L1CalBuffer now supports methods:

```
virtual void beginHeader(const L1CaloModuleType &, int index) ;
virtual void endHeader(const L1CaloModuleType &, int index) ;

virtual void beginHeader(const L1CaloSubModuleType &, int index) ;
virtual void endHeader(const L1CaloSubModuleType &, int index) ;

virtual void makeRodTrailer(std::list <Word32> &statusVector) ;
virtual void makeRodHeader(Word32 sourceId, Word32 runNumber,
Word32 sequenceNumber, Word32 sequenceType)  ;

virtual void setCompatibilityMode(const bool &)  ;
```

SimpleBuffer is the implementation of L1CalBuffer, and is still backward compatible to the "old" implementation. FormattedBuffer is a SimpleBuffer with the compatibilityMode set to false.

The paradigm is that the run-controller in a crate, or a special universal readout-kicker to be used until the controller supports the readout, sets up its list of modules to readOut.

It creates a buffer:

```
p_readoutBuffer = new FormattedBuffer (bufferSize);
```

Then Loops:

```
for each acquisition {
        p_readoutBuffer->makeRodHeader(sourceId, runNumber, sequenceNumber,
        sequenceType);

        for each module i {
                p_readoutBuffer->beginHeader(L1CaloModuleType::moduleType, i);
                module->readOut(p_readoutBuffer);
                p_readoutBuffer->endHeader(L1CaloModuleType::moduleType, i);
        }
        statusVector.push_back(statusWords);
        p_readoutBuffer->makeRodTrailer(statusVector);

        (At this point the buffer contents are published to the Monitoring.)

        p_readoutBuffer->clear();
        statusVector.clear();
}
```

Normally, the module readout places some data into the buffer with (*p_readoutBuffer)+=data and then delegates readout to its submodules. In this case the submodule type is determined automatically using Part::parttype() , along with the index (from the last index given in the sub-module name [j,k,.....m]).

Should the s/w designer wish to insert his own data, he is free to do so, using the syntax:

```
p_readoutBuffer->beginHeader(L1CaloSubModuleType::submoduleType, index);
(*p_readoutBuffer)+=data ...;
p_readoutBuffer->endHeader(L1CaloSubModuleType::submoduleType, index);
```

In the latter case, he must either use the actual names of DaqSubModules which he has created as parts, or enter a virtual-submodule name at the end of the L1CaloSubModuleType list given in the enumgen_direct.dat file found within the infraL1Calo package.

The formatting is able to handle nested or un-nested readout sequences. (Nested being the case when a new submodule header is encountered before the corresponding end-header.)

Formatting.

Two long words for each header. The bit assignments are:

First word.

hs iiiiii tttttt oooooooooooooooo

h = header bit, always set to 1
s = submodule bit, only set when a submodule is being readout
iiiiii = module/submodule index field - this is index corresponding to the module name (0-127)
tttttt = module/submodule type field - this is the type of module/submodule (0-127)
oooooooooooooooo = this is the total number of long words, including header words, for this module

Second Word.

hs ee rrrr cccccccc oooooooooooooooo

h = header bit, always set to 1
s = submodule bit, only set when a submodule is being readout
ee = error field. 0x1 is counterror, and 0x2 is offsetError
rrrr = reserved.
cccccccc = this is the number of subModule readOuts nested at the next level
oooooooooooooooo = this is the number of words from the firstword of the current header to the next nested
header.

Here is a sample output from test_simplebuffer.cxx (the 3rd iteration.)

```
ee1234ee  00000009  02040000  00005555  00006666  00000000  00000000  00000000
00000051  800b0027  80030003  a5a5a5a5  c01d000c  c0030003  feedc0de  c0200003
c0000000  b0f00000  c0a00003  c0000000  00000001  c1200003  c0000000  00000002
c09d000c  c0030003  feedc0de  c0200003  c0000000  b0f00000  c0a00003  c0000000
00000001  c1200003  c0000000  00000002  c11d000c  c0030003  feedc0de  c0200003
c0000000  b0f00000  c0a00003  c0000000  00000001  c1200003  c0000000  00000002
808b0027  80030003  a5a5a5a5  c01d000c  c0030003  feedc0de  c0200003  c0000000
b0f00000  c0a00003  c0000000  00000001  c1200003  c0000000  00000002  c09d000c
c0030003  feedc0de  c0200003  c0000000  b0f00000  c0a00003  c0000000  00000001
c1200003  c0000000  00000002  c11d000c  c0030003  feedc0de  c0200003  c0000000
b0f00000  c0a00003  c0000000  00000001  c1200003  c0000000  00000002  810b0027
80030003  a5a5a5a5  c01d000c  c0030003  feedc0de  c0200003  c0000000  b0f00000
c0a00003  c0000000  00000001  c1200003  c0000000  00000002  c09d000c  c0030003
feedc0de  c0200003  c0000000  b0f00000  c0a00003  c0000000  00000001  c1200003
c0000000  00000002  c11d000c  c0030003  feedc0de  c0200003  c0000000  b0f00000
c0a00003  c0000000  00000001  c1200003  c0000000  00000002  deadbeee  deadbeef
00000002  00000075  00000001
```