Manual on using Version 43 of the Offline Code

by **J. F. McCullough**

June 19, 2000

Abstract

I have written a short reference manual to use with version 43 of the offline code. This is meant to be used as a reference only and is by no means a comprehensive guide on how to use the offline code.

1 Introduction

Because the offline (and online) codes are continually evolving, it can be difficult to keep track of all the different structures and routines that we use. This is especially true when the routines are constantly changing with each new version. Another difficulty is trying to (correctly) access a variable that is buried 5 structures down (i.e. the Julian Date is found in eventData.recEvent.reconInfo.clockData.julDate). I have therefore written a short memo that can be used as a reference when analyzing data with version 43 of the offline code.

The two data types that are saved to tape are raw data and reconstructed data. Raw data basically consists of edges and arrivals times (for each edge) for every PMT hit. Raw data is approximately 500 bytes/events (when running at 1500Hz) but varies in size depending upon the number of PMTs hit and the number of edges per PMT. We currently save raw data for the Crab (SA tapes), Mrk421/501 (SB tapes), the Sun/Moon (SC tapes), triggered GRBs (SD tapes) and for the entire sky at a higher threshold (D tapes). For the Crab, Mrk421, and the Sun/Moon, we save all raw data within ± 8 degrees (in declination) of the source. For triggered GRBs, we save all raw data for the entire sky 1/2 hour before the trigger and 1 and 1/2 hours after the trigger. For the entire sky, we save all the raw data for a 200 PMT threshold. This corresponds to an event rate of about 50Hz.

Reconstructed data is data that has been reconstructed online. It currently consists of 16 different quantities (see Section 2), the most important of which are the time and direction (RA and Dec) of each event. Reconstructed data is exactly 24 bytes (6 words) in length for each event. Both the raw and reconstructed data are saved to tape in a compressed format.

Everything in this memo assumes the following variable declarations. These should be declared at the beginning (right after main is declared) of any program that reads data:

```
EVENT_DATA        eventData;
ANGLE_RECON       AngleFit;
CORE_RECON        CoreFit;
```

MUON_RECON	Muon;
GH_RECON	ghInfo;
GRID_COORD	coords;
PMT_INFO	pmtInfo;
ANGLE_CONTROL	angleCon;
CORE_CONTROL	coreCon;
GH_CONTROL	ghCon;
DATA_CONTROL	dataCon;
CAL_STATUS	calStatus;
EDGE_INFO	edgeInfo;
CAL_CONTROL	calCon;

The structures are defined in Reconstruction.h and Offline.h, both of which are in /disk02/Milagro/analysis/offline_v43/include. Section 2 describes the reconstructed data and section 3 describes the raw data. The definitions and units for all the quantities are given in the Appendix. Also given in the Appendix is a list of what data runs were processed with which version of the offline (online) code.

2 Reconstructed Data

The following quantities are saved (compressed) from the online reconstruction for each event:

Quantity	word	bits	precision saved
ra	0	16-31	0.01 degrees
dec	0	0-15	0.01 degrees
chisq	1	24-31	0.25 degrees
xCore	1	15-23	100 cm
yCore	1	6-14	100 cm
errCore	1	0-5	200 cm
PEs top	2	22-31	nHit_top PEs
nHit top	2	13-21	no uncertainty
nFit	2	4-12	no uncertainty
nHit out	2	0-3	no uncertainty
PEs bottom	3	22-32	nHit_bottom PEs
nHit bottom	3	13-21	no uncertainty
weighted sum	3	0-12	0.0625
max PE bottom	4	22-31	1 PE
future hadronicity	4	0-21	-
time	5	8-31	1 us
time Error	5	0-7	n/a

The precision that the total number of PEs for the air-shower (PEs top) and muon layer (PEs bottom) are saved to is equal to the number of PMTs hit in each layer (nHit_top and nHit_bottom).

For example, if 120 PMTs are hit in the air-shower layer, then the total number of PEs will be saved to a precision of 120 PEs. The value given by time Error is an integer that describes any errors (given by the GPS clock) in the event time (0 = no error). There is also a header at the beginning of each subrun which gives the run number, the subrun number, and the Julian date.

When reading reconstructed data, the only functions which need to be called are InitControl(&angleCon,&coreCon,&dataCon,&ghCon,&calCon) and MilagroEvent2(&eventData, &dataCon, &coords). InitControl doesn't really need to be called as long as dataCon.source is defined (i.e. dataCon.source = DATA_MGRORAW4 for version 43) before MilagroEvent2 is called. Defining dataCon.source lets MilagroEvent2.c know which DataRead program to use (i.e. DataRead_MCASCII3.c for MC data or DataRead_MGRORAW4.c for raw or reconstructed data).

