StEvent I/O Model
And
Writing a Maker
Or
How to Add a New Detector

Akio Ogawa

BNL

2003 Nov 20 @ Dubna

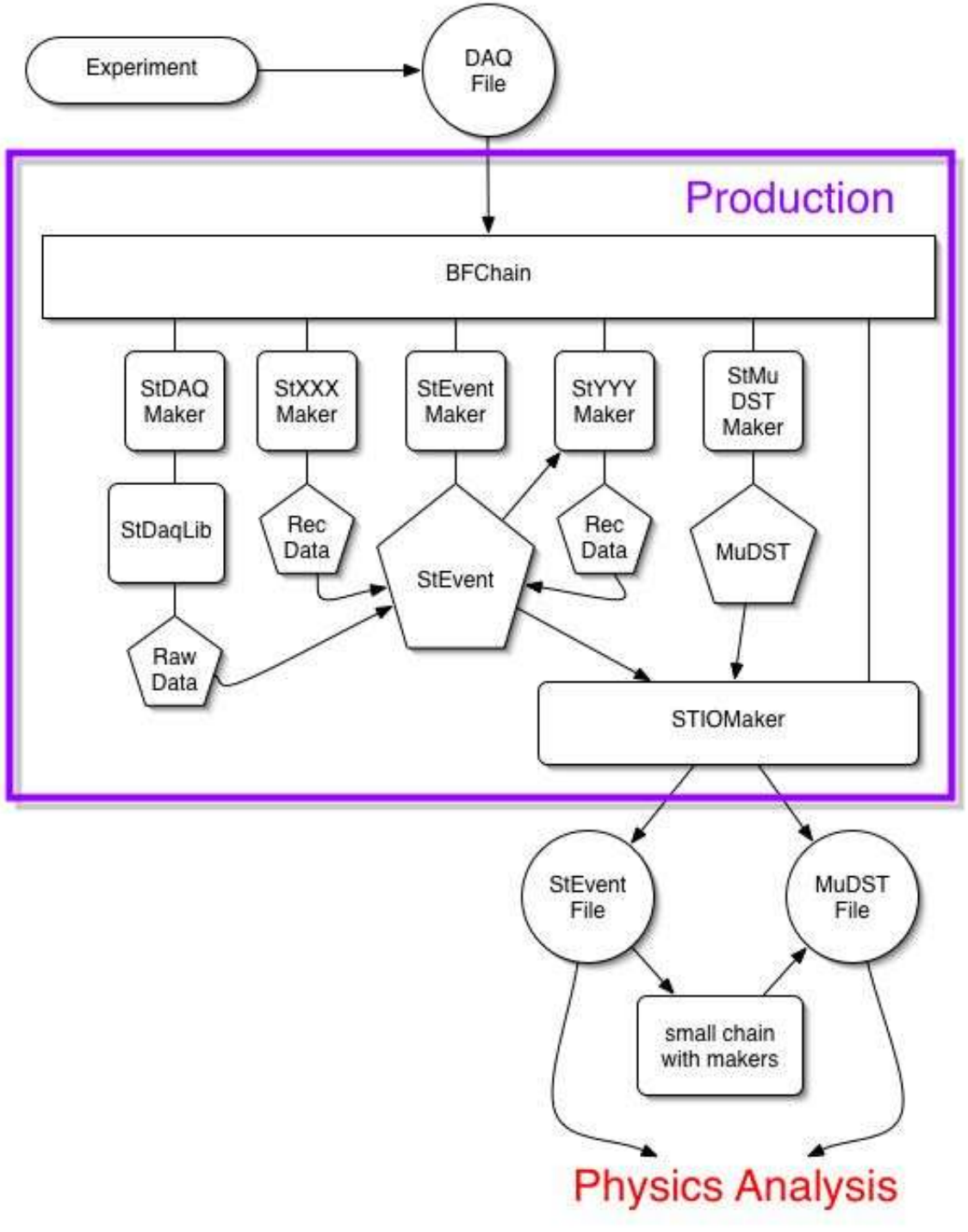
What is StEvent?

- StEvent is our DST
  - It has all reconstructed (and some raw) data
    - Run#, beam conditions, trigger information, etc
    - Hits/Tracks from TPC and other tracking detectors
    - ADC/Enegry/Cluster/"Points" from EMC
    - All other detector information
  - StEvent can be saved to a file, using root I/O (StTreeMaker)
  - StEvent need to be always backward compatible - requires good design from the beginning
- StEvent is our "common block" + subroutines
  - Created or read back from files on memory
  - Contains quite some useful functions associated with data
- StEvent is collection of C++ classes

What is Maker (and chain)?

