# OPEN_FILE            (see 63)
# CONDENSE             (see 23)
# READSET              (see 45)
# GETERR               (see 61)
# SETLOG               (see 57)
# CONDENSE             (see 23)
# OPEN_FILE            (see 63)
# DINFO                (see 43)
# CLOSE_FILE           (see 64)
# SHORTSTRING          65
# DERROR               (see 46)
# SETLOG               (see 57)
" DEF_PARAMETER        66
    # SETLOG            (see 57)
    # DERROR            (see 46)
    # INIT_COOLHANDS    (see 6)
    # SETLOG            (see 57)
    # CRESET            67
        $ COMPAR        (see 58)
        $ GETERR        (see 61)
        $ MEXIST        68
        $ COMPAR        (see 58)
        $ MRESET        69
            % MIDENT    70
            % MEXIST    (see 68)
            % MERROR    71

```

% MIDENT	(see 70)
\$ READSET	(see 45)
\$ GETERR	(see 61)
\$ MEXIST	(see 68)
\$ DERROR	(see 46)
\$ MRESET	(see 69)
# DEFINE_LOGIC	72
\$ EXMATCH	(see 17)
\$ GETERR	(see 61)
\$ DERROR	(see 46)
\$ CODE_VAR	(see 21)
\$ DERROR	(see 46)
\$ READ_LOGIC	73
% EXMATCH	(see 17)
% GETERR	(see 61)
% INDX_CUT_SP	74
& SAMESPEC	75
& DERROR	(see 46)
% CONDENSE	(see 23)
% NOSPAC	(see 18)
% CONDENSE	(see 23)
% DERROR	(see 46)
% EXMATCH	(see 17)
% NOSPAC	(see 18)
% DERROR	(see 46)
% CODE_VAR	(see 21)
% DERROR	(see 46)
% CODE_VAR	(see 21)
% DERROR	(see 46)
% INDX_CUT_SP	(see 74)
% GETERR	(see 61)
% DERROR	(see 46)
% RPN	76
& IPF	77
& IPOP	(see 37)
& IRNK	78
& ISPF	79
& DERROR	(see 46)
& IPF	(see 77)
& ISPF	(see 79)
& IPOP	(see 37)
& IPF	(see 77)

```

        & ISPF          (see 79)
        & DERROR        (see 46)
        & IRNK          (see 78)
        & DERROR        (see 46)
        & IPUSH         (see 40)
        & DERROR        (see 46)
    $ GETERR           (see 61)
    $ EXMATCH          (see 17)
    $ NOSPAC           (see 18)
# DEFINE_VARF        80
    $ FF_INT           81
        % FF_ELALLFUN      82
            & FF_ELFUN      (see 42)
    $ FF_LOCIND        (see 19)
    $ DERROR           (see 46)
    $ FF_DEFVARF       (see 16)
    $ DERROR           (see 46)
    $ FF_INT           (see 81)
    $ DERROR           (see 46)
    $ FF_LOCIND        (see 19)
# GETCUTS_1D         83
    $ GETERR           (see 61)
    $ CODE_CUT         84
        % SAMESPEC         (see 75)
        % INDX_CUT_SP       (see 74)
        % CODE_VAR          (see 21)
        % DERROR           (see 46)
        % CODE_VAR          (see 21)
        % DERROR           (see 46)
        % SAMESPEC         (see 75)
        % INDX_CUT_SP       (see 74)
        % DERROR           (see 46)
    $ GETERR           (see 61)
# GETCUTS_2D         85
    $ COMPAR           (see 58)
    $ GETERR           (see 61)
    $ CODE_CUT         (see 84)
    $ GETERR           (see 61)
    $ COMPAR           (see 58)
# GETCUTS_POLYGON    86
    $ GETERR           (see 61)
    $ SHORTSTRING      (see 65)

```

```

    $ NOSPAC          (see 18)
    $ DERROR          (see 46)
    $ CODE_CUT        (see 84)
    $ GETERR          (see 61)
" DEF_SPECTRA      87
  # GETERR           (see 61)
  # CONDENSE         (see 23)
  # DERROR           (see 46)
  # CONDENSE         (see 23)
  # SAVE_IT          88
    $ GETERR          (see 61)
    $ READ_LOGIC     (see 73)
    $ GETERR          (see 61)
    $ NOSPAC         (see 18)
    $ CODE_SAVE      89
      % INDX_SAVE_SP  90
        & SAMESPEC    (see 75)
      % SAMESPEC      (see 75)
      % DERROR        (see 46)
      % CODE_VAR      (see 21)
      % DERROR        (see 46)
      % SAMESPEC      (see 75)
      % INDX_SAVE_SP  (see 90)
    $ GETERR          (see 61)
    $ MAP_CURRENT    91
      % CH_FUN        92
      % CH_SP         93
        & NOSPAC      (see 18)
      % REVERSE       (see 26)
      % MGETERR       94
      % OUTFILE       95
        & REVERSE     (see 26)
        & NOSPAC      (see 18)
        & REVERSE     (see 26)
      % CH_FUN        (see 92)
      % CH_SP         (see 93)
      % NOSPAC        (see 18)
      % REVERSE       (see 26)
      % NOSPAC        (see 18)
      % REVERSE       (see 26)
      % DERROR        (see 46)
      % NOSPAC        (see 18)

```

% MBOOK1	96
& MEXIST	(see 68)
& MIDENT	(see 70)
& MERROR	(see 71)
& MROOM	97
& MEXIST	(see 68)
& MIDENT	(see 70)
& MERROR	(see 71)
& MDELETE	98
' MEXIST	(see 68)
' MIDENT	(see 70)
' MINIT	(see 13)
' MEXIST	(see 68)
' MERROR	(see 71)
' MIDENT	(see 70)
& MERROR	(see 71)
& MCOMMENT	99
' MEXIST	(see 68)
' MERROR	(see 71)
% MBOOKN	100
& MEXIST	(see 68)
& MIDENT	(see 70)
& MERROR	(see 71)
& MROOM	(see 97)
& MEXIST	(see 68)
& MIDENT	(see 70)
& MDELETE	(see 98)
& MCOMMENT	(see 99)
% MBOOK2	101
& MEXIST	(see 68)
& MIDENT	(see 70)
& MERROR	(see 71)
& MROOM	(see 97)
& MEXIST	(see 68)
& MIDENT	(see 70)
& MERROR	(see 71)
& MDELETE	(see 98)
& MERROR	(see 71)
& MCOMMENT	(see 99)
% DERROR	(see 46)
% MGETERR	(see 94)
% DERROR	(see 46)

```

    $ GETERR          (see 61)
    $ ASSIGN_SPECTRUM      102
# GETERR          (see 61)
# CONDENSE        (see 23)
# ROTDET          103
    $ READSET        (see 45)
    $ DERROR         (see 46)
# CONDENSE        (see 23)
# WRITEOUT        104
    $ GETERR          (see 61)
    $ READSET        (see 45)
    $ GETERR          (see 61)
    $ NOSPACE        (see 18)
    $ PREPARE_INFO      105
        % ICHFUNG      106
        % RCHFUNG      107
        % CH_RPN       108
            & CH_SPEC      109
                ' CH_FUN      (see 92)
                ' CH_SP       (see 93)
                ' CH_FUN      (see 92)
                ' NOSPACE     (see 18)
            & DERROR       (see 46)
                % CH_SPEC     (see 109)
                % CH_RPN     (see 108)
                % CH_SPEC     (see 109)
                % RCHFUNG     (see 107)
                % ICHFUNG     (see 106)
                % RCHFUNG     (see 107)
                % ICHFUNG     (see 106)
                % MCOMMENT    (see 99)
    $ INSERT_HEADER      110
        % RCHFUNG          (see 107)
        % CHANGE_HEADER    111
        % MCOMMENT          (see 99)
        % NOSPACE          (see 18)
        % RCHFUNG          (see 107)
        % NOSPACE          (see 18)
        % MCOMMENT          (see 99)
# CLOSE_FILE          (see 64)
# OPEN_FILE           (see 63)
# GETERR              (see 61)

```