After MilagroEvent2(readMode,&eventData, &dataCon, &coords) is called, the following quantities are filled:

runNum	=	eventData.subRunInfo2.runNum;	(int)
subRunNum	=	eventData.subRunInfo2.subRunNum;	(int)
version	=	eventData.subRunInfo2.versionNum;	(int)
time	=	eventData.recEvent.reconInfo.clockData.eventTime;	(double)
julDate	=	eventData.recEvent.reconInfo.clockData.julDate;	(unsigned int)
time_err	=	eventData.recEvent.reconInfo.clockData.timeErr;	(short int)
nhit	=	eventData.recEvent.reconInfo.nPMTs;	(short int)
nhit_as	=	eventData.recEvent.reconInfo.poolFit.nPMT[0];	(unsigned int)
nhit_muon	=	eventData.recEvent.reconInfo.poolFit.nPMT[1];	(unsigned int)
nhit_out	=	eventData.recEvent.reconInfo.poolFit.nPMT[2];	(unsigned int)
pes_as	=	eventData.recEvent.reconInfo.poolFit.nPE[0];	(unsigned int)
pes_muon	=	eventData.recEvent.reconInfo.poolFit.nPE[1];	(unsigned int)
nfit	=	eventData.recEvent.reconInfo.angleFit.nFit;	(int)
chisq	=	eventData.recEvent.reconInfo.angleFit.chiSq;	(float)
ra	=	eventData.recEvent.reconInfo.angleFit.raF;	(float)
dec	=	eventData.recEvent.reconInfo.angleFit.decF;	(float)
weightSum	=	eventData.recEvent.reconInfo.angleFit.weightSum;	(float)
xcore	=	eventData.recEvent.reconInfo.coreFit.xCoreF;	(float)
ycore	=	eventData.recEvent.reconInfo.coreFit.yCoreF;	(float)
errCore	=	eventData.recEvent.reconInfo.coreFit.errCore;	(float)
gby_cut	=	eventData.recEvent.reconInfo.ghRecon.gh_GBY;	(float)
gby_cut_satcorr	=	eventData.recEvent.reconInfo.ghRecon.gh_GBY_satcorr;	(float)
max_muon	=	eventData.recEvent.reconInfo.ghRecon.maxPEmuon;	(float)
max_muon_satcorr	=	eventData.recEvent.reconInfo.ghRecon.maxPEmuon_satcorr;	(float)
sizeF	=	eventData.recEvent.reconInfo.showerFit.sizeF;	(float)
ageF	=	eventData.recEvent.reconInfo.showerFit.ageF;	(float)
theta	=	eventData.recEvent.reconInfo.angleFit.thetaF;	(float)
phi	=	eventData.recEvent.reconInfo.angleFit.phiF;	(float)

Note: sizeF and ageF are not currently reconstructed. They are assigned values of -999.0 in CompressRec2.c. Also, you must first call FillThPh(&(eventData.recEvent)) before accessing theta and phi. Otherwise, they will both equal -999.0.

Other less useful quantities (Note: many of these are not currently filled):

MONTE	=	eventData.recEvent.reconInfo.MONTE;	(short int)
eventNum	=	eventData.recEvent.reconInfo.eventNum;	(unsigned int)
triggerID	=	eventData.recEvent.reconInfo.triggerId;	(unsigned int)
headerID	=	eventData.subRunInfo2.headerID;	(int)
fileTypeID	=	eventData.subRunInfo2.fileTypeID;	(int)
dataType	=	eventData.subRunInfo2.dataType;	(int)
blocked	=	eventData.subRunInfo2.blocked;	(int)
recFlag	=	eventData.subRunInfo2.recFlag;	(int)
cmpFlag	=	eventData.subRunInfo2.cmpFlag;	(int)
YEAR	=	eventData.subRunInfo2.errInfo.YEAR;	(int)
nEvents	=	eventData.subRunInfo2.errInfo.nEvents;	(int)
badEvt	=	eventData.subRunInfo2.errInfo.badEvt;	(int)
buffSynch_errors	=	eventData.subRunInfo2.errInfo.buffSynch;	(int)
buffSkip_errors	=	eventData.subRunInfo2.errInfo.buffSkip;	(int)
fbHeader_errors	=	eventData.subRunInfo2.errInfo.fbHeader;	(int)
gpsBCD_errors	=	eventData.subRunInfo2.errInfo.wordCount;	(int)
wordCount_errors	=	eventData.subRunInfo2.errInfo.gpsBCD;	(int)
gpsSeq_errors	=	eventData.subRunInfo2.errInfo.gpsSeq;	(int)
gpsMin_errors	=	eventData.subRunInfo2.errInfo.gpsMin;	(int)
gpsHour_errors	=	eventData.subRunInfo2.errInfo.gpsHour;	(int)
gpsDay_errors	=	eventData.subRunInfo2.errInfo.gpsDay;	(int)
fsccSocket_errors	=	eventData.subRunInfo2.errInfo.fsccSocket;	(int)
arrayBounds_errors	=	eventData.subRunInfo2.errInfo.arrayBounds;	(int)
bytesOut_errors	=	eventData.subRunInfo2.errInfo.bytesOut;	(int)

Note: The above error information (any quantity in the structure eventData.subRunInfo2 except YEAR) refers to the previous subrun and not the current subrun. Also, eventNum has no physical meaning and is filled to -999 in CompressRec2.c

3 Raw Data

When looking at raw data, it is important to have the environmental variable CONFIG_MILAGRO correctly defined to the appropriate directory. When using tcsh shell, this is done by the following command:

setenv CONFIG_MILAGRO /disk02/Milagro/analysis/offline_v43/config_milagro

CONFIG_MILAGRO tells the offline code which directory to look in for configuration files (i.e. PMT positions, calibration data, ...). If this is set to the wrong directory (i.e. /disk02/Milagro/analysis/offline_v32/config_milagro) you may not be using the most recent calibrations. The easiest way to prevent this is to put the above command in your .cshrc file (if using tcsh shell).

