

Emt Calibration Packages
(L1EmtCalOnline & L1EmtCalOdf)

S. Ricciardi

November 18, 2001

The document presented here is my present understanding of these packages. It is far from being complete and might contain errors. It can be completed and corrected soon after this workshop.

Overview

These packages are built within the online software environment.

A general description of **L1EmtCalOnline** is in the README file:

00001

00002 —————> L1EmtCalOnline <—————

00003

00004 This package contains the level 1 calorimeter trigger calibration software

00005 which is used in both the online (Odf) world and the L3 trigger. It is an

00006 online/offline dual-use package.

00007

00008 26/04/99 P.Dauncey Set up package

NO README file for **L1EmtCalOdf**.

A Doxygen documentation of all classes has been created and is (temporary) in:

www.slac.stanford.edu/ricciars/trigger/CalOdf

www.slac.stanford.edu/ricciars/trigger/CalOnline

L1EmtCalOnline

The latest tag is: V00-00-08

The present package Coordinator is: P.Dauncey

L1EmtCalOnline is a dual-use package (online and offline)

It contains the code to produce

- CalCycleTC files (offline use)
- code for storing the data collected from the Frame Clash Calibration (online use)
- and code for defining the type of configuration for the calibration to be run (dual use) - *L1EmtXXXCycle classes*

A *CalCycleTC* is a tagged container which defines how many iterations are going to be performed at different level of the finite state machine (FSM), selects which actions are attached to the FSM and sets bits in the control, status and error register (CSER, a 6-byte register which contains a summary of the TPB status.)

The user has to put this cycleTC into the database and attach it to a key in order to use it. The procedure is described in Jason's online help:

<http://www.slac.stanford.edu/BFROOT/www/Detector/Trigger/emt/software/emtOnline.html>

link_L1EmtCalOnline.mk bin_L1EmtCalOnline.mk Makefile GNUmakefile	makefiles for creating libraries compiling and linking the executables both for the online and offline releases
emtBepbCycles.cc emtDataCycles.cc emtFClashCycles.cc emtFcxCalCycles.cc emtFepbCycles1.cc emtFepbCycles2.cc emtFepbCycles3.cc emtPhiCalCycles.cc emtTCCycles.cc emtTowerCycles.cc L1EmtOpSaApp.cc	builds CalCycleTCs for the EMT Back Playback calibrations physics data taking Frame Clash calibration EMT Fast Control Xilinx calibrations builds CalCycleTCs for the EMT Front Playback calibrations idem idem builds CalCycleTCs for the EMT phi calibration TC output calibration ?? serial number calibration ?
L1EmtCalOnlineTemplates.cc L1EmtCalOnlineTypeInit.cc L1EmtCalTypeIds.hh	creates an identification number for the CycleTC “ “
L1EmtConfigCycle.cc/.hh L1EmtMetaCycle.cc/.hh L1EmtMacroCycle.cc/.hh L1EmtMajorCycle.cc/.hh L1EmtMinorCycle.cc/.hh L1EmtOpSaMinorCycle.cc/.hh	does only OpSa depend on this?
L1EmtFCErrors.cc/.hh L1EmtFCSafeChan.cc/.hh L1EmtFCScatChan.cc/.hh	Frame clash calibration classes

Table 1: L1EmtCalOnline File list

You will notice there are three different applications corresponding to three different types FEPB runs. The difference is only in the choice of CSER (fe, fec, fecc for FepbCycle1,2,3 respectively)

Dependencies

Figure 1 shows the graphical class hierarchy.