```

# MAKE_ALIAS          112
  $ GETERR            (see 61)
  $ CH_FUN            (see 92)
  $ CONDENSE          (see 23)
  $ NOSPAC            (see 18)
  $ READSET           (see 45)
  $ GETERR            (see 61)
  $ CONDENSE          (see 23)
  $ CH_FUN            (see 92)
  $ NOSPAC            (see 18)
  $ DERROR            (see 46)
# DINFO              (see 43)
# DERROR             (see 46)
" SCAN_FILE          113
# GETERR            (see 61)
# CONDENSE          (see 23)
# DERROR            (see 46)
# CONDENSE          (see 23)
# DERROR            (see 46)
# DINFO              (see 43)
# FF_ELALLFUN        (see 82)
# DERROR             (see 46)
# INPUT              114
  $ GETERR            (see 61)
  $ READ_DATABASE     115
    % NR_ITEMS        (see 7)
    % GETERR           (see 61)
    % SHORTSTRING     (see 65)
    % OPEN_DATABASE   116
    % CLOSE_DATABASE  117
    % DINFO           (see 43)
    % DERROR           (see 46)
    % OPEN_DATABASE   (see 116)
    % READ_GEOMETRY   118
      & CH_FUN         (see 92)
      & DERROR         (see 46)
      & COPY_GEOM_GL   119
      & COPY_GEOM_SB   120
    % GETERR           (see 61)
    % CLOSE_DATABASE  (see 117)
    % DERROR           (see 46)
    % NR_ITEMS        (see 7)

```

```

% DINFO          (see 43)
% DERROR         (see 46)
% NR_ITEMS       (see 7)
% DINFO          (see 43)
% NR_ITEMS       (see 7)
% DINFO          (see 43)
% DERROR         (see 46)
$ READ_HEADER    121
% EXMATCH        (see 17)
% GETPIECE       (see 8)
% INI_HEADER     122
  & GETPIECE     (see 8)
  & READ_HDRLINE 123
    ' GETPIECE   (see 8)
    ' DINFO      (see 43)
    ' GETPIECE   (see 8)
    ' DINFO      (see 43)
  & GETPIECE     (see 8)
  & DINFO        (see 43)
  & GETPIECE     (see 8)
  & DINFO        (see 43)
  & DERROR       (see 46)
% READ_HDRLINE   (see 123)
% INI_HEADER     (see 122)
% DERROR         (see 46)
% SHORTSTRING    (see 65)
% DINFO          (see 43)
% MCGIVE         124
  & MEXIST       (see 68)
  & MIDENT       (see 70)
  & MEXIST       (see 68)
  & MERROR       (see 71)
  & MIDENT       (see 70)
% GETPIECE       (see 8)
% DERROR         (see 46)
% READ_HDRLINE   (see 123)
% EXMATCH        (see 17)
% DERROR         (see 46)
% EXMATCH        (see 17)
% DERROR         (see 46)
$ SETUP_DET_LOGIC 125
% EXMATCH        (see 17)

```

```

$ DERROR          (see 46)
$ READ_HEADER     (see 121)
$ DERROR          (see 46)
$ HEADER_TO_PROGRAM      126
  % NOSPAC        (see 18)
  % PROCESS_REACTION      127
    & REACTION      128
      ' NOSPAC      (see 18)
      ' CAPITALIZE  129
      ' SPSET       130
      ' GET_MASS    131
        ( CAPITALIZE (see 129)
        ( ORDERNAME  132
        ( LOOKUP     133
      ' LOOKUP      (see 133)
    & DERROR        (see 46)
$ SETUP_DET_LOGIC (see 125)
$ GETERR          (see 61)
$ DERROR          (see 46)
$ READ_DATABASE   (see 115)
$ GETERR          (see 61)
$ DERROR          (see 46)
$ INI_RAWDATA     134
  % DETMAP_DATASTREAM      135
    & NOSPAC        (see 18)
  % NOSPAC        (see 18)
  % LOAD_DETMAP   136
    & INIT_DECODER      137
  % LIST_DETMAP   138
  % DETMAP_DATASTREAM (see 135)
  % DERROR        (see 46)
  % LOAD_DETMAP   (see 136)
  % LIST_DETMAP   (see 138)
  % INIT_RUNCOUNTERS      139
$ INI_SPEC_DETECTOR      140
  % FPP_INIT       141
$ INI_SPEC_TRANSPORT     142
  % POLY           143
  % CTRANS         144
  % POLY           (see 143)
$ SHORTSTRING          (see 65)
$ NOSPAC              (see 18)

```

```

$ RESET_ALLSCALERS          (see 52)
$ PROCESS_DATA              145
  % FILTER_TIL_EOF          146
    & INTERPRET_EVENT_COIN  147
      ' READ_CHER_E          148
        ( WHICH_CHER_MIRROR  149
      ' READ_SCINT_E         150
        ( WHICH_SCINT_PADDLE  151
      ' READ_SHOW_E          152
        ( WHICH_SHOW_BLOCK    153
          ( READ_SHOW_COEF    154
            ) SHCAL_COEF_IO   155
              * NOSPAC        (see 18)
            ( WHICH_SHOW_BLOCK (see 153)
      ' READ_VDC_E           156
        ( SELECT_PADDLE       157
        ( GET_FPD_VERTEX      158
          ) CALC_MATRIX       159
            * POLY            (see 143)
        ( GET_POS             160
        ( GET_WIRES           161
          ) RANGAUSS          162
            * RAN2            (see 48)
          ) HUNT              163
            ) RANGAUSS        (see 162)
        ( TDC_COR_INI         164
        ( TDC_COR_TRIG        165
          ) DIST_DET_VERTEX   166
            ) POLY            (see 143)
            ) CALC_MATRIX     (see 159)
            ) DIST_DET_VERTEX (see 166)
            ) POLY            (see 143)
        ( TDC_COR_TX1         167
          ) POLY              (see 143)
      ' READ_CHER_H          168
        ( WHICH_CHER_MIRROR   (see 149)
      ' READ_FPP_H           169
      ' READ_SCINT_H         170
        ( WHICH_SCINT_PADDLE   (see 151)
      ' READ_VDC_H           171
        ( SELECT_PADDLE        (see 157)
        ( GET_FPD_VERTEX       (see 158)

```

( GET_POS	(see 160)
( GET_WIRES	(see 161)
( TDC_COR_INI	(see 164)
( TDC_COR_TRIG	(see 165)
( TDC_COR_TX1	(see 167)
’ PULSER_FIRED	172
’ READ_HELICITY_H	173
( SUPERPERIOD_CHECK	174
’ READ_SCINT_E	(see 150)
’ READ_CHER_E	(see 148)
’ READ_SHOW_E	(see 152)
’ READ_VDC_E	(see 156)
’ READ_SCINT_H	(see 170)
’ READ_CHER_H	(see 168)
’ READ_VDC_H	(see 171)
’ READ_FPP_H	(see 169)
’ READ_HELICITY_H	(see 173)
’ PULSER_FIRED	(see 172)
& INTERPRET_EVENT_E	175
’ READ_CHER_E	(see 148)
’ READ_SCINT_E	(see 150)
’ READ_SHOW_E	(see 152)
’ READ_VDC_E	(see 156)
’ PULSER_FIRED	(see 172)
’ READ_SCINT_E	(see 150)
’ READ_CHER_E	(see 148)
’ READ_SHOW_E	(see 152)
’ READ_VDC_E	(see 156)
’ PULSER_FIRED	(see 172)
& INTERPRET_EVENT_H	176
’ READ_CHER_H	(see 168)
’ READ_FPP_H	(see 169)
’ READ_SCINT_H	(see 170)
’ READ_VDC_H	(see 171)
’ PULSER_FIRED	(see 172)
’ READ_HELICITY_H	(see 173)
’ READ_SCINT_H	(see 170)
’ READ_CHER_H	(see 168)
’ READ_VDC_H	(see 171)
’ READ_FPP_H	(see 169)
’ READ_HELICITY_H	(see 173)
’ PULSER_FIRED	(see 172)