When analyzing raw data, the following functions must be called in the following order:

1) **GetBadPMTs(&pmtInfo)**

This opens DontFits.txt and reads a list of PMTs that should not be used in the core and angle fitters. This is currently not used (DontFits.txt is empty), however it may be implemented in future versions of the online and offline code.

After GetBadPMTs(&pmtInfo) is called, the following values are filled:

dontFit	=	pmtInfo.dontFit [i=0 - i=PMT_cal];	(int)
nToss	=	pmtInfo.nToss [i=0 - i=PMT_cal];	(int)

2) **InitControl(&angleCon,&coreCon,&dataCon,&ghCon,&calCon)**

InitControl controls which methods to use for calibrations and which algorithms to use for core fitting, sampling, angle fitting, ... All of the reconstruction routines are called in Recon.c, but which algorithms to use are defined in InitControl.c. The following control options are available:

(**Note:** In version 43 there are now InitControl functions for each of the older versions of the offline code. They are called InitControl_vXX.c where XX is the version number: 32,41,42, or 43 (InitControl_v43.c and InitControl.c are identical). To change to an older version of the offline code, you can just replace the call to InitControl in your example program to InitControl_vXX. Each InitControl_vXX.c program contains the correct defaults for that particular version.)

Data Source - MC or Data
dataCon->source=DATA_MCASCII;
dataCon->source=DATA_MCASCII2;
dataCon->source=DATA_MCASCII3;
dataCon->source=DATA_MGRORAW;
dataCon->source=DATA_MGRORAW2;
dataCon->source=DATA_MGRORAW3;
dataCon->source=DATA_MGRORAW4; **(default for v43)**
dataCon->inFile = data file or directory
dataCon->nEventsWant = number of events to process
dataCon->skipSwitch = number of events to skip

dataCon->source tells MilagroEvent2.c which DataRead program to use. MCASCII3 calls DataRead_MCACII3.c which is used for MC data. MGRORAW4 calls DataRead_MGRORAW4.c which is used for raw and reconstructed data. The others are older versions of the DataRead programs and should not be used (the newer versions are backwards compatible). DataCon->inFile tells MilagroEvent2.c which file or directory you want to read while dataCon->nEventsWant is how many events you want to process. DataCon->skipSwitch is how many events you want to skip (the default value is zero).

Edge Finder
calCon->EdgeMethod = MILAGRO_EDGEFINDER;
calCon->EdgeMethod = MILAGRO_EDGEFINDER2;
calCon->EdgeMethod = MILAGRO_EDGEFINDER3; **(default for v43)**

calCon->EdgeMethod tells GetEdge.c which edgefinder to use. MILAGRO_EDGEFINDER calls GetEdgeMilagro.c which is the first edge finder optimized for Milagro data. MILAGRO_EDGEFINDER2 calls GetEdgeMilagro2.c which is identical to the old edge finder except that it requires GOODTIME cuts for all hits, including those with exactly 2 or 4 edges. MILAGRO_EDGEFINDER3 calls GetEdgeMilagro3.c which is the new edge finder which takes into account the fact that the airshower and muon layers have different TZEROs.

TOT-PE calibration
calCon.PeMethod = ADC_TOTPE;
calCon.PeMethod = OCC1_TOTPE;
calCon.PeMethod = OCC2_TOTPE;
calCon.PeMethod = OCC3_TOTPE;
calCon.PeMethod = SPEC_TOTPE; **(default for v43)**
calCon.PeMethod = LASER1_TOTPE;

calCon.PeMethod tells CalibrateRaw.c which TOT to PE calibrations to use. ADC_TOTPE calls TOTPE_ADC.c which is the ADC based TOT-PE calibrations. OCC1_TOTPE and LASER1_TOTPE (they are equivalent) call TOTPE_Occ1.c which is the first laser calibrations of TOT-PE. OCC2_TOTPE calls TOTPE_Occ2.c which is Tony's (the Flesher's) laser based TOT-PE calibrations. SPEC_TOTPE calls TOTPE_SPEC.c which is the spectrum based TOT-PE calibrations. OCC3_TOTPE calls TOTPE_OCC3.c which is the Flesher's recently released laser based TOT-PE calibrations.