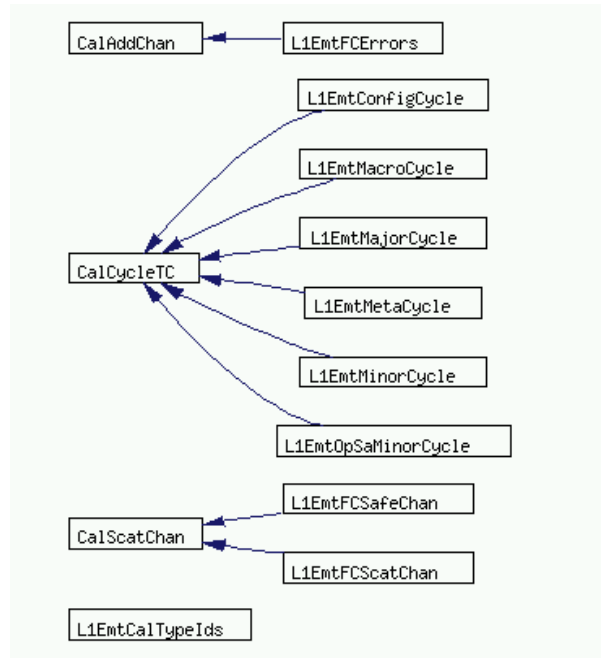


Figure 1: CalOnline class hierarchy

Classes in this package inherit from the following packages (in brackets the inherited classes):

- BbrCalib (CalAddChan)
- CalOnline (CalCycleTC, CalSatChan)

CalOnline holds the online specific calibration code common to all subsystems.

BbrCalib puts together the tools and base classes for online interactions with calibration information.

It also depends on the following packages/classes:

- L1EmtOnline;
- OdfContainer & OdfContainerTools

Different types of calibration

Calibration in the EMT world means dedicated diagnostic runs, or runs to check the configuration constants. Note that also normal data-taking is treated as a special type of calibration run from the software point of view.

Tower, FCX and Phi Calibration all change the tower input mask and then collect data.

Tower calibration allows to test the physical connection with EMC using either the set serial number pattern output of the EMC UPC's or system noise when reading directly from the front-end electronics.

FCX calibration is meant to monitor the system response to a change in the `synchDelay` variable within the FCX configuration data to ensure that the EMT latency budget was used as efficiently as possible.

Phi calibration is used to check the connection between EMT and GLT. The tower mask is changed, performing a complete rotation of the calorimeter phi sector and the output of the GLT should match the input mask loaded.

FrameClash calibration is used to synchronize the EMC and the EMT system interconnection. The EMC frame offset is altered to see when it clashes with the TPB framebit.

FEPB/BEPB calibration are described in BABAR note 519 Front End PlayBack can check exactly the same TPB functionality as for the normal running case, but does not need and cannot occur during a run in progress. FC configuration data are feed into front End memories instead of data from UPC. In Back End PB run data are instead sent down the Global output line, bypassing almost all the TPB logic

There are also **TB & TC** calibrations, which I believe stand for Tagged Container and Test Board, whose function are not clear to me yet.

Application to generate CalCycleTCs

An application is needed to generate CalCycleTC for each of the different type of calibration. Note there is no source code for the BOARD TEST.

Initially an ODFArena, i.e. a memory space allocated for the complete CycleTC tree, is created.

A TopTC is then created, on which the subsystem specific CycleTC's are appended. The Config level TC is then constructed and appended to TopTC. Similar for Meta, Macro, Major and Minor which are created in nested loops and appended in cascade. Within the MetaCycle is a variable determining the number of Major cycles and so on.

The sequence can be seen in the diagram of the final state machine.

The CalChoice, CserChoice variables are constructor arguments of the Meta level creation to select the type of calibration, and therefore the actions to be performed later and write the bit in CSER.

For instance, this is for normal data taking in emtDataCycles.cc:

```
00080 L1EmtMetaCycle::L1EmtCalChoice metacal=L1EmtMetaCycle::fex;
00081 L1EmtMetaCycle::L1EmtCserChoice metacser=L1EmtMetaCycle::norm;
00082 L1EmtMetaCycle* metaTC = L1EmtMetaCycle::createMetaCycle
00083 (myarena,nmacro,
00084 (CalCycleFlag::cycleTransmission)metatrans,metacal,metacser);
```