```

& NOSPAC          (see 18)
& DERROR          (see 46)
& INTERPRET_EVENT_COIN      (see 147)
& INTERPRET_EVENT_E        (see 175)
& INTERPRET_EVENT_H        (see 176)
& DERROR          (see 46)
% GETERR          (see 61)
% RAWDATA_TIL_EOF      177
  & INTERPRET_EVENT_COIN      (see 147)
  & INTERPRET_EVENT_E        (see 175)
  & INTERPRET_EVENT_H        (see 176)
  & CORRUPT_SYNCH          178
    ' NOSPAC          (see 18)
    ' DERROR          (see 46)
  & INTERPRET_BPM_RAST      179
  & NOSPAC          (see 18)
  & DERROR          (see 46)
  & NOSPAC          (see 18)
  & CORRUPT_SYNCH          (see 178)
  & CONFIG_PARSE          180
  & SCALER_UPDATE          181
    ' SUPERPERIOD_CHECK      (see 174)
    ' CALC_CHARGE          182
    ' SUPERPERIOD_CHECK      (see 174)
    ' DERROR          (see 46)
    ' CALC_CHARGE          (see 182)
  & DERROR          (see 46)
  & GET_BCM          183
    ' GET_VALUE          184
  & INIT_EVENTCOUNTERS      185
  & EVENT_DECODER          186
  & DERROR          (see 46)
  & INTERPRET_BPM_RAST      (see 179)
  & INTERPRET_EVENT_COIN      (see 147)
  & INTERPRET_EVENT_E        (see 175)
  & INTERPRET_EVENT_H        (see 176)
  & READ_SPARE          187
% DOT3          188
% REACTION_Z          189
  & YTG_ERROR          190
    ' POW          191
  & TRANSPORT_TO_BEAM      192

```

```

      ' DERROR          (see 46)
    & YTG_ERROR        (see 190)
% RAWDATA_TIL_EOF      (see 177)
% FILTER_TIL_EOF       (see 146)
% GETERR              (see 61)
% DERROR              (see 46)
% FILTER_EVENT        193
  & DERROR            (see 46)
% PROCESS_SPECTROMETER 194
  & SPECTROMETER_RECONSTRUCT 195
    ' VDC_PRESORT      196
      ( ORDER_VDC_WIRES 197
      ( FIND_CLUSTERS   198
      ( VDC_PATTERN     199
    ' VDC_TRACKS       200
      ( FIND_TRACKS     201
        ) SETUP_TREE   202
          * FIT_TRACK   203
            + FDERIV_CLUS 204
              , DERROR   (see 46)
            + FUNCTN_CLUS 205
              , DERROR   (see 46)
            + CURFIT      206
              , FCHISQ   207
              , MATINV   208
                - DERROR (see 46)
                - LUDCMP 209
                - LUBKSB 210
              , PROTECT  211
              , FCHISQ   (see 207)
            + FDERIV_CLUS (see 204)
            + FUNCTN_CLUS (see 205)
          ) CLIMB_TREE   212
            * SAVE_PATH  213
          ) ORDER_PATHS  214
          ) FILL_TRACK   215
            * LAYER_TO_WC 216
              + DERROR   (see 46)
          ( TDC_COR_INI   (see 164)
          ( TDC_COR_TRIG  (see 165)
          ( TDC_COR_TX1   (see 167)
          ( TDC_COR_TX2   217

```

)	POLY	(see 143)	
(	TDC_COR_TX3		218
(	FINECH_FIT		219
)	POLFIT		220
*	MATINV	(see 208)	
(	FINECH_INTER		221
(	FINECH_FIT_CL		222
)	FDERIV_CLUS	(see 204)	
)	FUNCTN_CLUS	(see 205)	
)	CURFIT	(see 206)	
)	FDERIV_CLUS	(see 204)	
)	FUNCTN_CLUS	(see 205)	
)	CURFIT	(see 206)	
)	FDERIV_CLUS	(see 204)	
)	FUNCTN_CLUS	(see 205)	
(	FIND_TRACKS	(see 201)	
'	VDC_VERTEX		223
'	VDC_VERTEX_GLOB		224
'	CHERENKOV_PARAMETERS		225
(	DET_VERTEX		226
'	SCINT_CORRECTIONS		227
(	POLY	(see 143)	
(	DET_VERTEX	(see 226)	
(	POLY	(see 143)	
'	SELECT_PADDLE	(see 157)	
'	VDC_PRESORT	(see 196)	
'	VDC_TRACKS	(see 200)	
'	VDC_VERTEX	(see 223)	
'	VDC_VERTEX_GLOB	(see 224)	
'	CONSISTENT_SC_VDC		228
'	FP_VERTEX		229
(	CALC_MATRIX	(see 159)	
'	SCINT_CORRECTIONS	(see 227)	
'	UPDATE_SCINT_TDC		230
'	BETA_CALC		231
(	DIST_DET_VERTEX	(see 166)	
'	CHERENKOV_PARAMETERS	(see 225)	
'	UPDATE_SCINT_TDC	(see 230)	
'	BETA_CALC	(see 231)	
'	PLOT_TRACKS_GLOB		232
(	DERROR	(see 46)	
(	DEFINE_TRANSFORMATION		233

```

      ( FRAME          234
      ( LABEL          235
' TG_VERTEX          236
      ( CALC_VAR_TG    237
      ( POLY_INV       238
        ) POLY        (see 143)
      ( POW            (see 191)
      ( CALC_MATRIX    (see 159)
      ( CALC_VAR_TG    (see 237)
      ( POLY_INV       (see 238)
      ( DERROR         (see 46)
' CALC_COVARIANCE_FPD      239
' CALC_COVARIANCE_FPR      240
      ( POW            (see 191)
' CALC_COVARIANCE_FPT      241
      ( POW            (see 191)
' ADD_MS_COVARIANCE        242
      ( DIST_PLANE_VERTEX 243
      ( CALC_COVARIANCE_MS 244
      ( DIST_PLANE_VERTEX (see 243)
      ( CALC_COVARIANCE_MS (see 244)
      ( DIST_PLANE_VERTEX (see 243)
      ( CALC_COVARIANCE_MS (see 244)
' FPP_TRACK                245
      ( FPP_DRIFT_DIST    246
      ( TRK4              247
        ) TRACKIT        248
          * SETI          249
      ( FPP_ALIGN_CORR_FRONT 250
      ( FPP_ALIGN_CORR_REAR  251
      ( FPP_ALIGN_SUMS      252
      ( FPP_PLOT_TRACKS     253
        ) DEFINE_TRANSFORMATION (see 233)
        ) FRAME          (see 234)
        ) LABEL          (see 235)
' TOT_SHOWER              254
      ( DET_VERTEX        (see 226)
      ( SH_CLU_FIND       255
      ( SH_PRESORT        256
% REACTION_Z              (see 189)
% TRANSPORT_TO_BEAM      (see 192)
% DOT3                    (see 188)

```

% BEAM_VERTEX	257
& ELOSS_ELECTRON	258
& ELOSS_HADRON	259
& TRANSPORT_TO_BEAM	(see 192)
& CTRANS	(see 144)
& NEWANG	260
' ROTATE	261
( ROTATE3V	262
' CTRANS	(see 144)
' ROTATE	(see 261)
' CTRANS	(see 144)
' DERROR	(see 46)
& EXT_TAR_COR	263
' CORRECT_FP	264
( CALC_VAR_FP	265
( POW	(see 191)
( CALC_VAR_FP	(see 265)
( POW	(see 191)
( CALC_VAR_FP	(see 265)
' TRANSPORT_TO_BEAM	(see 192)
' BEAM_TO_TRANSPORT	266
( DERROR	(see 46)
' CORRECT_FP	(see 264)
' FP_VERTEX	(see 229)
' TG_VERTEX	(see 236)
& TARGET_EXIT	267
' FLAT_TARGET	268
' CIRCULAR_TARGET	269
' CYLINDRICAL_TARGET	270
& ELOSS_ELECTRON	(see 258)
& TARGET_EXIT	(see 267)
& ELOSS_ELECTRON	(see 258)
& TARGET_EXIT	(see 267)
& ELOSS_HADRON	(see 259)
% CALC_KINEMATICS	271
& ELECTRON_CALC	272
' CONTRACT	273
' PMAG	274
' PUT4V	275
' CONSERVE	276
' CONTRACT	(see 273)
' PMAG	(see 274)

'	BOOST	277
'	PMAG	(see 274)
'	EXTANG4V	278
'	PMAG	(see 274)
'	CONTRACT	(see 273)
&	SECONDARY_CALC	279
'	PMAG	(see 274)
'	PUT4V	(see 275)
'	NEWANG	(see 260)
'	CONSERVE	(see 276)
'	EXTANG4V	(see 278)
'	NEWANG	(see 260)
'	PMAG	(see 274)
'	PUT4V	(see 275)
'	ROTATE4V	280
(	ROTATE3V	(see 262)
'	BOOST	(see 277)
'	EXTANG4V	(see 278)
'	PMAG	(see 274)
&	CORRECT_KINEMATICS	281
'	POLY	(see 143)
'	POLY_INV	(see 238)
'	DERROR	(see 46)
'	POLY	(see 143)
%	CORRECT_COINTIME	282
&	CALC_PATHDIFF	283
&	POW	(see 191)
&	CALC_PATHDIFF	(see 283)
%	REVSUB	284
%	SCANSUB	285
&	BOUNDS_I	286
&	BOUNDS_R	287
&	INSIDE_POLYGON	288
'	ANGLE_BETWEEN	289
&	STACK_CRAWL	290
'	LPOP	291
'	DINFO	(see 43)
'	DERROR	(see 46)
'	LPUSH	292
'	DINFO	(see 43)
'	DERROR	(see 46)
&	USERVAR	(see 29)

