TRIUMF Experiment E614 Data Analysis Code
MOFIA 1.0 and 1.5

Maher Quraan
University of Alberta/TRIUMF
February 16, 1999

Contents

1 Introduction

The framework of MOFIA has been adapted from KOFIA, the software package developed for BNL experiment E787. The present version, **MOFIA 1.5**, includes only a "skeleton" that would allow the user to access the Monte Carlo data for purposes of developing the software code for E614. The previous version **MOFIA 1.0** contains the code used for analysing the test chamber data. Since most users are interested in MOFIA 1.5, the focus of this document will be on that version. A section is attached at the end to describe briefly MOFIA 1.0 for those who are interested in pursuing the task of the drift chamber calibrations and/or cross talk studies.

2 Libraries Used by MOFIA

The following libraries are used by MOFIA 1.0 and MOFIA 1.5:

Library	MOFIA 1.0	MOFIA 1.5
cfm	2.2	2.3
io	2.0	-
fiowa	2.0	2.0
bru	1.1	1.1
gplot	1.1	1.1
ybos	1.2	4.0
cernlib	v95a	v97a_mips2

3 MOFIA 1.5 Contents and Description

MOFIA 1.5 includes:

1. Geometry description of the E614 detector read in from an ascii data file common to both the Monte Carlo and the data analysis codes. The current geometry input file name is *dt_geo*.00010, where the extension number specifies the version number (to be associated with run numbers). The geometry input file will be managed through Calibration File Management CFM, where the association between version numbers and run numbers is made. This will allow us to keep track of any geometry changes of the E614

detector. The file is read in file *geometry.F* and the results are saved in common block *dectect_geom.cmn*, which is also common to both the Monte Carlo and the analysis codes, as is the geometry parameter file *detect_geom.par*.

2. Setting up the Drift Chamber (DC) by calculating useful quantities from the geometry data and calculating the positions of the DC wires. These calculations are performed in *setup_det.F*. The results are saved in *dc.cmn*, and corresponding parameters are defined in *dc.par*.

3. Reading the calibration input files (plane position corrections, plane rotation corrections, wire position corrections and chamber resolution parameters). All the correction files contain nil values at this point and have only been implemented to allow the inclusion of corrections in setting up the detector. These files are managed through CFM and carry extension numbers similar to the geometry input file to allow association with run numbers. The calibration files are read in $read_calib.F$ and the variables are saved in *calibrations.cmn*.

4. Definition of detector subsystems, where four subsystems have been defined: drift chamber (DC), proportional chamber (PC), scintillators (SC), muon chamber (MC). Each one of these subsystems have three flags associated with it: **decode**, **track** and **cuts** for purposes of turning on/off decoding, tracking and cuts, respectively.

5. Implementation of namelists that contain flags, cuts and settings. These namelists will be expanded in the future as needed. Some of the contents of these namelists are not used in this version of MOFIA, and have only been defined for future use. Some of the useful namelists implemented for E614 in MOFIA 1.5 are:

Cuts:

DCCUTS	Drift Chamber CUTS
PCCUTS	Proportional Chamber CUTS - not installed yet
SCCUTS	SCintillator CUTS
MCCUTS	Muon Chamber CUTS - not installed yet

Settings:

DCSET	Drift Chamber SETtings
SCSET	SCintillator SETtings

Flags:

DCCFLAGS	Drift Chamber calibration FLAGS

Histogramming:

HIST	HISTogramming parameters

The contents of all these namelists are saved in common blocks, where the filename of the common is the same as the namelist title preceded by *nl* (for example *nldcset.cmn*, corresponding to the namelist $DCSET$). A description of how to access these namelists and/or edit the values of their contents is described below.

