What else bugs me about Milagro Software?

Our software sucks.

Reconstruction.h is indecipherable, and the documentation is out of touch with the code.

There are many empty/unused structures.

There is no naming convention for data structures or functions.

The structures are hopelessly intertwined.

Loads of ugly code.

Gratuitous Ugliness

```
/* Event data */
typedef struct
{
    RAW_EVENT      rawEvent;
    RAW_EVENT      cmpEvent;
    unsigned int *cmpPtr;
    CAL_ALL        calEvent;
    REC_DATA       prcEvent;
    REC_DATA       recEvent;
    ERR_STAT_INFO errInfo;
    SUB_RUN_HDR    subRunInfo;
    SUB_HEADER     subRunInfo2;
} EVENT_DATA;
```

Gratuitous Ugliness

```
int closeEdge(int ichan, int e1, int e2, RAW_EVENT *rawInfo, double tzero){
  return (fabs(TIME2(ichan,e1)-tzero) < fabs(TIME2(ichan,e2)-tzero));
}
```

TIME2 () is a macro:

```
#define TIME2(x,y)      ((double)(rawInfo->tdc.cnts[x][y]))
```

Milinda – King of Bactria

Alexander the Great invaded India in 326 BC and after his death the territories he had conquered in what today is Afghanistan, Pakistan and north-west India were ruled by his heirs. The greatest of the Greek Indian Kings was Milinda, known in Greek sources as Menander, who ruled from about 163 to 150 BC. Menander was both an effective statesman and fine soldier and he greatly extended his empire into India. He was converted to Buddhism by monk Nagasena.

http://www.buddhanet.net

Milinda – Coding Standards

Milagro software is more a product of evolution than design:

- Empty unused structures.
- Some data in 2 places (time)
- Data missing (x2 for example)
- The strange roll of PMT_INFO
- Awkward steering/control
- Mishmash of coding styles
- Macros – Yucko
- Disjoint naming scheme: INFO == DATA == RECON

Milinda – Coding Standards

Structure have descriptive names:

- CAL_DATA
- EVENT_DATA
- EVENT

Instantiation of structures:

- CalData
- EventData
- Event

Header Files:

- CalData.h
- EventData.h
- Event.h

Milinda – Coding Standards

All functions only take one structure as an argument!
-> Object Oriented, but not C++

All data is owned by the main. Same as the current paradigm.

Loads of new structures and variable names:

TimeData.h <-- Replaces old Time Structure.
Includes TJD,MJD,JD, and double versions
HitList.h <-- Makes uniform the nHits/GridNo[] interface
GHData.h <-- Actually has x0,x2, and compactness

Loads of extras.

Milinda – Include Files

Reconstruction.h becomes:

```
AnalysisControl.h
AnalysisData.h
AngleControl.h
AngleDataChiSq.h
AngleDataChiSqResults.h
AngleData.h
AngleDataSummary.h
CalControl.h
CalData.h
CalibData.h
CalStatus.h
CalTable.h
CmpBlockHeader.h
CmpDataBlockHeader.h
Control.h
CoreControl.h
CoreDataCom.h
CoreData.h
CoreDataOff.h
CoreDataOrcom.h
CoreDataSummary.h
DataControl.h
EdgeData.h
ErrorData.h
ErrStatInfo.h
EventData.h
Event.h
EventSummary.h
GHControl.h
GHDataFFT.h
GHData.h
GHDataMuon.h
GHDataStat.h
GHDataSummary.h
GridCoord.h
HitList.h
MCData.h
MilagroConstants.h
Milinda.h
MuonDataMC.h
PEData.h
PMTData.h
RawADC.h
RawData.h
RawTDC.h
RawVise.h
RecBlockHeader.h
RecDataCmp.h
RecData.h
Reco.h
SlewData.h
StandardInc.h
Steer.h
SubRunHeader.h
TimeData.h
TrigData.h
```

Milinda – Include Files

```
// Reco.h

// Reconstruction container with event and function steering.
// Here is everything that you need to do reconstruction
// and analysis. Missing is the code that steers the framework.

#ifndef RECO_H
#define RECO_H

#include "Event.h"
#include "Control.h"
#include "GridCoord.h"
#include "PMTData.h"

typedef struct {
  EVENT          Event;          // Data specifc to event
  CONTROL        Control;        // Steering Routines for Reconstruction
  GRID_COORD     GridCoord;      // Coordinate Structure
  PMT_DATA       PMTData;        // Data not specific to a single event
                                 // includes DontFits
} RECO;

// function prototypes

int SimpleSensi( RECO * );
int Monitor( RECO * );

// Standard Milagro data reading routine for all data types
int MilagroEvent4( RECO * );

#endif
```