```

& DET_SUB          293
  ' DET_SUB_USER    294
& VARIABLE          (see 11)
& INDEX_RANGE      295
  ' INDEX_RANGE_USER 296
  ' DERROR          (see 46)
& DET_SUB          (see 293)
& BOUNDS_I         (see 286)
& USERVAR         (see 29)
& BOUNDS_R         (see 287)
& DET_SUB          (see 293)
& BOUNDS_I         (see 286)
& USERVAR         (see 29)
& INSIDE_POLYGON   (see 288)
& STACK_CRAWL      (see 290)
& SAVESPEC         297
& VARIABLE          (see 11)
& STACK_CRAWL      (see 290)
& FILTER_EVENT     (see 193)
& OPTIMIZE_EVENT   298
  ' IN_EM_WINDOW    299
  ' IN_SP_WINDOW    300
    ( STACK_CRAWL    (see 290)
  ' SPECTROMETER_RECONSTRUCT (see 195)
  ' DIST_DET_VERTEX (see 166)
  ' POLY_INV        (see 238)
  ' DERROR          (see 46)
  ' SPECTROMETER_RECONSTRUCT (see 195)
  ' IN_SP_WINDOW    (see 300)
  ' IN_EM_WINDOW    (see 299)
  ' POLY_INV        (see 238)
  ' DIST_DET_VERTEX (see 166)
& INDEX_RANGE      (see 295)
& DET_SUB          (see 293)
& BOUNDS_I         (see 286)
& SAVEDIM          301
& USERVAR         (see 29)
& MFILL            302
  ' MEXIST          (see 68)
  ' MIDENT          (see 70)
  ' MEXIST          (see 68)
  ' MERROR          (see 71)

```

```

        ' MIDENT          (see 70)
        ' MLOCK           303
            ( MIDENT      (see 70)
            ( MEXIST      (see 68)
            ( MERROR      (see 71)
            ( MIDENT      (see 70)
% SHCAL_EVENTS          304
% FILTER_EVENT          (see 193)
$ ACCUMULATE_RUNS      305
$ WRITEOUT_ALLSCALERS  306
% EXMATCH              (see 17)
% NOSPACE              (see 18)
% EXMATCH              (see 17)
% WRITEOUT_SCALER      307
    & EXMATCH          (see 17)
    & RCHFUN           (see 107)
    & EXMATCH          (see 17)
    & RCHFUN           (see 107)
    & EXMATCH          (see 17)
    & RCHFUN           (see 107)
    & EXMATCH          (see 17)
    & RCHFUN           (see 107)
$ ACCUMULATE_RUNS      (see 305)
$ WRITEOUT_ALLSCALERS  (see 306)
$ ACCUMULATE_RUNS      (see 305)
$ WRITEOUT_ALLSCALERS  (see 306)
$ ACCUMULATE_RUNS      (see 305)
$ WRITEOUT_ALLSCALERS  (see 306)
$ PROGRAM_TO_HEADER    308
$ DERROR               (see 46)
# GETERR               (see 61)
# OPTIMIZE_DO          309
    $ DERROR           (see 46)
    $ OPTIMIZE_TRANSPORT 310
        % POLY_INV      (see 238)
        % DERROR        (see 46)
        % FILENAME      311
            & NOSPACE    (see 18)
        % POLY_INV      (see 238)
        % DERROR        (see 46)
        % CALC_MATRIX    (see 159)
        % TRANSPORTFIT   312

```

& ALLOCATE_MEM	313
& DEALLOCATE_MEM	314
& DPKIN_ERROR	315
' POW	(see 191)
& PHTG_ERROR	316
' POW	(see 191)
& THTG_ERROR	317
' POW	(see 191)
& YTG_ERROR	(see 190)
& FCHISQ_TRANSP_INDV	318
& FTEST	319
' BETAI	320
( BETACF	321
) DERROR	(see 46)
( GAMMLN	322
) DERROR	(see 46)
( GAMMLN	(see 322)
( BETACF	(see 321)
& ALLOCATE_MEM	(see 313)
& NOSPAC	(see 18)
& DERROR	(see 46)
& DEALLOCATE_MEM	(see 314)
& CURFIT_TRANSP	323
' FCHISQ_TRANSP	324
' PNTR2	325
' ALLOCATE_MEM	(see 313)
' DEALLOCATE_MEM	(see 314)
' ALLOCATE_MEM	(see 313)
' DERROR	(see 46)
' FUNCTN_TRANSP1	326
( CALC_VAR_TG	(see 237)
( POW	(see 191)
( CALC_MATRIX	(see 159)
( POW	(see 191)
( CALC_VAR_TG	(see 237)
' FUNCTN_TRANSP2	327
( CALC_VAR_TG	(see 237)
( POW	(see 191)
( CALC_MATRIX	(see 159)
( POW	(see 191)
( CALC_VAR_TG	(see 237)
' FCHISQ_TRANSP	(see 324)

```

' FDERIV_TRANSP1          328
  ( CALC_VAR_TG           (see 237)
  ( FOC_DERIV             329
  ( POW                   (see 191)
  ( PNTR2                 (see 325)
  ( CALC_MATRIX           (see 159)
  ( POW                   (see 191)
  ( CALC_VAR_TG           (see 237)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
' FDERIV_TRANSP2          330
  ( CALC_VAR_TG           (see 237)
  ( FOC_DERIV             (see 329)
  ( POW                   (see 191)
  ( PNTR2                 (see 325)
  ( CALC_MATRIX           (see 159)
  ( POW                   (see 191)
  ( CALC_VAR_TG           (see 237)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
  ( FOC_DERIV             (see 329)
  ( PNTR2                 (see 325)
' PNTR2                   (see 325)
' MATINV                   (see 208)
' PROTECT_TRANSP          331
' FUNCTN_TRANSP1         (see 326)

```

```

      ' FUNCTN_TRANSP2      (see 327)
      ' FCHISQ_TRANSP      (see 324)
      ' DEALLOCATE_MEM      (see 314)
& DERROR      (see 46)
& DEALLOCATE_MEM      (see 314)
& DPKIN_ERROR      (see 315)
& THTG_ERROR      (see 317)
& YTG_ERROR      (see 190)
& PHTG_ERROR      (see 316)
& DPKIN_ERROR      (see 315)
& THTG_ERROR      (see 317)
& YTG_ERROR      (see 190)
& PHTG_ERROR      (see 316)
& CURFIT_TRANSP      (see 323)
& FTEST      (see 319)
& CURFIT_TRANSP      (see 323)
& FCHISQ_TRANSP_INDV      (see 318)
& DEALLOCATE_MEM      (see 314)
% DERROR      (see 46)
% POLFIT      (see 220)
% LINSOLV      332
  & DERROR      (see 46)
  & LUDCMP      (see 209)
  & LUBKSB      (see 210)
% TRANSPORTFIT      (see 312)
% WRITE_DATABASE      333
  & DERROR      (see 46)
$ OPTIMIZE_COINTIME      334
  % DERROR      (see 46)
  % FILENAME      (see 311)
  % DERROR      (see 46)
  % PATHLENGTH_FIT      335
    & TC_ERROR      336
      ' POW      (see 191)
    & ALLOCATE_MEM      (see 313)
    & DEALLOCATE_MEM      (see 314)
    & ALLOCATE_MEM      (see 313)
    & NOSPAC      (see 18)
    & DERROR      (see 46)
    & DEALLOCATE_MEM      (see 314)
    & DERROR      (see 46)
    & DEALLOCATE_MEM      (see 314)

```