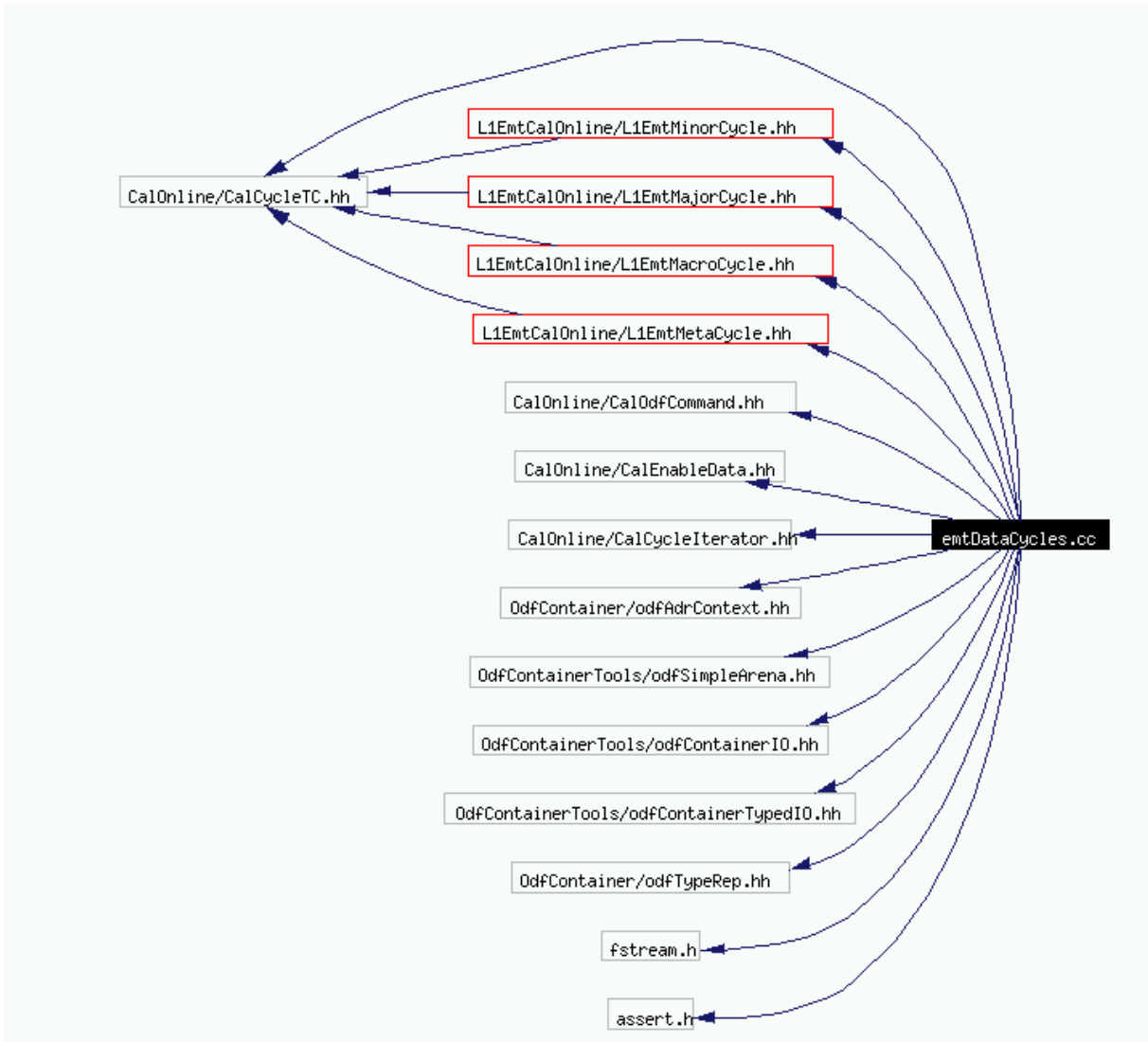


Figure 2: emtDataCycles dependency graph

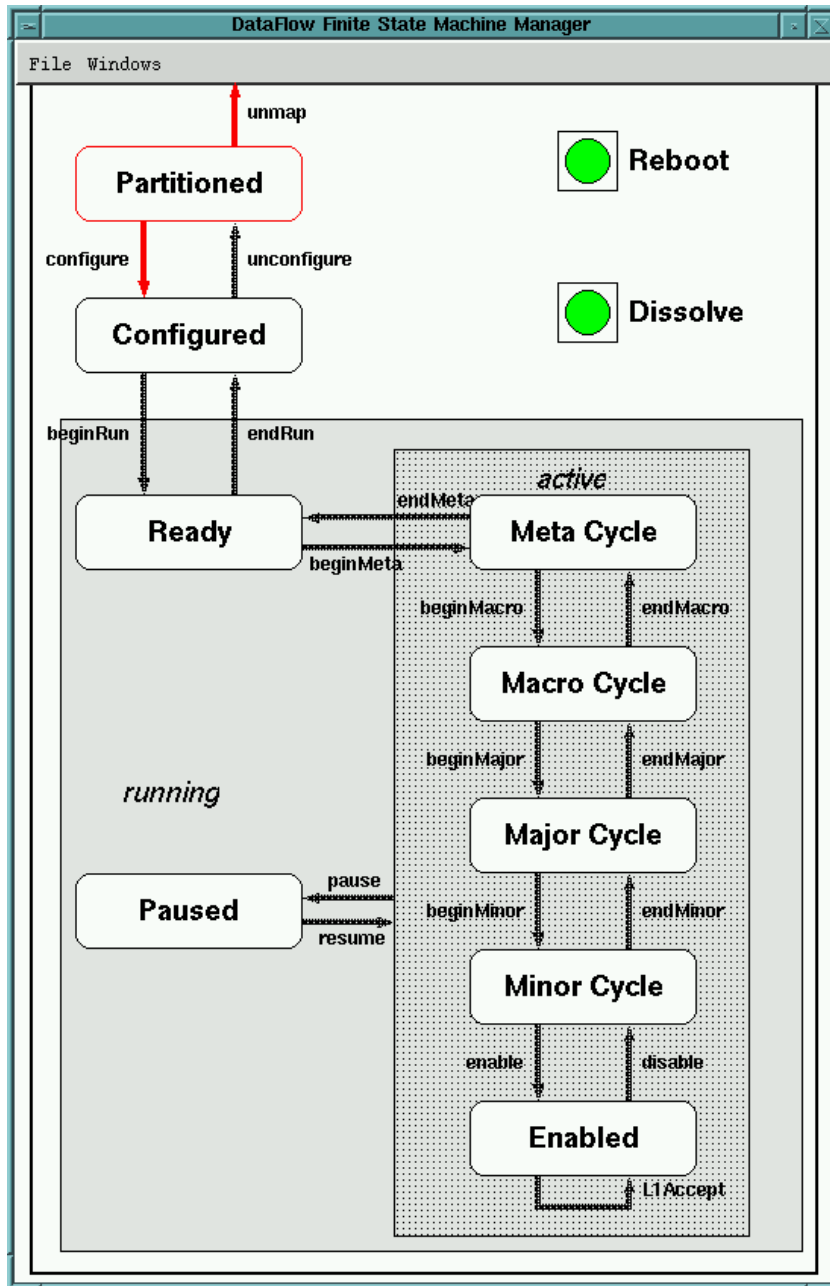


Figure 3: Finite State machine diagram

An informative text can be found in L1EmtMetaCycle.hh and it is reported here

```
00077 //
00078 //
00079 // informative text
00080 //
00081 static const char* calName(L1EmtCalChoice level) {
00082     switch(level) {
00083     case frameclash:
00084         return "Frameclash Calibration";
00085     case phi:
00086         return "Phi Calibration";
00087     case tower:
00088         return "Tower Calibration";
00089     case fcx:
00090         return "Fcx Calibration";
00091     case fepb:
00092         return "Front Playback Calibration";
00093     case bepb:
00094         return "Backend Playback Calibration";
00095     case tc:
00096         return "Tagged Container Calibration";
00097     case boardtest:
00098         return "Boardtest Calibration";
00099     case newframeclash:
00100         return "New FrameClash Construction";
00101     case fex:
00102         return "Data Taking Feature Extraction Mode";
00103     default:
00104         return "Unknown";
00105     }
00106 }
00107
00108
00109 static const char* cserName(L1EmtCserChoice cser)
00110 {
00111     switch(cser) {
00112     case rec:
00113         return "Configurec";
00114     case fe:
00115         return "Configfe";
00116     case fec:
00117         return "Configfec";
00118     case fecc:
00119         return "Configfecc";
00120     case be:
00121         return "Configbe";
00122     case bec:
00123         return "Configbec";
00124     case lat:
00125         return "Configlat";
00126     case norm:
00127         return "Configure";
00128     default:
00129         return "Default Configure";
00130     }
00131 }
```