Slewing/TPED calibration
calCon.SlewMethod = TIMING_INITIAL;
calCon.SlewMethod = TIMING_DATA1;
calCon.SlewMethod = TIMING_LASER1;
calCon.SlewMethod = TIMING_LASER2;
calCon.SlewMethod = TIMING_LASER3; **(default for v43)**

calCon.SlewMethod tells CalibrateRaw.c which slewing/Tped calibrations to use. TIMING_INITIAL calls ElecSlew.c which is the slewing calibrations from a single PMT with no TPeds. TIMING_DATA1 calls ElecSlew1a.c which is the old slewing from a single PMT but with TPeds derived from data. TIMING_LASER1 calls ElecSlew2.c which is the initial laser timing calibrations. TIMING_LASER2 calls ElecSlew3.c which is Tony's (the Flesher's) laser timing calibrations. TIMING_LASER3 calls ElecSlew4.c which is the initial laser timing calibrations with HiTOT extrapolation.

Core fitters
coreCon->method = CORE_FIT_CENTER_OF_MASS; **(default for v43)**
coreCon->method = CORE_FIT_FFT;
coreCon->method = CORE_FIT_GAUSS;

coreCon->method defines which core fitter to use. CORE_FIT_CENTER_OF_MASS calls CoreFit_COM.c which is the standard center of mass core fitter we used in Milagrito. CORE_FIT_FFT calls CoreFit_FFT.c which is an FFT core fitter based upon the frequency spectrum of the pulse height distributions. CORE_FIT_GAUSS calls CoreFit_GAUSS.c which is a gaussian core fitter which fits the signal distribution to a four parameter axisymmetric gaussian model.

Angle Fitters
angleCon->method = ANGLE_FIT_CHISQ;
angleCon->method = ANGLE_FIT_CHISQ_FLATTOP; **(default for v43)**
angleCon->method = ANGLE_FIT_LIKELIHOOD;
angleCon->method = ANGLE_FIT_3D;

angleCon->method defines which angle fitter to use. ANGLE_FIT_CHISQ calls AngleFit_ChiSq.c which is the chi-square angle fitter we've been using since the beginning of time. ANGLE_FIT_CHISQ_FLATTOP calls AngleFit_FlatTop.c which is the same as AngleFit_ChiSq.c except that you can only use the air-shower layer (AngleFit_ChiSq.c has the option of using both). ANGLE_FIT_LIKELIHOOD does not call anything. It is a place holder for a future likelihood angle fitter. ANGLE_FIT_3D calls AngleFit_3D.c which is 3D chi-square angle fitter that can also incorporate the outriggers. **Note:** There have been significant changes to how the angle fitters are organized in version 43. See REVISION.HISTORY in /disk02/Milagro/analysis/offline_v43 for an explanation of these changes.

Sampling Correction
angleCon->sampling_method = SAMPLE_MC_1;
angleCon->sampling_method = SAMPLE_MC_2;
angleCon->sampling_method = SAMPLE_MC_3; **(default for v43)**

angleCon->sampling_method defines which sampling correction to use. SAMPLE_MC_1 calls sampleCorr1.c which is the earliest sampling correction derived from MC gamma showers. SAMPLE_MC_2 calls sampleCorr2.c which is also derived from (better) MC gamma showers (this is the sampling we've used for Milagro data so far). SAMPLE_MC_3 calls sampleCorr3.c which is the sampling correction derived from data.

Curvature Correction
angleCon->curvature = 0.04;

angleCon->curvature defines the (linear) curvature correction in ns/m.

Weights used in the Angle Fitter
angleCon->weight_method = WEIGHT1;
angleCon->weight_method = WEIGHT2;
angleCon->weight_method = WEIGHT3; **(default for v43)**

angleCon->weight_method defines the weighting function used in the angle fitters. The functions are given in AngleFit_Variables.c in the function Sigma(pes,angleCon). WEIGHT1 are the weights from version 32 and 41, WEIGHT2 from version 42, and WEIGHT3 from version 43.

PE and SIG cuts used in the Angle Fitter
angleCon->PE_SIG_cuts_method = CUTS1;
angleCon->PE_SIG_cuts_method = CUTS2;
angleCon->PE_SIG_cuts_method = CUTS3; **(default for v43)**

angleCon->PE_SIG_cuts_method defines the PE and SIG cuts used in the 5 passes of the angle fitter. The PE and SIG cuts are given in AngleFit_Variables.c in the function InitialzeCuts(angleCon). CUTS1 are the cuts from version 32 and 41, CUTS2 from version 42, CUTS3 from version 43.

Relax condition used in the Angle Fitter
angleCon->relax_method = RELAX1;
angleCon->relax_method = RELAX2; **(default for v43)**

angleCon->relax_method defines the relax condition for the angle fitter. The relax conditions are given in AngleFit_Variables.c in the function InitialzeCuts(angleCon). RELAX1 sets the relax condition to 20 PMTs and is used in versions 32 and 41. RELAX2 sets the relax condition to 450 PMTs and is used in versions 42 and 43.