```

& TC_ERROR          (see 336)
& CURFIT_PATH      337
  ' FCHISQ_PATH      338
  ' PNTR2            (see 325)
  ' ALLOCATE_MEM     (see 313)
  ' DEALLOCATE_MEM   (see 314)
  ' ALLOCATE_MEM     (see 313)
  ' FUNCTN_PATH      339
    ( CALC_PATHDIFF  (see 283)
    ( POW            (see 191)
    ( CALC_PATHDIFF  (see 283)
    ( POW            (see 191)
    ( CALC_PATHDIFF  (see 283)
  ' FCHISQ_PATH      (see 338)
  ' FDERIV_PATH      340
    ( POW            (see 191)
    ( PNTR2          (see 325)
    ( POW            (see 191)
    ( PNTR2          (see 325)
  ' PNTR2            (see 325)
  ' MATINV           (see 208)
  ' PROTECT_PATH     341
  ' FUNCTN_PATH      (see 339)
  ' FCHISQ_PATH      (see 338)
  ' DEALLOCATE_MEM   (see 314)
& DERROR           (see 46)
& DEALLOCATE_MEM   (see 314)
% WRITE_DATABASE   (see 333)
$ OPTIMIZE_EM      342
% FILENAME         (see 311)
% DERROR           (see 46)
% EM_FIT           343
  & POLY            (see 143)
  & ALLOCATE_MEM    (see 313)
  & DEALLOCATE_MEM  (see 314)
  & ALLOCATE_MEM    (see 313)
  & NOSPAC          (see 18)
  & DERROR          (see 46)
  & DEALLOCATE_MEM  (see 314)
& CURFIT_EM        344
  ' FCHISQ_EM       345
  ' PNTR2           (see 325)

```

```

      ' ALLOCATE_MEM          (see 313)
      ' DEALLOCATE_MEM       (see 314)
      ' ALLOCATE_MEM          (see 313)
      ' FUNCTN_EM            346
        ( POLY                (see 143)
      ' FCHISQ_EM            (see 345)
      ' FDERIV_EM            347
        ( POLY                (see 143)
        ( PNTR2               (see 325)
        ( POLY                (see 143)
        ( PNTR2               (see 325)
        ( POLY                (see 143)
        ( PNTR2               (see 325)
        ( POLY                (see 143)
        ( PNTR2               (see 325)
      ' PNTR2                (see 325)
      ' MATINV               (see 208)
      ' PROTECT_EM           348
      ' FUNCTN_EM            (see 346)
      ' FCHISQ_EM            (see 345)
      ' DEALLOCATE_MEM       (see 314)
& DERROR                   (see 46)
& DEALLOCATE_MEM           (see 314)
& POLY                     (see 143)
& DEALLOCATE_MEM           (see 314)
% WRITE_DATABASE           (see 333)
$ OPTIMIZE_HEAP            349
% FILENAME                 (see 311)
% DERROR                   (see 46)
% HEEP_FIT                 350
  & POLY                   (see 143)
  & ALLOCATE_MEM           (see 313)
  & DEALLOCATE_MEM        (see 314)
  & ALLOCATE_MEM           (see 313)
  & NOSPAC                 (see 18)
  & DERROR                 (see 46)
  & DEALLOCATE_MEM        (see 314)
  & CURFIT_HEAP            351
    ' FCHISQ_HEAP          352
    ' FUNCTN_HEAP          353
      ( POLY                (see 143)
    ' WEIGHT_HEAP          354

```

```

        ( POLY          (see 143)
        ( DERROR        (see 46)
        ( POLY          (see 143)
        ( MATINV        (see 208)
    ' FCHISQ_HEAP      (see 352)
    ' FDERIV_HEAP      355
        ( POLY          (see 143)
    ' WEIGHT_HEAP      (see 354)
    ' MATINV           (see 208)
    ' PROTECT_HEAP     356
    ' FUNCTN_HEAP      (see 353)
    ' WEIGHT_HEAP      (see 354)
    ' FCHISQ_HEAP      (see 352)
    & DERROR           (see 46)
    & DEALLOCATE_MEM   (see 314)
    & POLY             (see 143)
    & DEALLOCATE_MEM   (see 314)
    % WRITE_DATABASE   (see 333)
$ OPTIMIZE_ADC_SC     357
    % DERROR           (see 46)
    % NOSPAC           (see 18)
    % WRITE_DATABASE   (see 333)
$ OPTIMIZE_BETA       358
    % ALLOCATE_MEM     (see 313)
    % DEALLOCATE_MEM   (see 314)
    % PNTR2            (see 325)
    % DERROR           (see 46)
    % ALLOCATE_MEM     (see 313)
    % NOSPAC           (see 18)
    % POLFIT           (see 220)
    % PNTR2            (see 325)
    % FILENAME         (see 311)
    % CURFIT_BETA      359
        & BOUNDARY_BETA 360
        & FCHISQ        (see 207)
        & PNTR2         (see 325)
        & ALLOCATE_MEM   (see 313)
        & DEALLOCATE_MEM (see 314)
        & ALLOCATE_MEM   (see 313)
        & DERROR        (see 46)
        & FUNCTN_BETA    361
            ' POLY      (see 143)

```

```

& FCHISQ          (see 207)
& FDERIV_BETA     362
    ' PNTR2        (see 325)
& PNTR2           (see 325)
& BOUNDARY_BETA   (see 360)
& PNTR2           (see 325)
& MATINV          (see 208)
& PROTECT_BETA    363
& FUNCTN_BETA     (see 361)
& FCHISQ          (see 207)
& DEALLOCATE_MEM  (see 314)
% DERROR          (see 46)
% DEALLOCATE_MEM  (see 314)
% WRITE_DATABASE  (see 333)
$ OPTIMIZE_VDC_T0 364
    % NOSPACE      (see 18)
    % DERROR       (see 46)
    % WRITE_DATABASE (see 333)
$ OPTIMIZE_VDC_DRIFT 365
    % NOSPACE      (see 18)
    % WRITE_DATABASE (see 333)
$ OPTIMIZE_VDC_TX 366
    % ALLOCATE_MEM (see 313)
    % DEALLOCATE_MEM (see 314)
    % ALLOCATE_MEM (see 313)
    % OPEN_FILE     (see 63)
    % NOSPACE       (see 18)
    % ALLOCATE_MEM (see 313)
    % DEALLOCATE_MEM (see 314)
    % CLOSE_FILE    (see 64)
$ OPTIMIZE_ADC_CH 367
    % NOSPACE       (see 18)
    % WRITE_DATABASE (see 333)
$ DERROR          (see 46)
# DINFO           (see 43)
# SHCAL_DOFIT     368
    $ SHCAL_COEF_IO (see 155)
    $ READ_SHOW_PEDS 369
    $ FUMILI        370
        % SGZ        371
            & ARITH1    372
            & ARITH2    373

```

```

        & ARITH3                374
        % MCONV                  375
        % MONITO                  376
        $ SHCAL_COEF_IO          (see 155)
        $ READ_SHOW_COEF        (see 154)
# WREV                          377
# GETERR                        (see 61)
# NOSPACE                       (see 18)
# WRITE_CUT_SUM                 378
    $ NOSPACE                    (see 18)
# CONDENSE                      (see 23)
# FF_ELALLFUN                   (see 82)
# DERROR                        (see 46)
# INPUT                          (see 114)
# GETERR                        (see 61)
# CONDENSE                      (see 23)
# FILTER_IT                     379
    $ GETERR                     (see 61)
    $ READ_LOGIC                 (see 73)
    $ GETERR                     (see 61)
    $ ASSIGN_FILTER              380
# GETERR                        (see 61)
# NOSPACE                       (see 18)
# WRITE_CUT_SUM                 (see 378)
# DERROR                        (see 46)
" DEF_CALIBRATION              381
# GETERR                        (see 61)
# CONDENSE                      (see 23)
# DERROR                        (see 46)
# CONDENSE                      (see 23)
# OPTIMIZE_IT                   382
    $ GETERR                     (see 61)
    $ READ_LOGIC                 (see 73)
    $ GETERR                     (see 61)
    $ ASSIGN_OPTIMIZE            383
        % DERROR                 (see 46)
        % CODE_VAR               (see 21)
        % DERROR                 (see 46)
        % OPTIMIZE_INI           384
            & DERROR              (see 46)
            & READFIT             385
                ' NR_ITEMS        (see 7)

```