Figure 4: Possible values for Calchoice and CSERchoice.

In L1EmtCalOdf/L1EmtCalConfig.cc you can find a description of the different CSER setup:

```
case L1EmtMetaCycle::rec:
// Configure playback off, spystart always, stop on full
control=Configurec();
...
case L1EmtMetaCycle::fe:
// Configure spy/playback for frontend playback oneshot run. // Spy start on StartPb, stop on
full
control=Configfe();
...
case L1EmtMetaCycle::fec:
// Configure spy/playback for frontend playback continuous
// Spy start always, stop on full
control=Configfec();
...
case L1EmtMetaCycle::fecc:
// Configure spy/playback for frontend playback cont run.
// Spy start on StartPb, stop on full
control=Configfecc();
...
case L1EmtMetaCycle::be:
// Configure spy/playback for backend playback single shot
control=Configbe();
...
case L1EmtMetaCycle::bec:
// Configure spy/playback for backend playback continuous
control=Configbec();
...
case L1EmtMetaCycle::lat:
// Configuration for testing the latency of boards using spy continuous stop on L1Accept
control=Configlat();
case L1EmtMetaCycle::norm:
// Normal configure
// Spy start always, stop on full (Actually SPY is never started, I believe)
control=Configure();
```

L1EmtCalOdf

The Latest tag: V00-04-02

The Present Package Coordinator: P. Clark

L1EmtCalOdf is a pure online package.

It contains all the finite state machine actions for all the different modes of operation of the system for data collection, calibration and test runs.

Note that from the point of view of the software, data-taking is a special kind of “calibration”, i.e. the actions to be taken during a physics run are defined during configuration through a “dummy” CalCycleTC file (with nmajor=nmacro=nminor=1)

Dependencies

This Package depends directly on the following packages:

- Specific EMT packages
 - L1EmtOnline
 - L1EmtOdf
 - L1EmtCalOnline
- Babar General Calibration packages:
 - CalOdf
 - CalOnline
 - BbrCalib
- DataFlow packages:
 - Rdf
 - Odf
 - OdfContainer & OdfContainerTools

L1EmtBase

Many action classes within package share the functions defined here.

```
Public Methods
L1EmtBase ()
virtual ~L1EmtBase ()
void setVerbose (unsigned)
void resetFps ()
virtual void setup ()

Protected Methods
int allWrite ()
int axWrite ()
int fxWrite ()
int fxWrite ()
int eserWrite ()
int frontPhWrite ()
int backPhWrite ()
void allRead ()
unsigned eserRead ()
unsigned axRead ()
unsigned fxRead ()
unsigned fxRead ()
unsigned frontPhRead ()
unsigned backPhRead ()
void runNode ()
void configureNode ()
void startPlayback ()
int CserSet (unsigned short control)
int AllSet (unsigned short control)
int AxSet (const char *fileName, unsigned axMask, unsigned short mask)
int FrontPhSet (const char *fileName, unsigned short mask)
int OffsetSet (const char *fileName, unsigned short mask)
int BackPhSet (const char *fileName, unsigned short mask)
int FxSet (const char *fileName, unsigned short mask)
int FocSet (const char *fileName, unsigned short mask)
int AllTCSet (const L1EmtFocTC *foctc, const L1EmtFocTC *foc, const L1EmtAxtC *axtc, unsigned short control)
int FocTCSet (const L1EmtFocTC *foctc, unsigned short mask)
int FxTCSet (const L1EmtFocTC *foc, unsigned short mask)
int AxtCSet (const L1EmtAxtC *axtc, unsigned short axMask, unsigned short mask, unsigned short channel)
int AxReadMaskFromFlatFile (const char *fileName)
int AxSave (unsigned slot, unsigned ax, const char *fileName)
int FrontPhSave (unsigned slot, const char *fileName)
int OffsetSave (unsigned slot, const char *fileName)
int BackPhSave (unsigned slot, const char *fileName)
int FxSave (unsigned slot, const char *fileName)
int FocSave (unsigned slot, const char *fileName)
unsigned short Configure ()
unsigned short Configuree ()
unsigned short Configfe ()
unsigned short Configfee ()
unsigned short Configfeece ()
unsigned short Configfeece ()
unsigned short Configfeece ()
unsigned short Configfeece ()
```