Early light condition used in the Angle Fitter
angleCon->early_light_method = EARLY_LIGHT1;
angleCon->early_light_method = EARLY_LIGHT2; **(default for v43)**

angleCon->early_light_method defines the early light condition for the angle fitter. The early light conditions are given in AngleFit_Variables.c in the function InitialzeCuts(angleCon). EARLY_LIGHT1 is used in versions 32 and 41. EARLY_LIGHT2 is used in versions 42 and 43.

See the memo on Tuning the Angle Fitter (4/5/2000) for an explanation of the relax and early light conditions as well as a description of the PE and SIG cuts.

3) **GetCoords(&coords,&dataCon)**

GetCoords reads in the coordinates of each phototube. The PMT coordinates are used in the core and angle fitters. After GetCoords(&coords,&dataCon) is called the following values are filled:

i	=	coords.i [i=0 - i=NTUBES=1500];	(int)
j	=	coords.j [i=0 - i=NTUBES=1500];	(int)
k	=	coords.k [i=0 - i=NTUBES=1500];	(int)
ijkmap	=	coords.ijkmap [i=0 - i=25] [j=0 - j=18] [k=0 - k=2];	(int)
mc2data	=	coords.mc2data [i=0 - i=NTUBES=1500];	(int)
data2mc	=	coords.data2mc [i=0 - i=NTUBES=1500];	(int)
fbAdd	=	coords.fbAdd [i=0 - i=NTUBES=1500];	(int)
tdcChan	=	coords.tdcChan [i=0 - i=NTUBES=1500];	(int)
tdcMap	=	coords.tdcMap [i=0 - i=29] [i=0 - i=95];	(int)
x	=	coords.x [i=0 - i=NTUBES=1500];	(float)
y	=	coords.y [i=0 - i=NTUBES=1500];	(float)
z	=	coords.z [i=0 - i=NTUBES=1500];	(float)
xxi	=	coords.xxi [i=0 - i=NTUBES=1500];	(float)
yyi	=	coords.yyi [i=0 - i=NTUBES=1500];	(float)
zzi	=	coords.zzi [i=0 - i=NTUBES=1500];	(float)
xzi	=	coords.xzi [i=0 - i=NTUBES=1500];	(float)
yzi	=	coords.yzi [i=0 - i=NTUBES=1500];	(float)
xyi	=	coords.xyi [i=0 - i=NTUBES=1500];	(float)

4) **MilagroEvent2(&eventData, &dataCon, &coords)**

MilagroEvent2 is the routine that reads all the data on an event by event basis. After calling MilagroEvent2(&eventData, &dataCon, &coords), the following values are filled:

nhit	=	eventData.cmpEvent.nHits;	(int)
igrid	=	eventData.cmpEvent.gridNo [i=0 - i=nhit];	(int)
rawLatch	=	eventData.cmpEvent.rawLatch [i=0 - i=9];	(unsigned int)
julDate	=	eventData.cmpEvent.gps.julDate;	(unsigned int)
eventTime	=	eventData.cmpEvent.gps.eventTime;	(double)
timeErr	=	eventData.cmpEvent.gps.timeErr;	(short int)
adc	=	eventData.cmpEvent.adc.cnts [igrid];	(unsigned int)
nedge	=	eventData.cmpEvent.tdc.nEdge [igrid];	(unsigned int)
phase_nedge	=	eventData.cmpEvent.tdc.edge [igrid] [nedge];	(unsigned int)
arrival_time_nedge	=	eventData.cmpEvent.tdc.cnts [igrid][nedge];	(unsigned int)

NOTE: ALL OF THE QUANTITIES RECONSTRUCTED ONLINE ARE ALSO FILLED (See Section 2)

5) **CalibrateRaw(&(eventData.cmpEvent), &pmtInfo, &calStatus, &edgeInfo, &(eventData.calEvent.calData), &calCon)**

CalibrateRaw is the routine that calibrates all of the raw data. The raw data basically consists of edges and times (of each edge) for each PMT. After the data is calibrated, each PMT now has an arrival time and a pulse height.

After calling CalibrateRaw(&(eventData.cmpEvent), &pmtInfo, &calStatus, &edgeInfo,&(eventData.calEvent.calData), &calCon) the following quantites are filled:

eventNumber	=	eventData.calEvent.calData.eventNum;	(unsigned int)
nPMTs_cal	=	eventData.calEvent.calData.nPMTs;	(int)
nAS_cal	=	eventData.calEvent.calData.nAS;	(int)
nMUON_cal	=	eventData.calEvent.calData.nMUON;	(int)
nOut_cal	=	eventData.calEvent.calData.nOUT;	(int)
igrid	=	eventData.calEvent.calData.gridNo [i=0 - i=nPMTs_cal];	(int)
pes	=	eventData.calEvent.calData.pes [igrid];	(float)
rawLatch	=	eventData.cmpEvent.calData.rawLatch [i=0 - i=9];	(unsigned int)
PMT_time	=	eventData.calEvent.calData.time [igrid];	(float)
julDate	=	eventData.calEvent.calData.gps.julDate;	(unsigned int)
eventTime	=	eventData.calEvent.calData.gps.eventTime;	(double)
timeErr	=	eventData.calEvent.calData.gps.timeErr;	(short int)
nPMTs	=	edgeInfo.nPMTS;	(int)
goodEdge	=	edgeInfo.goodEdge [i=0 - i=nPMTs_cal];	(int)
startEdge	=	edgeInfo.startEdge [i=0 - i=nPMTs_cal];	(int)