```

        ' GETERR          (see 61)
        ' SHORTSTRING    (see 65)
        ' DERROR         (see 46)
        ' NR_ITEMS       (see 7)
        ' READ_LOGIC     (see 73)
        ' GETERR         (see 61)
        ' DERROR         (see 46)
        ' NR_ITEMS       (see 7)
        & NOSPAC         (see 18)
        & OPEN_FILE      (see 63)
    % DERROR            (see 46)
# GETERR              (see 61)
" PLOTIT              386
" USER_EXIT          387
# EXVERIFY            388
    $ NOSPAC          (see 18)
# CLOSE_FILE          (see 64)
# WRITEOUT_ALLSCALERS (see 306)
# SCALER_HISTORY      389
    $ NOSPAC          (see 18)
# OPTIMIZE_END        390
    $ NOSPAC          (see 18)
    $ CLOSE_FILE      (see 64)
    $ NOSPAC          (see 18)
# FPP_LAST            391
    $ DT              392
# FPP_PULS_OUT        393
# TOD                  394
# DINFO               (see 43)

```

# Appendix F

## COOLHANDS Programming Guide

### F.1 Concept

The COOLHANDS[12] system is designed to attack the problem of scanning and histogramming event-mode data in a flexible and general way. The program assumes only that the data contain events in which one can define

1. the number of words in the event,
2. the detector firing sequence (or hitbit pattern),
3. a set of variables which the user may wish to histogram and make cuts upon.

If the data can be represented in these general terms, then it can be scanned with COOLHANDS by attaching a few simple subroutines. The program is designed so that a user-programmer needs to know very little about the bulk of the program to interface a particular system to it.

### F.2 FORTRAN Components

The executable scanning program is formed by linking several object files together. Some of the components have been made into libraries. The following shows the makefile which is used to obtain the executable of a program called analyzer.

```
LOC_HALLA_LIB = /work/halla/com96/offermand/halla/lib/SunOS
LDLFLAGS      = -L$(LOC_HALLA_LIB)
LIBS          = $(LOC_HALLA_LIB)/coolhands.o -lcool default -lfcalc
                                   -lmbook4d -lutil -lvms
FC            = f77
LINKER        = f77
MAKEFILE      = Makefile
OBJS          = analyzer.f
```

```

PROGRAM      = analyzer
SRCS        = analyzer.f
all:        $(PROGRAM)
$(PROGRAM): $(OBJS)
            $(LINKER) $(LDFLAGS) $(OBJS) $(LIBS) -o $(PROGRAM)

```

The source file `analyzer.f` should contain the main program and a group of subroutines describing the processing of an event. Further on in this document is described the contents of the `analyzer.f` file. If the user is not interested in defining some of the routines one has to include the library `cool_default.f` at the end of the library list. It contains a default set of user subroutines and by including it in the linking process, routines not found in the user files, will be picked up from here.

The object file `coolhands.o` contains the important routines for scanning, cutting and histogramming the event stream. The user-supplied file `analyzer.f` contains all routines which are specific to the data acquisition of the event stream. Included here are an input subroutine to read data and interpret the event format and a routine to define and return variables for scanning and histogramming. These routines are described later in this document.

The library of default routines `COOL_DEFAULT` exists so that options may be added which the user need not know about. Many of the subroutines which may be included in the user set are not essential and the default routines may suffice. Since object libraries such as this only look for routines which have not been found in the linking process, the user may supercede any or all of these routines in the user set.

The histogram manipulation library `mbook4d` provides an interface to the CERN `HBOOK` package for initializing, defining, filling, and storing histograms. The `util` library contains very general routines which are used in many programs. Most of the routines in this library are all simple ‘independent’ routines which do not require routines from other libraries.

### F.3 FORTRAN Unit Numbers and Common Blocks

The FORTRAN components of COOLHANDS, excluding the user-written subroutines (e.g., `analyzer.f`), use the following FORTRAN unit numbers:

5 standard program input (STDIN)

6 standard program output (STDOUT)

29 graphical cut info are being used

The following named common blocks are used in the COOLHANDS system and should not be redefined by the user.

common	file
cool_i cool_r cool_c cool_l cool_rec	coolhands/coolhands.h
defvars	coolhands/defspec.h
mbook_i mbook_l mbook_s mbook_comm	mbook4d/mbook.h
menu	util/util.h
fcalc1 fcalcpass	fcalc/fcalc.h

The problem of duplicating subroutine names should be taken care of by the compiler.

## F.4 User-Supplied Subroutines

The coolhands user-programmer must supply a set of routines. All but the first two of these listed below may be included by linking to the default set `cool_default.f`.

- input
- variable
- det\_sub.f
- filter\_close
- filter\_event
- filter\_open
- index\_range.f
- insert\_header
- optimize\_do
- optimize\_event
- optimize\_ini
- option\_do
- option\_setup

- `user_exit`
- `user_init`

The `input` routine reads the event format for a file name which is passed. It is called after a `scan` command. This routine should read a block of data and, for each event, call one or both of `scansub` and `revsub`. `scansub` makes cuts and increment histograms, whereas `revsub` acquires statistics on the data. The routines `scansub` and `revsub`, which are found in the main set of COOLHANDS routines, call the second required user routine `variable` to get information for a given spectrum from each event. This routine should communicate with the `input` routine via user-defined common blocks. The routine `user_init` is called upon program entry and after the command `initialize`. Its function is to initialize the histogram package or to begin a CPU clock or define names needed for the option command to work. The routine `user_exit` is called only after the `exit` command and may be used by the user for miscellaneous purposes. For instance, one might stop the CPU clock and print the elapsed time. After the return from this routine to the main COOLHANDS program, a FORTRAN stop is encountered. The `filter_event` and `filter_open` routines allow the user to accomplish filtering.

What follows now are examples of the routines the user has to provide to COOLHANDS for some fictitious data stream.

#### F.4.1 `input`

```
SUBROUTINE INPUT(SCANFILE,TEST,SCAN,REVIEW,LOOP,FIRST,LAST)
CHARACTER SCANFILE*(*)
LOGICAL TEST,SCAN,REVIEW,LOOP
INTEGER FIRST,LAST
```

The function of this routine is to read the data, interpret the event format, and call the proper COOLHANDS routines to perform the action desired. `SCANFILE` is name of the data file to be scanned. `TEST` is a logical telling the routine that it is to simply open the file to test for its existence and output an error message if necessary before closing the file and returning. `SCAN` tells whether or not to call a COOLHANDS routine for each event to scan the file, make cuts, and accumulate histograms. `REVIEW` tells whether to call the proper COOLHANDS routine for reviewing each file to accumulate statistics. See the reference chapter for explanations of these functions.

The routine should be organized to first open the file, then to return if `test` is `.TRUE.` or otherwise to read in a block of data. This data should then enter a loop in which each event is interpreted and placed into a common block (e.g., `common /event/`) which is shared with the user-supplied function `variable` described below. This common block should contain the number of words in the event, the hitbit pattern, and a list of variables required by the function variable. The hitbit pattern is a 32 bit integer in which each bit represents a detector and is set if that detector fired. An example of such a common is shown in the example routine below.

After an event has been decoded into the common block within this loop, the user should call the appropriate COOLHANDS routines:

```
IF (SCAN) CALL SCANSUB
IF (REVIEW) CALL REVSUB
```

An example of an input subroutine is shown below.