4 MOFIA 1.5 Initialization and Analysis Branches

Figure 1 shows the initialization and analysis branch for MOFIA 1.5. The names of the subroutines are shown in bold-face letters, followed by a brief description of what each subroutine does. Comments are provided within each of these subroutines, as well as in the associated parameter and common-block files. These subroutines are organized within files as follows:

File	**Subroutines**
geometry.F	OpenGeometry
	ReadDetGeom
read_calib.F	ReadCalibFiles
	ReadPlaneCorr
	ReadPlaneRot
	ReadWireCorr
	ReadResParams
setup_det.F	SetupDCPlanes
	SetupDCWires

Figure 2 shows the analysis branch. It was suggested at the last software meeting that histograms should be user defined. Two user files have been provided for this purpose: *udefine_hists.F* and *ufill_hists.F*, where the user may define and fill their histograms. HBOOK has been adopted as the package of choice for histogramming. Histograms may be written to file every "nEVTprocessed" events, where "nEVTprocessed" is a namelist variable (namelist HIST) that maybe assigned a value by the user (section 5.3). This variable is turned off by default so that histograms are not written to disk unless the user chooses otherwise. The user may also choose to write histograms to disk at any point during the analysis by issuing the command **func 5** at the MOFIA command line (see section 5.4).

The unpacked data is mapped on variables and arrays in the subroutine *MapSlide*, where

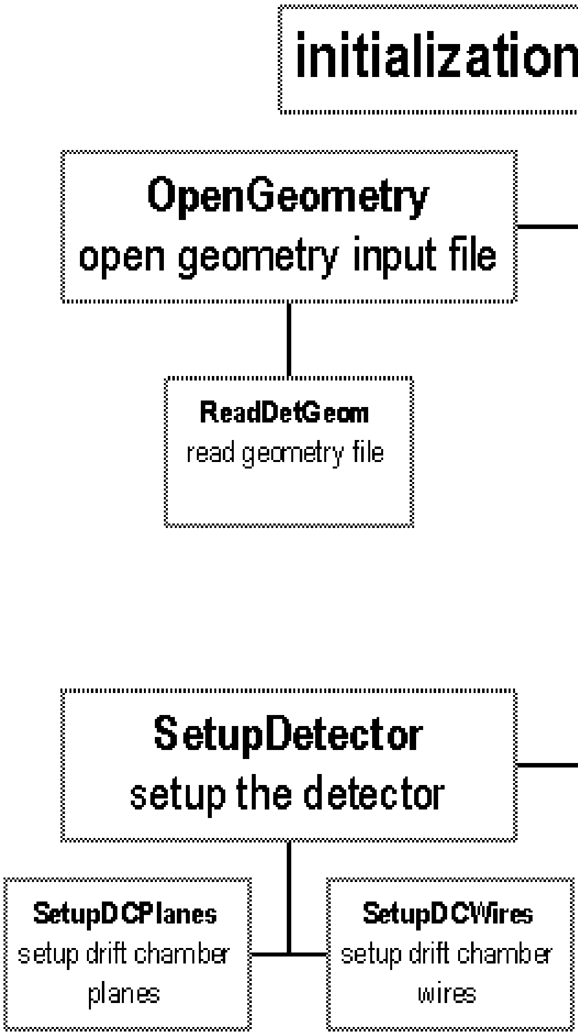

Figure 1: MOFIA 1.5 initialization branch.

the mapping is done in terms of "slides". Since some users might find this confining, an alternative subroutine, *MapNoSlide*, was provided. Since the most convenient mapping of the data depends on the tracking method, this subroutine is left for the user to write in a way that best satisfies his/her needs. Once a tracking method is adopted, the corresponding map file will be implemented.

The analysis subroutines are organized within files as follows:

File	**Subroutines**
tdcunp.F	tdcunp
tdcunp_bank.F	tdcunp_bank
mapping.F	ResetMapEvent MapEvent SortWires
udefine_hists.F	DefineRawHists DefineHists DefineTrackHists
ufill_hists.F	FillRawHists FillHists FillTrackHists
test_event.F	TestEvent
dplot.F	dplot

5 Compiling and Running MOFIA 1.5

MOFIA has a variety of commands that the user could execute at the MOFIA command line. A short listing and description of these commands is provided below. A complete