Milinda Structure

```
Typedef struct {
  RECO     Reco;
  STEER    Steer;
} MILINDA;
```

```
Typedef struct {
  EVENT          Event;        // Data specifc to event
  CONTROL        Control;      // Steering Routines for Reconstruction
  GRID_COORD     GridCoord;    // Coordinate Structure
  PMT_DATA       PMTData;      // Data about PMT configuration
                               // includes DontFits
} RECO;
```

```
typedef struct{
  int CurrentStage;    /* Current Stage of Reconstruction: init=0, 1-3 processing, 99=end  */
  int Continue;        /* User Control Structure, 1=continue to next level, 0=next event   */
  int RepeatEvent;     /* Repeat Calibration/Reconstruction cycle for current event        */
  int TerminateLoop;   /* Terminate Program, call finalize routine.                        */
  int LogRecEvent;     /* set to 1 if Reconstructed event should be written to a file      */
  int LogRawEvent;     /* set to 1 if Raw event should be written to an output file        */
  int DisplayEvent;    /* Call event Display for this event                                */
  int SpecialArgs;     /* Arguments passed from command line indicate filename and Nevents */
  int Verbose;         /* If 1 print statistics and stuff                                  */
  int ComputeSensi;    /* Compute point source sensitivity                                 */
  int MilagroEventStatus;    /* Status variable returned by MilagroEvent4                  */
  int StageCount;      /* Number of stages in Event Loop                                   */
  int RecEventsLogged; /* =0 if no Rec Events were logged, =1 if Rec Events were Logged    */
  int argc;            /* number of parameters passwd from command line                    */
  char *argv[40];      /* command line parameters; GNUstep wants the array size specified  */
  // Modules that can be turned on and off
  int MonitorActive;   /* Turn on Monitor                                                  */
  int CoreFitORCOMActive;     // ORCOM core fitter
  int CoreFitCOMActive;       // COM core fitter
  int CoreFitOFFActive;       // OFF core fitter
} STEER;
```

Reco Structure

```
typedef struct {
  EVENT           Event;
  CONTROL         Control;
  GRID_COORD      GridCoord;
  PMT_DATA        PMTData;
} RECO;
```

```
Typedef struct {
  EVENT_DATA          EventData;          // Raw Event as read from disk
  CAL_DATA            CalData;            // Calibrated Event derived offline
  ANGLE_DATA          AngleData;          // Results of Angle Fit(s)
  CORE_DATA           CoreData;           // Results of Core Fit(s)
  GH_DATA             GHData;             // Results of Gamma/Hadron sep. code
  REC_DATA            RecData;            // Reconstructed Data Results
  ANALYSIS_DATA       AnalysisData;       // Analysis Data specific to this event
} EVENT;
```

```
typedef struct {
  DATA_CONTROL       DataControl;
  CAL_CONTROL        CalControl;
  CORE_CONTROL       CoreControl;
  ANGLE_CONTROL      AngleControl;
  GH_CONTROL         GHControl;
  ANALYSIS_CONTROL   AnalysisControl;
} CONTROL;
```

Event Structure

```
typedef struct {
    RAW_DATA          RawData;            // Compressed Raw Event
    CAL_DATA          CalData;            // Calibrated Event
    REC_DATA          RecData;            // The contents the reconstruction results
    // ---- These are filled by all data types ----
    SUB_RUN_HEADER    SubRunHeader;       // subrun header
    ERROR_STAT_DATA   ErrorStatData;      // An error structure different than ERROR_DATA
    MC_DATA           MCData;             // The true numbers from the MC
    TIME_DATA         TimeData;           // GPS Clock time data
    TRIG_DATA         TrigData;           // VME Trigger information
    CALIB_DATA        CalibData;          // Calibration Data encoded in the event.
    REC_BLOCK_HEADER  RecBlockHeader;     // The rec block header
    CMP_BLOCK_HEADER  CmpBlockHeader;     // The compressed raw data block header
} EVENT_DATA;
```

```
Typedef struct {
  EVENT_DATA          EventData;
  CAL_DATA            CalData;
  ANGLE_DATA          AngleData;
  CORE_DATA           CoreData;
  GH_DATA             GHData;
  REC_DATA            RecData;
  ANALYSIS_DATA       AnalysisData;
} EVENT;
```

EventData Structure

```
typedef struct {
    RAW_DATA          RawData;
    CAL_DATA          CalData;
    REC_DATA          RecData;
    //-------------------------
    SUB_RUN_HEADER    SubRunHeader;
    ERROR_STAT_DATA   ErrorStatData;
    MC_DATA           MCData;
    TIME_DATA         TimeData;
    TRIG_DATA         TrigData;
    CALIB_DATA        CalibData;
    REC_BLOCK_HEADER  RecBlockHeader;
    CMP_BLOCK_HEADER  CmpBlockHeader;
} EVENT_DATA;
```

Loads of Stuff/Decisions?

```
typedef struct {
    RAW_DATA          RawData;            // Compressed Raw Event
    CAL_DATA          CalData;            // Calibrated Event
    REC_DATA          RecData;            // The contents the reconstruction results
    // ---- These are filled by all data types ----
    SUB_RUN_HEADER    SubRunHeader;       // subrun header
    ERROR_STAT_DATA   ErrorStatData;      // An error structure different than ERROR_DATA
    MC_DATA           MCData;             // The true numbers from the MC
    TIME_DATA         TimeData;           // GPS Clock time data
    TRIG_DATA         TrigData;           // VME Trigger information
    CALIB_DATA        CalibData;          // Calibration Data encoded in the event.
    REC_BLOCK_HEADER  RecBlockHeader;     // The rec block header
    CMP_BLOCK_HEADER  CmpBlockHeader;     // The compressed raw data block header
} EVENT_DATA;
```

```
Typedef struct {
  EVENT_DATA          EventData;
  CAL_DATA            CalData;
  ANGLE_DATA          AngleData;
  CORE_DATA           CoreData;
  GH_DATA             GHData;
  REC_DATA            RecData;
  ANALYSIS_DATA       AnalysisData;
} EVENT;
```