```
SUBROUTINE INPUT(SCANFILE,TEST,SCAN,REVIEW,LOOP,FIRST,LAST)

IMPLICIT NONE

CHARACTER SCANFILE*(*)
LOGICAL TEST,SCAN,REVIEW,LOOP
INTEGER FIRST,LAST

C This routine reads and interprets the event buffer from interrupt
C FIRST upto LAST and sends events to
C subroutines SCANSUB or REVSUB to either execute desired cuts in spectra
C or get statistics on the data, respectively.
C   SCANFILE..... the name of the event file to be scanned
C   TEST ..... a logical which, if true, tells the routine to
C               simply try to open the file, close it, and return
C   SCAN ..... a logical which, if true, tells the routine to
C               call for each event the routine SCANSUB
C   REVIEW ..... a logical which, if true, tells the routine to
C               call for each event the routine REVSUB
C   LOOP ..... a logical indicating whether the routine INPUT
C               is being called in a loop and therefore the input
C               file should not be opened at each entry.
C   FIRST ..... first interrupt to be analyzed
C   LAST ..... last interrupt to be analyzed

INTEGER NRECORD
PARAMETER(NRECORD=500)
```

C The data will be read into the following array

```
INTEGER IBLOCK(NRECORD)
```

C To make the example simple, suppose we have fixed records of 500 I\*4  
C words per record and events with a constant length of 100 words. The  
C event format will be (1) the number of words in the event, (2) the event  
C type, (3) the event hitbit pattern, (4) 32 adc words, each corresponding

C to a bit in the hitbit pattern, (5) 32 TDC words, each corresponding to  
 C a bit in the hitbit pattern, (6) 33 electron words, where the first electron  
 C word is the on-line calculated fine momentum and other electron words may  
 C be drift times, ...etc. Each event will be placed into the following  
 C common block to be passed to function VARIABLE

```

    INTEGER NWIE,TYPE,HITBIT,ADC(0:31),TDC(0:31),ELE(0:31),FINE_MOM
    COMMON/EVENT/NWIE,TYPE,HITBIT,ADC,TDC,ELE,FINE_MOM
  
```

C miscellaneous declaration

```

    INTEGER IWORD,JEVENT,J,NR_EVENT
  
```

C open the event file to be scanned

```

    OPEN(1,FILE=SCANFILE,FORM='UNFORMATTED',STATUS='OLD',READONLY,
    &ERR=1000)
  
```

C if the "test" command is in effect, then it is enough to open the file and  
 C make sure it is there; in this case, close and return.

```

    IF (TEST) THEN
      CLOSE(1)
      RETURN
    ENDF
  
```

```

    NR_EVENT=0
  
```

C read in an event record

```

10  READ(1,END=100,ERR=2000) (IBLOCK(J), J=1,NRECORD)
    IWORD=0
  
```

C decode the 5 events in the record and "scan" or "review" each event

```

    DO JEVENT=1,NRECORD/100
      NR_EVENT=NR_EVENT+1
      IF (NR_EVENT.GE.FIRST) THEN
        NWIE=IBLOCK(IWORD+1)
        TYPE=IBLOCK(IWORD+2)
        HITBIT=IBLOCK(IWORD+3)
        DO JDETECTOR=0,31
          ADC(JDETECTOR)=IBLOCK(IWORD+JDETECTOR+4)
          TDC(JDETECTOR)=IBLOCK(IWORD+JDETECTOR+36)
          ELE(JDETECTOR)=IBLOCK(IWORD+JDETECTOR+69)
        ENDDO
        FINE_MOM=IBLOCK(IWORD+68)
      
```

C The event is decoded, now scan or review:

```

        IF (SCAN) CALL SCANSUB
        IF (REVIEW) CALL REVSUB
      
```

C move onto the next event in the record

```

        IWORD=IWORD+NWIE
      
```

```

        ENDIF
    ENDDO
C now read another record
    IF (NR_EVENT.LT.LAST) GOTO 10
C
100  RETURN

C the following calls are to a routine in the MLIB library.
1000 CALL DERROR(' Error opening the event file. ')
    RETURN
2000 CALL DERROR(' Error reading the event file. ')
    END

```

### F.4.2 variable

```

REAL FUNCTION VARIABLE(IVAR, IDET, INDX, NWORDS, EVENT_HBIT, SETBIT, WEIGHT)
INTEGER IVAR, IDET, INDX, NWORDS
LOGICAL*1 EVENT_HBIT(*)
LOGICAL SETBIT
REAL WEIGHT

```

This routine is the function which returns event variables.

`ivar` specifies the variable type to return by indicating the index in character array `goodsp` below. The variable type `adc`, for example, would be named in `goodsp` and assigned within the routine as `variable=adc(idet)`, where the `adc` array is contained in the event common block. There are two special cases. If `ivar` is 0, `intvar` should return the number of words in the event. If `ivar` is less than 0, `intvar` should return the hitbit pattern.

`idet` is the detector number.

`intvar` returns the integer-variables which specify the hit pattern and the number of words in the event.

`setbit` is an output logical which tells the calling routine if the bit `idet` was set in the hit pattern for the event. Additionally, special cuts may be accomplished by using `setbit`.

`weight` is the weight to give the event in a histogram.

The variables returned are defined in the common block `defspec`, which is the only common block shared with the main program `COOLHANDS`. This common block is contained in the include file `defspec.h`.

```

structure /var_type/
  character*12 name
  integer intbin
  logical array
  logical subdet
  logical*1 all_det(32),all_det_opt(32)
  logical*1 enf_det(32)
  real range(2)
end structure

structure /det_type/
  character*80 string
  integer number
end structure

record /det_type/ defdet(32)
record /var_type/ defvar(10)
character var_txt(10)*80,det_txt(32)*80

common /defvars/ defdet,defvar,var_txt,det_txt

```

`goodsp` defines the names of the spectra as they will be identified in `input` by the user.

`iadet` is an array of integers which specifies allowed detectors for each spectrum; e.g., if detector 7 is allowed to be associated with spectrum 4, then bit 7 of `iadet(4)` will be set.

`iadeto`pt is an array of integers which specifies allowed detectors for each spectrum which can be optimized.

`range` provides low and high limits for each spectrum. The spectrum range is used only by the `review` facility and as a window for the spectrum when `auto-window` is `.TRUE.`. Out-of- bounds variables are not corrected to the nearest window.

`intbin` is an array which types each defined spectrum as an integer or a real one. If the  $i^{\text{th}}$  element of `intbin` is 1, the corresponding spectrum will be declared to be an integer one and thus integer-binning will be done. Otherwise, real-binning will be used. If the spectrum is set up as an integer one, the values will still be returned in the real variable but should be truncated to an integer number. See the COOLHANDS user manual[12] for a description of each type of binning.

The variable `ivar` in the calling sequence corresponds to the index of dimension `NSTYPES` for each of these arrays. Sample data statements to define the spectra are:

```

DATA GOODSP /'adc','tdc','fm','mom',26*' '/
DATA IADET / -2 , -2 , 1 , 1, 26*0 /
DATA IADETOPT / 30*0 /
DATA INTBIN /1,1,1,0,26*0/
DATA RANGE / 0.,2047. , 0.,2047. , 0.,2047. , 0.,5. , 52*0. /

```

The above data statements define 4 variables:

`adc` is an integer variable within an allowed range of 0 to 2047; it may correspond to detectors 1-31 but not to detector 0

`tdc` is an integer variable within an allowed range of 0 to 2047; it may correspond to detectors 1-31 but not to detector 0

`fm` is an integer variable within an allowed range of 0 to 2047; it may correspond to detector 0 but not to detectors 1-31

`mom` is a real variable with an allowed range of 0. to 5.; it may correspond only to detector 0

An example of the function is

```

REAL FUNCTION VARIABLE(IVAR,IDET,INDX,NWORDS,EVENT_HBIT,
>                      SETBIT,WEIGHT)

```

C input: `ivar,idet`  
c output: `variable,intvar,setbit,weight`

```

IMPLICIT NONE

```

```

INTEGER IVAR,IDET,INDX,NWORDS
LOGICAL*1 EVENT_HBIT(*)
LOGICAL SETBIT
REAL WEIGHT

```

C include the common block `/defspec/`

```

INCLUDE 'DEFSPEC.H'

```

C the following common block should be shared with the input routine