6) **Recon(&(eventData.calEvent), &coords, &pmtInfo, &AngleFit, &CoreFit, &Muon, &ghInfo, &angleCon, &coreCon, &ghCon)**

This is where all of the reconstruction routines are called. In Recon.c, the following functions are called (the order is important):

Core Fit
coreCon->layer=1; (Air Shower Layer)
CoreFit(pmtInfo,&cal->calData,coords,coreInfo,coreCon)

Sampling Correction
sampleCorr(cal,angleCon)

Curvature Correction
curvCorr(&cal->sampData,coords,coreInfo,&cal->curvData,angleCon)

Angle Fit angleCon->layer=1; (Air Shower Layer)
AngleFit(pmtInfo,&cal->curvData,coords,coreInfo,angleInfo,angleCon)

(Combine angles using CombineAngle.c - Only relevant if the muon layer is also used)
CombineAngle(angleInfo,angleCon)

Muon Fit
muonCand(&cal->calData,coords,Muon)

Gamma-Hadron Seperation
ghSep(&cal->calData,coreInfo,ghInfo,ghCon)

After calling Recon(&(eventData.calEvent),&coords,&pmtInfo, &AngleFit, &CoreFit, &Muon, &ghInfo,&angleCon,&coreCon, &ghCon) the following values are filled:

theta_fit	=	AngleFit.thetaF;	(float)
phi_fit	=	AngleFit.phiF;	(float)
theta_fit_as	=	AngleFit.thetaF_as;	(float)
phi_fit_as	=	AngleFit.phiF_as;	(float)
theta_fit_mu	=	AngleFit.thetaF_mu;	(float)
phi_fit_mu	=	AngleFit.phiF_mu;	(float)
nfit	=	AngleFit.nFit;	(int)
nfit_as	=	AngleFit.nFit_as;	(int)
nfit_mu	=	AngleFit.nFit_mu;	(int)
ra_fit	=	AngleFit.raF;	(float)
dec_fit	=	AngleFit.decF;	(float)
ra_fit_as	=	AngleFit.raF_as;	(float)
dec_fit_as	=	AngleFit.decF_as;	(float)
ra_fit_mu	=	AngleFit.raF_mu;	(float)
dec_fit_mu	=	AngleFit.decF_mu;	(float)
weighted_sum	=	AngleFit.weightSum;	(float)

xcore_fit	=	CoreFit.xCoreF;	(float)
xcore_fit_as	=	CoreFit.xCoreF_as;	(float)
xcore_fit_mu	=	CoreFit.xCoreF_mu;	(float)
ycore_fit	=	CoreFit.yCoreF;	(float)
ycore_fit_as	=	CoreFit.yCoreF_as;	(float)
ycore_fit_mu	=	CoreFit.yCoreF_mu;	(float)
errCore	=	CoreFit.errCore	(float)
coreDist	=	CoreFit.coreDist [i=0 - i=nPMTs_cal];	(float)
tChis	=	pmtInfo.tChis [i=0 - i=nPMTs_cal];	(float)
weights	=	pmtInfo.weights [i=0 - i=nPMTs_cal];	(float)
TOTlo	=	pmtInfo.TOTlo [i=0 - i=nPMTs_cal];	(int)
TOThi	=	pmtInfo.TOThi [i=0 - i=nPMTs_cal];	(int)
Used	=	pmtInfo.Used [i=0 - i=nPMTs_cal];	(int)
WhyTossed	=	pmtInfo.WhyTossed [i=0 - i=nPMTs_cal];	(int)
nNoEdge	=	pmtInfo.nNoEdge;	(int)
nNoSlew	=	pmtInfo.nNoSlew;	(int)
nNoPETOT	=	pmtInfo.nNoPETOT;	(int)
nNoSlewOOB	=	pmtInfo.nNoSlewOOB;	(int)
nMuon	=	Muon.nMuon;	(short int)
muonPesTot	=	Muon.muonPesTot;	(short int)
muonPes	=	Muon.muonPes [i=0 - nMuon];	(float)
ambPes	=	Muon.ambPes [i=0 - nMuon];	(float)
muonX	=	Muon.muonX [i=0 - nMuon];	(float)
muonY	=	Muon.muonY [i=0 - nMuon];	(float)
maxPEmuon	=	ghInfo.maxPEmuon;	(float)
maxPEmuon_satcor	=	ghInfo.maxPEmuon_satcorr;	(float)
gh_GBY	=	ghInfo.gh_GBY;	(float)
gh_GBY_satcorr	=	ghInfo.gh_GBY_satcorr;	(float)

4 Appendix

The variables are given in the order that they appear in the memo. Some variables are listed more than once if they appeared numerous times within the memo. The variables that are listed as not used are currently not filled by any of the offline programs. If accessed, the will always give a value of 0.