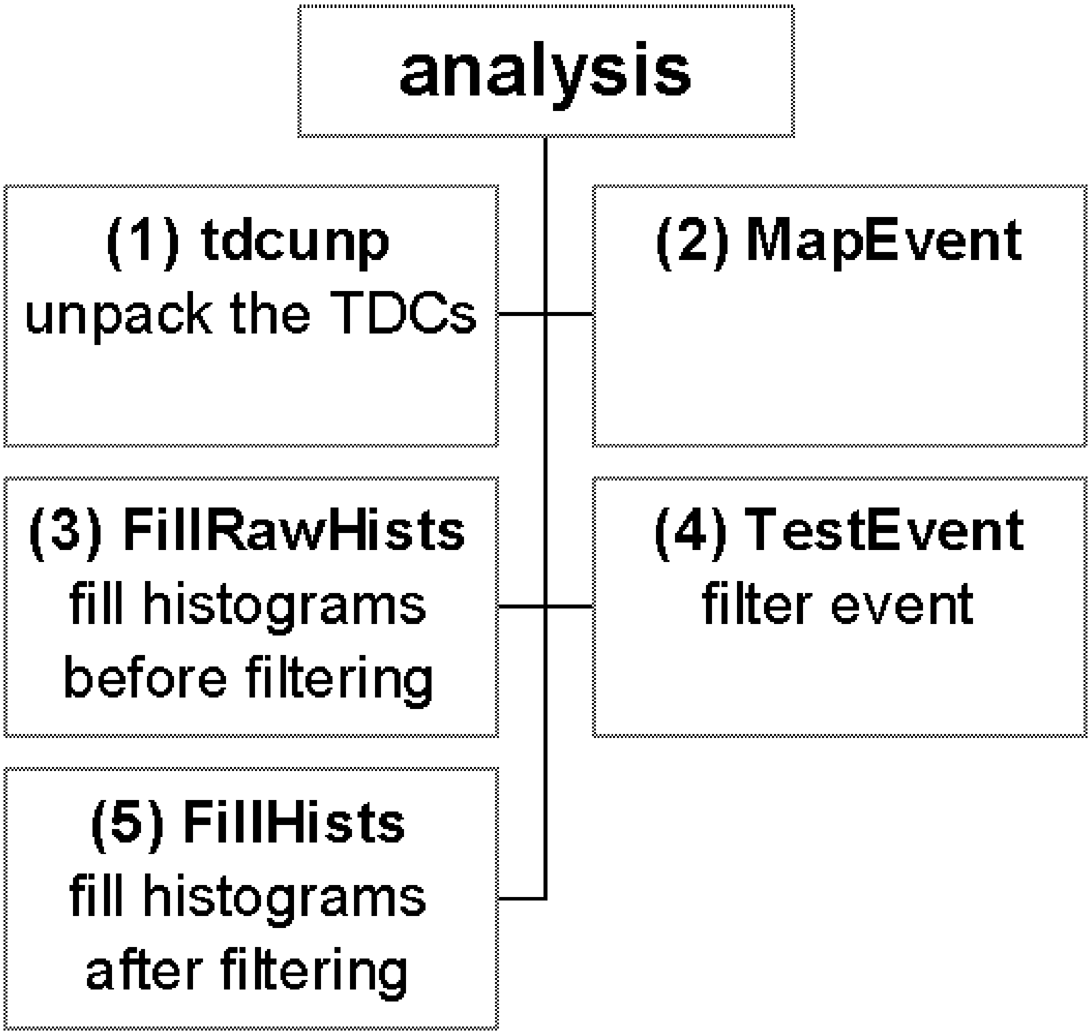

Figure 2: MOFIA 1.5 analysis branch.

listing and description may be accessed online using the MOFIA **help** command.

To compile MOFIA 1.5 type **make** in the directory that contains your *Makefile* and soure code. If you wish to add subroutines to the list of files to be compiled, edit the *Makefile* and add the ".o" file (corresponding to the ".F") to the **OBJECTS** list.

The executable name is specified in the *Makefile* and has the default name **mofia15**. To run MOFIA 1.5, type **mofia15** on the command line in the directory where you keep your executable. The user may wish to change the executable name, and may do so either on the command line when compiling MOFIA or by editing the *Makefile* (see the *Makefile* for more details).