Figure 5: Methods in class L1EmtBase

It includes:

- functions for reading and writing the configuration from/to the boards to/from a transient class held in the ROM memory space, *xxxRead / xxxWrite*;
- function for reading and writing the configuration from/to an external storage to/from a transient class held in the ROM memory space, *xxxSet or xxxTCSet/ xxxSave*. The configuration can be read either from a flat file or from a Tagged Container;
- functions to setup the CSER, *Configxxx*; As example, Configure is the one for data-taking, where
 - Playback enable is set to playbackOff
 - SpyStart set to startOnStartPlayback
 - SpyStop set to stopOnFull
- functions to change the system state from configure to run or start PlayBack functions *runMode, configureMode, startPlayback*.

Common classes to all calibration runs

TestEmtCycles.cc L1EmtCalSegTest.cc	Simple test program for test CalCycle classes Application source code for FSM and action construction for Emt data-taking/calibration
L1EmtRomBoot.cmd L1EmtSegTest.cmd	Loads on the ROM (not existing) current libraries. Not in use. Loads libraries wallom/2.2.6 . Not in use.
L1EmtBase.cc L1EmtBase.hh	Emt basic functions - inherited by most of the classes in this package
L1EmtReset.hh/cc L1EmtCalConfig.hh/.cc L1EmtBeginRun.hh/.cc L1EmtBeginMeta.hh/.cc L1EmtEndMeta.hh/.cc	Finite state machine actions of which are independent the calibration type chosen are in this group
L1EmtCalOdfTemplates.cc	Template instantiation for calibration classes

Some fundamental actions have to be performed by all type of calibration runs.

The classes corresponding to these actions are:

- L1EmtReset - on reboot of the EMT ROM, the FSM is initialized, then a reset action occurs. In the *fire* method of this class a *resetFee* is executed to clear the event, playback and spy buffers of any spurious data. The system is then explicitly set to Configure mode. The CSER is filled with default values hard coded in a CSERRawData class. All error flags are cleared. C/D-link connections are verified by checking the board serial number.
- L1EmtCalConfig - here the configuration constants for the 3 Xilinx's register are loaded as well as the EMT top map (i.e. also a CalCycleTC? where? how?)

The CSERChoice variable is also read from the cycleTC in memory space and used to determine which CSER to load. Then in

the globalconfig method of this class these actions are performed

```
00173 //AllTCSet is a stripped down version of AllSet that uses flat
00174 //file transfer for the playback files & the Mask. This is done
00175 //since the procedure to load the TC's into the database is
00176 //excessivly complicated and the mask changes too frequently
00177 //for this to be practical.
00178 AllTCSet(fcxtc,fxtc,axtc,control);
00179
00180 //AllSet(control);
00181 allWrite();
00182 // Need to flag configuration errors.....
```

Here I am completely confused by the comment, I thought these TC were indeed loaded from the database on the contrary of the flat files which are on disk...

- L1EmtBeginRun - This action is used to validate the data that was downloaded onto the TPB's. This is implemented in the *fire method of this class by AllRead*. The contents are checked against the software model for error during download of constants. Why should it be done here? Could it be done in the transition to configure the system as last action?
- L1EmtBeginMeta - Here the decision on which type of calibration will be run is made and the appropriate next transition is added to FSM.
- L1EmtEndMeta - This class is included to match BeginMeta.