Variable name	**definition**
ra	Right Ascension (degrees) in 2000 coordinates
dec	Declination (degrees) in 2000 coordinates
chisq	weighted ChiSqr (degrees) from the angle fit
xCore	x position (cm) of fitted core
yCore	y position (cm) of fitted core

errCore	estimated error (cm) in fitted core position
PEs top	sum of PEs of air-shower layer phototubes
nHit top	total number of air-shower layer PMTs hit (and calibrated)
nFit	number of PMTs used in the angle fit
nHit out	total number of outriggers hit (and calibrated)
PEs bottom	sum of PEs of muon layer phototubes
nHit bottom	total number of muon layer PMTs hit (and calibrated)
weight sum	weighted sum of PMTs used in the angle fit
max PE bottom	PMT with the highest PE value in the muon layer
future hadronicity	place holder for future gamma/hadron separation quantities
time	time of event (seconds from noon UT)
time Error	integer which describes time error (0 = no time error)
runNum	run number of event (runs change with Julian Date)
subRunNum	sunRun number of event
version	version of online code used for reconstruction
time	time of event (seconds from noon UT)
julDate	(Milagro) modified Julian Date = MJD - 50000
time_err	integer which describes time error (0 = no time error)
nhit	total number of PMTs hit (air-shower + muon + outrigger)
nhit_as	total number of air-shower layer PMTs hit
nhit_muon	total number of muon layer PMTs hit
nhit_out	total number of outriggers hit
pes_as	sum of PEs of air-shower layer PMTs
pes_muon	sum of PEs of muon layer PMTs
nfit	number of PMTs used in the angle fit
chisq	weighted ChiSqr (degrees) from the angle fit
ra	Right Ascencion (degrees) in 2000 coordinates
dec	Declination (degrees) in 2000 coordinates
weight sum	weighted sum of PMTs used in the angle fit
xcore	x position (cm) of fitted core
ycore	y position (cm) of fitted core
errCore	estimated error (cm) in fitted core position
maxPEmuon	PMT with the highest PE value in the muon layer
maxPEmuon_satcorr	saturation corrected maxPEmuon
gh_GBY	(float)nhit_muon/maxPEmuon
gh_GBY_satcorr	(float)nhit_muon/maxPEmuon_satcorr
sizeF	fitted size of the air shower - not currently used
ageF	fitted age of the air shower - not currently used
theta	fitted theta (degrees)
phi	fitted phi (degrees)
MONTE	integer which describes data type
eventNum	event number - not used
triggerID	not used

headerID	flag for type of run header
fileTypeID	flag for cmp or rec file
dataType	data type - All, Sun/Moon, Mrk501, ...
blocked	flag for whether data is written in blocks
recFlag	flag for whether data is of rec format
cmpFlag	flag for whether data is of cmp format
YEAR	the year the data was taken
nEvents	not used
badEvt	not used
buffSynch_errors	not used
buffSkip_errors	not used
fbHeader_errors	not used
gpsBCD_errors	not used
wordCount_errors	not used
gpsSeq_errors	not used
gpsMin_errors	not used
gpsHour_errors	not used
gpsDay_errors	not used
fsccSocket_errors	not used
arrayBounds_errors	not used
bytesOut_errors	not used
dontFit	flag for whether a PMT should be used in fitters (not used)
nToss	not currently used
i	i grid coordinate of PMT
j	j grid coordinate of PMT
k	k grid coordinate of PMT
ijkmap	map of ijk coordinates of PMT
mc2data	conversion of MC grid coordinates to actual grid coordinates
data2mc	conversion of actual grid coordinates to MC grid coordinates
fbAdd	not used for Milagro
tdcChan	TDC channel of PMT
tdcMap	not used for Milagro
x	x position (cm) of PMT
y	y position (cm) of PMT
z	z position (cm) of PMT
xxi	x position $\times$ x position (cm^2) of PMT
yyi	y position $\times$ y position (cm^2) of PMT
zzi	z position $\times$ z position (cm^2) of PMT
xzi	x position $\times$ z position (cm^2) of PMT
yzi	y position $\times$ z position (cm^2) of PMT
xyi	x position $\times$ y position (cm^2) of PMT
nhit	total number of PMTs hit (air-shower + muon + outrigger)

igrid	grid number of phototube
rawLatch	raw bits as read from latch (only first 2 words used)
julDate	(Milagro) modified julian date = MJD - 50000
eventTime	event time (seconds from noon UT)
timeErr	integer which describes time error (0 = no time error)
adc	adc measure of number of photoelectrons
nedge	number of edges for a PMT
phase_nedge	phase of each edge
arrival_time_nedge	arrival time of each edge