```

INTEGER NWIE,TYPE,HITBIT,ADC(0:31),TDC(0:31),ELE(0:31),FINE_MOM
COMMON /EVENT/ NWIE,TYPE,HITBIT,ADC,TDC,ELE,FINE_MOM

```

C miscellaneous declarations

```

REAL UNBIN,CALC_MOM
INTEGER ISEED
DATA ISEED/536543/

```

C define the spectra

```

DATA GOODSP /'Adc','Tdc','Fm','Mom',16*' '/
DATA IADET / -2 , -2 , 1 , 1, 16*0 /
DATA IADETOPT / 20*0 /
DATA INTBIN /1,1,1,0,16*0/
DATA RANGE / 0.,2047. , 0.,2047. , 0.,2047. , 0.,5. , 32*0. /
C initialize returned values
WEIGHT = 1.0
VARIABLE = 0.
INTVAR = 0.
C did the detector IDET fire for this event?
SETBIT = BTEST(HITBIT,IDET)
C return the appropriate variable for each spectrum type
IF (IVAR.EQ.0) THEN
  INTVAR = NWIE
ELSE IF (IVAR.LT.0) THEN
  INTVAR = HITBIT
ELSE IF (.NOT.SETBIT) THEN
C
C Detector did not fire.
C
  ELSE
C
C Detector fired.
C
  IF (IVAR.EQ.1) THEN
    VARIABLE = ADC(IDET)
  ELSE IF (IVAR.EQ.2) THEN
    VARIABLE = TDC(IDET)
  ELSE IF (IVAR.EQ.3) THEN
    VARIABLE = FINE_MOM
  ELSE IF (IVAR.EQ.4) THEN
C the variable unbin spans one channel and simply "unbins" data to
C prevent problems associated with binning discrete data with bin sizes
C not the same as the original binning done by the tdc's and adc's.
C Note that a second course of action could have been taken here. The
C variable "fine_mom" could have set up as an integer variable by
C multiplying the actual calculated momentum by a number sufficiently
C large to gain the desired precision and then truncating it to an
C integer value.
    UNBIN = 0.5*( 1 -2.*RAN(ISEED))
    FINE_MOM = FINE_MOM+UNBIN
C the routine "calc_mom" below should be a routine to take the fine

```

```

C momentum and spectrometer field and generate the electron momentum
      VARIABLE = CALC_MOM(FINE_MOM)
      ELSE
        STOP 'Illegal spectrum type in module scan'
      ENDIF
    ENDIF
  C
    RETURN
  END

```

### F.4.3 user\_init

USER\_INIT is called upon program entry and simply defines default values for some program variables upon entry. Other functions (like beginning the CPU clock or defining names for the option command) may be included here. The default routine which is provided in the COOL\_DEFAULT library is shown below.

```

      SUBROUTINE USER_INIT(AWIND,NDIM,W1,W2,NBIN,DO_NTUPLE)
c
c-----
c
      IMPLICIT NONE

c all passed variables are output
      LOGICAL AWIND,DO_NTUPLE
      INTEGER NDIM,NBIN(*)
      REAL W1(*),W2(*)
      LOGICAL first
      INTEGER jdim

      SAVE FIRST

      DATA FIRST /.true./

c for the 1st time through...
      IF (FIRST) THEN
c call a routine to begin accumulating cpu and real time (this routine is
c in MENU library)
        CALL CPUBGN
c set up the names for the option command
        CALL OPTION_SETUP
        FIRST = .FALSE.
      ENDIF

```

```

c assign default values for windows, \# bins, and ascii logical
  AWIND = .FALSE. ! autowindow
  DO JDIM = 1,NDIM
    W1(JDIM) = 0. ! lower window
    W2(JDIM) = 2048. ! upper window
    NBIN(JDIM) = 50 ! number of bins
  ENDDO
  DO_NTUPLE = .FALSE. ! multi-dim histograms, no ntuple format
  RETURN
  END

```

#### F.4.4 user\_exit

USER\_EXIT is called upon a quit command and may be used for any purpose. The default routine which is provided in the COOL\_DEFAULT library is shown below.

```

      SUBROUTINE USER_EXIT

      IMPLICIT NONE

      REAL TIMCPU,TIMCLK
      CHARACTER TIME*8,DAY*8,TEXT*80
C find the time of day and stop the cpu clock (these routines are in MLIB)
      CALL TOD(TIME,DAY)
      CALL CPUEND(TIMCPU,TIMCLK)
C
      WRITE(TEXT,100) TIMCPU/60.,TIMCLK/60.
      WRITE(6,110) TIME,DAY,TEXT
C
100  FORMAT(' CPU time (min):',F6.2,' Real time (min):',F6.2)
110  FORMAT(' Ending time: ',A,' Ending day: ',A,/A)
C
      RETURN
      END

```

#### F.4.5 filter\_open, filter\_event and filter\_close

If one wishes to filter data based upon certain cuts, these routines can accomplish that. `filter_open` is called to allow the user to open the filter file. `filter_event` is called each time an event has passed the specified cuts for that filter. Multiple filters may be accomplished simultaneously. The default routines supplied in the COOL\_DEFAULT library are shown below.

```

SUBROUTINE FILTER_EVENT(IUNIT)

IMPLICIT NONE

INTEGER IUNIT
CHARACTER*(*) FNAME

C this entry is called only when the event has passed the conditions
C placed upon the filter specified by IUNIT
RETURN

C
ENTRY FILTER_OPEN(IUNIT,FNAME)
C this entry opens filter files
RETURN

C
ENTRY FILTER_CLOSE(IUNIT)
C this entry closes filter files
RETURN
END

```

#### F.4.6 option\_do

If the user wished to provide himself with options given during normal COOLHANDS input, he may supply this routine to accomplish that. Calls to option define (a COOLHANDS library routine) define the option names (one per call), and this routine executes them. The default routine supplied in COOL\_DEFAULT library is shown below.

```

SUBROUTINE OPTION_DO(IOPTION, ARGUMENT)

IMPLICIT NONE

INTEGER NOPTIONS, IOPTION
PARAMETER (NOPTIONS=20)
CHARACTER OPTION(NOPTIONS)*80, ARGUMENT*(*)
INTEGER*4 JO

DATA OPTION/20*' '/

C IOPTION gives the option number, and ARGUMENT is its argument
RETURN

C
ENTRY OPTION_SETUP
C This entry should be called from USER_INIT
C DO JO = 1,NOPTIONS

```

```
C CALL OPTION_DEFINE(OPTION(JO))  
C ENDDO  
C  
    RETURN  
    END
```

# Bibliography

- [1] KUIP, CERN Program Library Long Writeup I102, version 2.05; Manual can be obtained at the anonymous ftp-site asis01.cern.ch in the directory doc/cernlib.
- [2] PAW, CERN Program Library Long Writeup Q121; Manual can be obtained at the anonymous ftp-site asis01.cern.ch in the directory doc/cernlib.
- [3] J.B. Mandeville, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1993.
- [4] G. Heyes *et al.*, www-site <http://alcor.ceba.gov/coda> .
- [5] E.A.J.M. Offermann, White paper for calibrating the High Resolution Spectrometers in hall A, TJNAF (1995)
- [6] Backtracing particle rays through magnetic spectrometers: avoiding systematic errors in the reconstruction of target coordinates, Th. Veit, J. Friedrich and E.A.J.M. Offermann, Nucl. Instr. and Meth. **A 336** (1993) 572.
- [7] A. Ketikyan, H. Voskanyan, and B. Wojtsekhovski, *About Shower Detector Software*, Sept. 1997, [http://www.jlab.org/Hall-A/data\\_reduc/calibrations/shower.ps.gz](http://www.jlab.org/Hall-A/data_reduc/calibrations/shower.ps.gz)
- [8] K.L. Brown, D.C. Carey, C. Iselin, and F. Rothacker, TRANSPORT: A Computer Program for Designing Charged Particle Beam Transport Systems, Stanford Linear Accelerator Center, 1991, SLAC-91, NAL-91, and CERN-73-16.
- [9]
- [10]
- [11] C.Hyde-Wright, L. Todor, and G.Laveissiere, *Beam Position Studies for E93050*, <http://www.jlab.org/luminita/BPM2.ps>
- [12] COOLHANDS programming guide, J. Mandeville and E.A.F.M. Offermann