Alternatively, one may use a command file to assign the environment variables and run the executable. This command file may also include any instructions to MOFIA that would otherwise be issued on the MOFIA command line (some of which are listed below). A sample command file $mofia15.kcm$ has been supplied with the code as an example.

5.1 General Commands

MTIN "argument": Assign the input file/device to argument. For example, **MTIN "/ralph1/usr/data8/e614/data/run00092.dat"** assigns the input file to $run00092.dat$ in directory $/ralph1/usr/data8/e614/data$. Similarly the command **MTIN "/dev/mx3d"** assigns the input to the tape drive $/dev/mx3d$.

analyze argument: Start the analysis process. The number of events to be analyzed may be used as an argument; for example **analyze 1000** analyzes 1000 events. If no argument is specified the entire file will be analyzed.

event argument: Move FORWARD to the event specified and analyze it. For example **event 237** moves forward to event 237 and analyzes it.

show argument: Show the current contents of the argument. For example **show MTIN** shows the name of the file/device that MTIN is assigned to. If no argument is provided, a listing of possible arguments is displayed.

@filename: Execute the command file specified by **filename**. Command files may include any MOFIA command that may be otherwise issued at the MOFIA command line. If the file extension is not specified the extension $.kcm$ is assumed. Command files may also be nested (so that a command file may call another) up to 10 layers deep.

exit: Exit MOFIA

help: Run the MOFIA help facility.

5.2 Flags

Several flags have been installed in MOFIA 1.5. To view these flags Type **show flags** at the MOFIA command line. MOFIA responds with the following:

MOFIA FLAGS:

BPRINT	=	OFF
PHOTO_FLAG	=	OFF
SKIM	=	OFF

	DECODE	TRACK	CUTS
DC	OFF	OFF	OFF
PC	OFF	OFF	OFF
SC	OFF	OFF	OFF
MC	OFF	OFF	OFF

The set command is used to change the contents of these flags. For example, to turn on the *PHOTO_FLAG* type **set photo_flag on** at the MOFIA command line. To turn on the the decoding for the DC detector subsystem type **set dc on**. To turn on the tracking and/or the cuts a qualifier following the subsystem name is expected. For example to turn on the tracking for the MC subsystem type **set mc/track on**.

5.3 Namelists

Several namelists have been installed in MOFIA 1.5. The following commands are used to access these namelists.

show namelist argument: Show namelist(s) and description. If no argument is provided all namelists and their descriptions are shown. If an argument is provided the contents of the namelist specified by the argument are shown as well as their respective description and default settings. For example **show namelist DCCUTS** shows the contents of the namelist DCCUTS.

namelist argument: Access the namelist specified by the argument, for purposes of checking and/or modifying the current settings. Invoking this command puts the user inside the namelist editor. Once inside the editor, the user may check the current values of the namelist by entering "?", exit the editor by entering "&", or change the value of a namelist variable by entering "**variable = value**".

5.4 Functions

Several functions have also been installed in MOFIA 1.5. The command: **function argument** allows the user to execute these functions. For example executing the command **function 7** prints out the drift chamber geometry files. If no argument is provided a listing of all functions and their descriptions is provided. The following is a list of all the functions currently available:

Function	Description
1	Turn ON printing for list of wire hits
2	Turn OFF printing " "
3	Turn ON printing for TDCUNP debugging
4	Turn OFF printing " "
5	Write HBOOK histograms to disk
6	Print geometry files
7	Print mapped variables for current event

5.5 Environment variables

Environment variables may also be assigned before executing MOFIA. Two convenient environment variables that come in handy for MOFIA are worth mentioning here: **MTIN** which specifies the input file/device as explained above, and **MHIST** which specifies the directory to which the hbook histograms should be written. If **MHIST** is not specified, the hbook histograms will be written to the current directory in which MOFIA is running. Some environment variables may be best to assign through the *.login* file. If multiple users are sharing the same account and wish to have their own definitions of environment variables, a file of the form *.user* (where user stands for the user name) would be handy. Each user would then have to **source .user** before running MOFIA.