eventNumber	not filled (always = 0)
nPMTs_cal	total number of PMTs calibrated (air-shower + muon + outrigger)
nAS_cal	total number of air-shower PMTs calibrated
nMUON_cal	total number of muon PMTs calibrated
nOut_cal	total number of outriggers calibrated
igrid	grid number of phototube
pes	number of photoelectrons
rawLatch	raw bits as read from latch (only first 2 words used)
PMT_time	arrival time (seconds) of PMT
julDate	(Milagro) modified julian date = MJD - 50000
eventTime	event time (seconds from noon UT)
timeErr	integer which describes time error (0 = no time error)
nPMTs	total number of PMTs hit (air-shower + muon + outrigger)
goodEdge	number of good edges (2 or 4)
startEdge	which edge is the starting edge

theta_fit	fitted theta (degrees) using both layers
phi_fit	fitted phi (degrees) using both layers
theta_fit_as	fitted theta (degrees) using air-shower layer
phi_fit_as	fitted phi (degrees) using air-shower layer
theta_fit_mu	fitted theta (degrees) using muon layer
phi_fit_mu	fitted phi (degrees) using muon layer
nfit	number of PMTs used in angle fit (both layers)
nfit_as	number of PMTs used in angle fit (air-shower layer)
nfit_mu	number of PMTs used in angle fit (muon layer)
ra_fit	fitted ra (degrees - 2000) from angle fit (both layers)
dec_fit	fitted dec (degrees - 2000) from angle fit (both layers)
ra_fit_as	fitted ra (degrees - 2000) from angle fit (air-shower layer)
dec_fit_as	fitted dec (degrees - 2000) from angle fit (air-shower layer)
ra_fit_mu	fitted ra (degrees - 2000) from angle fit (air-shower layer)
dec_fit_mu	fitted dec (degrees - 2000) from angle fit (muon layer)
weighted_sum	weighted sum of PMTs used in the angle fit
xcore_fit	x position (cm) of fitted core (both layers)
xcore_fit_as	x position (cm) of fitted core (air-shower layer)
xcore_fit_mu	xposition (cm) of fitted core (muon layer)

ycore_fit	y position (cm) of fitted core (both layers)
ycore_fit_as	y position (cm) of fitted core (air-shower layer)
ycore_fit_mu	y position (cm) of fitted core (muon layer)
errCore	estimated error (cm) in fitted core position
coreDist	distance (cm) of each PMT from fitted core
tChis	T-chi (ns) of PMT
weights	should be weight of each tube but isn't used
TOTlo	low TOT (counts) for each PMT
TOThi	hi TOT (counts) for each PMT
Used	flag for whether a PMT is used in the angle fit (1 = used)
WhyTossed	not filled (always = 0)
nNoEdge	numbers of PMTs that failed the edge finder
nNoSlew	number of PMTs with no slewing
nNoPETOT	number of PMTs with no TOT to PE calibration
nNoSlewOOB	numbers of PMTs with no slewing (slweing out of bounds)
nMuon	number of muons hitting pond
muonPes	number of PEs from each muon
ambPes	ave. number of PEs from PMTs surrounding hotest PMT
muonX	x position (cm) of fitted muon
muonY	y position (cm) of fitted muon
maxPEmuon	PMT with highest PE value in the muon layer
maxPEmuon_satcor	saturation corrected maxPEmuon
gh_GBY	(float)nMUON_cal/maxPEmuon
gh_GBY_satcorr	(float)nMUON_cal/maxPEmuon_satcorr

All processed Milagro data is located at Maryland in /data02/data/milagro/rec. Runs 1150 - 2054 were processed with (the equivalent of) version 32 of the offline code. The following parameters were used:

```
dataCon->source=DATA_MGRORAW3;
calCon->EdgeMethod = MILAGRO_EDGEFINDER;
calCon->PeMethod = ADC_TOTPE;
calCon->SlewMethod = TIMING_INITIAL;
coreCon->method = CORE_FIT_CENTER_OF_MASS;
angleCon->curvature = 0.045;
angleCon->sampling_method = SAMPLE_MC_2;
angleCon->method = ANGLE_FIT_CHISQ_FLATTOP;
angleCon->PE_SIG_cuts_method = CUTS1;
angleCon->weight_method = WEIGHT1;
angleCon->relax_method = RELAX1;
angleCon->early_light_method = EARLY_LIGHT1;
```

Runs 2056 on were processed with (the equivalent of) version 41 of the offline code. The following parameters were (are) used:

```
dataCon->source=DATA_MGRORAW3;
```

```
calCon->EdgeMethod = MILAGRO_EDGEFINDER2;
calCon->PeMethod = SPEC_TOTPE;
calCon->SlewMethod = TIMING_LASER3;
coreCon->method = CORE_FIT_CENTER_OF_MASS;
angleCon->curvature = 0.045;
angleCon->sampling_method = SAMPLE_MC_2;
angleCon->method = ANGLE_FIT_CHISQ_FLATTOP;
angleCon->PE_SIG_cuts_method = CUTS1;
angleCon->weight_method = WEIGHT1;
angleCon->relax_method = RELAX1;
angleCon->early_light_method = EARLY_LIGHT1;
```