Specific calibration classes

Classes associated to FC calibration are listed in the table below.

There are two kind of FC calibrations. The “new” one has only actions associated to Macro and Minor, L1Accept transitions (no Major). Here work in progress to understand the difference between these 2 calibrations.

Good source of information are D. Wallom’s and P. Mc Grath’s PHD thesis.

L1EmtFCBeginMacro.hh/.cc	Frame clash
L1EmtFCBeginMajor.hh/.cc	
L1EmtFCBeginMinor.hh/.cc	
L1EmtFCData.hh/.cc	
L1EmtFCEndMacro.hh/.cc	
L1EmtFCEndMajor.hh/.cc	
L1EmtFCEndMinor.hh/.cc	
L1EmtFCIterator.hh/.cc	
L1EmtFCL1Accept.hh/.cc	
L1EmtNewFCBeginMacro.hh/.cc	
L1EmtNewFCBeginMinor.hh/.cc	
L1EmtNewFCEndMacro.hh/.cc	
L1EmtNewFCEndMinor.hh/.cc	
L1EmtNewFCIterator.hh/.cc	
L1EmtNewFCL1Accept.hh/.cc	
L1EmtL1AcceptIterator.hh	Iterator class for L1 accept used only in FC

To be done

- Improve comments, especially in the header the description of classes are often misleading.
- Clean up:
 - leave only one FC calibration? Which one is in use?
 - Are these xxOpSaxxx classes needed? If not Remove them.
 - Remove old cmd files, and dual files *testEmtCycles.cc* from L1EmtCalOdf.
- (For me) Complete understanding of the code.
- Request for Synch problem, means we have to remove all actions from BeginRun in data-taking?
- Other ideas?

L1EmtBEPBBeginMacro.hh/.cc L1EmtBEPBBeginMinor.hh/.cc L1EmtBEPBEndMacro.hh/.cc L1EmtBEPBEndMinor.hh/.cc L1EmtBEPBL1Accept.hh/.cc	Actions for the Back-end Play Back calibration
L1EmtBTBeginMacro.hh/.cc L1EmtBTBeginMajor.hh/.cc L1EmtBTBeginMinor.hh/.cc L1EmtBTEndMacro.hh/.cc L1EmtBTEndMajor.hh/.cc L1EmtBTEndMinor.hh/.cc	Actions for Board Test
L1EmtFCXCALBeginMacro.hh/.cc L1EmtFCXCALBeginMinor.hh/.cc L1EmtFCXCALEndMacro.hh/.cc L1EmtFCXCALEndMinor.hh/.cc	Fast control Xilinx
L1EmtFEPBBeginMacro.hh/.cc L1EmtFEPBBeginMinor.hh/.cc L1EmtFEPBEndMacro.hh/.cc L1EmtFEPBEndMinor.hh/.cc L1EmtFEPBL1Accept.hh/.cc	Front End Play Back
L1EmtPHICALBeginMacro.hh/.cc L1EmtPHICALBeginMinor.hh/.cc L1EmtPHICALEndMacro.hh/.cc L1EmtPHICALEndMinor.hh/.cc	Phi Calibration
L1EmtTCALBeginMacro.hh/.cc L1EmtTCALBeginMinor.hh/.cc L1EmtTCALEndMacro.hh/.cc L1EmtTCALEndMinor.hh/.cc	Tower Calibration
L1EmtTCBeginMacro.hh/.cc L1EmtTCBeginMinor.hh/.cc L1EmtTCEndMacro.hh/.cc L1EmtTCEndMinor.hh/.cc L1EmtTCL1Accept.hh/.cc	TC test
L1EmtFEXBeginMacro.hh/.cc L1EmtFEXBeginMinor.hh/.cc L1EmtFEXEndMacro.hh/.cc L1EmtFEXEndMinor.hh/.cc L1EmtFEXL1Accept.hh/.cc	Feature Extraction This is used only in data taking mode