6 Calibration File Manager (CFM)

E614 will also adopt the File Calibration Manager (CFM) developed for BNL experiment E787. CFM allows the user to associate run numbers with calibration files. Calibration types (plane position corrections, wire position corrections, etc.) may be defined through CFM. For example, DC_PPC defines the calibration type for the Drift Chamber Plane Position Corrections. CFM allows up to 5 characters to identify a component (Drift Chamber, DC, in this case) and up to 3 characters to identify its attributes (Plane Position Corrections, PPC). The file name for a calibration type has the form $XXXXX\_YYY.NNNNN$ where $NNNNN$ is a 5-digit (version) number called an *indicator*. For example $DC\_PPC.00001$ is a PPC file name with indicator number equal to 1. CFM is then asked to associate a run number, or a group of runs with indicator 1 for the calibration type DC_PPC. When MOFIA is run it checks the run number and obtains the *indicator* number and the file name for each calibration type from CFM. CFM expects all calibration files to reside in the directory defined by $\$CAL\_DB$. To run CFM type **CFM** (outside MOFIA). Commands may be issued at the CFM command line to view the contents of CFM or modify and change the existing information. The **help** command provides the user with online help. For a more detailed description of CFM, please refer to the E787 document by Morgan Burke.

Several calibration files have been defined in CFM for use with MOFIA 1.5, and are listed below:

1:	DC_PPC	Drift Chamber Plane Position Corrections
2:	DC_WPC	Drift Chamber Wire Position Corrections
3:	DC_RES	Drift Chamber RESolution parameter
4:	DC_PRC	Drift Chamber Plane Rotation Corrections
5:	DT_GEO	DeTector GEOmetry
6:	$FBC1\_MAP$	Mapping file for first FASTBUS crate
7:	$FBC2\_MAP$	Mapping file for second FASTBUS crate
8:	$FBC3\_MAP$	Mapping file for third FASTBUS crate

7 MOFIA 1.0 Contents and Description

As mentioned above, **MOFIA 1.0** contains the code used to analyse the data obtained in 1997 using a set of seven drift chamber planes. The main differences between MOFIA

1.5 and MOFIA 1.0 are the following:

1. The geometry description of the test chamber is implemented in the code through parameter files and common blocks.

2. Flags and cuts are assigned in two ascii input files: *switches.in* and *eventfilter.in* rather than through namelists. A description of the contents of these files is provided through comments within these files. The different settings were implemented through parameter files (again, rather than through namelists).

3. MOFIA 1.0 is a full analysis of the test chamber data that includes:

- straight line tracking
- efficiency calculations
- calibrations including finding plane positions, wire positions, TDC time zero, and tracking resolution.

8 Running MOFIA 1.0

The environment variables and libraries are set by default to values appropriate for MOFIA 1.5. To run MOFIA 1.0 the user needs to execute the follwing commands:

setup mofia 1.0
mlib

before running MOFIA. This will result in resetting the libraries and environment variables to values appropriate for MOFIA 1.0.
The commands for MOFIA 1.0 are the same as those for 1.5 (described in section 5.1. The namelists 5.3 and functions 5.4, however, are different. The namelists in this version are mostly left over from KOFIA, and have not been adapted for use in MOFIA 1.0. To view the available functions in MOFIA 1.0 type **func** at the MOFIA command line.

9 Coding Conventions

At the last software collaboration meeting, several ideas were suggested to be adopted as coding conventions for E614. In the list below, I will mention the ones that have been agreed to unanimously. The software group realizes, however, that the most important criteria for code writing is to choose clear program names, variable names, and provide detailed comments throughout the code.

1. Always use **implicit none** to force declarations of all variables.

2. Always use **include files** to define **common** variables. All include files should have a complete documentation to describe all variables.

3. Use only one common block per include file.

4. All variables should be completely defined as INTEGER*8, REAL*4, etc. (rather than INTEGER, REAL).

5. Variable declarations in commons should start with *8, then *4, then character and logical variables.

6. Variables will not be case sensitive.

7. Allow long lines (compile with $-extend\_source$).

8. Variable names can have up to a max of 31 characters

9. Re-compile all code when making changes (i.e. remove all *.o files before compiling).

10. Make sure that variable names used are not cpp reserved names (ex. UNIX, MIPS, ALPHA, etc).