- Maker is our reconstruction / analysis model
  - Standard functions:
    - Init() once at beginning
    - Make() every event
    - Finish() at the end
    - InitRun() every time new run number is found
    - Clear() at the end of every event
- Chain is our analysis framework
  - Chain automatically collects all makers initiated
  - Chain runs all makers in sequence
  - Allow makers to access object/data in other maker
- BFC (Big Full Chain = Bolshaya Polnaya Tsepochka) is our production chain

Big Picture


Experiment
DAQ File
Production
BFChain
StDAQ Maker
StXXX Maker
StEvent Maker
StYYY Maker
StMu DST Maker
StDaqLib
Rec Data
StEvent
Rec Data
MuDST
Raw Data
STIOMaker
StEvent File
MuDST File
small chain with makers
Physics Analysis

Writing your first StEvent class & a Maker

Example:

You got a new detector called Xxx

Adding a class in StEvent for the detector

Write a maker to put data in

This should be useful for other kind of Makers

Not covered by this talk:

- StDaqMaker, StDaqLib need update (contact experts)
- Documentation (See Petr's tutorial)
- Modifying BFChain (you need to contact Jerome)
- Usage of containers in StEvent
- Coding standards

Files and directory structure and how to compile

- At your working directory:
  - “cvs checkout StRoot/StEvent”
  - “mkdir StRoot/StXxxMaker”
- Your will be creating 4 new files
  - StRoot/StEvent/StXxxCollection.h
  - StRoot/StEvent/StXxxCollection.cxx
  - StRoot/StXxxMaker/StXxxMaker.h
  - StRoot/StXxxMaker/StXxxMaker.cxx
- To compile
  - “cons”
  - And have a coffee break

1) Name, and what it is

StXxxCollection.h

```
#ifndef StXxxCollection_hh
#define StXxxCollection_hh
#include "StObject.h"

class StXxxCollection : public StObject {
public:
   StXxxCollection();
   virtual ~StXxxCollection();

   int getADC(int ch);
   void setADC(int ch, int val);

   int getADCSum();

protected:
   enum {mNchannel=10;};//!
   int mADC[mNchannel];
   int mTDC[mNchannel];

   ClassDef(StXxxCollection, 1)
};
#endif
```

Do not use “#define Nch 10”

Data members start with “m”

StXxxCollection.cxx

```
#include "StXxxCollection.h"

ClassImp(StXxxCollection)

StXxxCollection::StXxxCollection(){
StXxxCollection::~StXxxCollection(){

int StXxxCollection:getADC(int ch){
   return mADC[ch];
}

void StXxxCollection:setADC(int ch, int val){
   mADC[ch]=val;
}

int StXxxCollection:getADCSum(){
   int sum=0;
   for(int i=0; i<mNchannel; i++){
      sum+=mADC[i];
   }
   return sum;
}
```

For class name:

First letter is capital “X”

Later are all non-capital “x”

2) Define as a delivered class from StObject

StXxxCollection.h

```
#ifndef StXxxCollection_hh
#define StXxxCollection_hh
#include "StObject.h"

class StXxxCollection : public StObject {
public:
   StXxxCollection();
   virtual ~StXxxCollection();

   int getADC(int ch);
   void setADC(int ch, int val);

   int getADCSum();

protected:
   enum {mNchannel=10;};//!
   int mADC[mNchannel];
   int mTDC[mNchannel];

    ClassDef(StXxxCollection, 1)
}
#endif
```

StXxxCollection.cxx

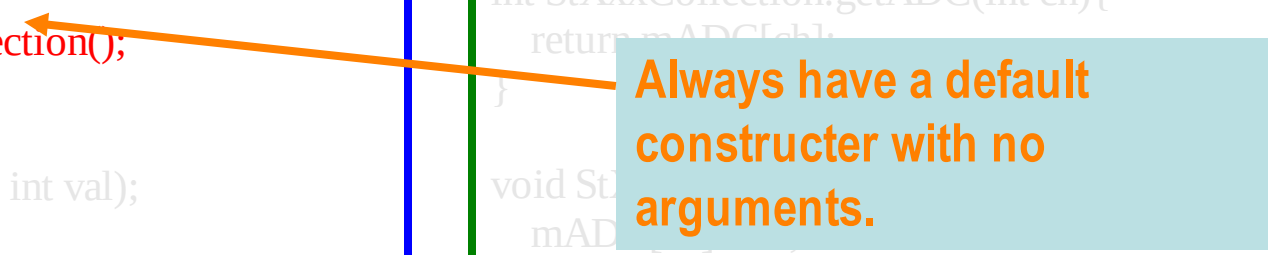
```
#include "StXxxCollection.h"
ClassImp(StXxxCollection)

StXxxCollection: StXxxCollection() {}
StXxxCollection: ~StXxxCollection() {}

int StXxxCollection:getADC(int ch){
   return mADC[ch];
}

void StXxxCollection:setADC(int ch, int val){
   mADC[ch]=val;
}

int StXxxCollection:getADCSum(){
   int sum=0;
   for(int i=0; i<mNchannel; i++){
      sum+=mADC[i];
    }
    return sum;
}
```



3) Put access functions

StXxxCollection.h

```
#ifndef StXxxCollection_hh
#define StXxxCollection_hh
#include "StObject.h"

class StXxxCollection : public StObject {
public:
   StXxxCollection();
   virtual ~StXxxCollection();

   int getADC(int ch);
   void setADC(int ch, int val);

   int getADCSum();
   void setDebug(bool b) {mDebug=b;}
protected:
   enum {mNchannel=10;};
   int mADC[mNchannel];
   int mTDC[mNchannel];
   bool mDebug;
    ClassDef(StXxxCollection, 1)
}
#endif
```

StXxxCollection.cxx

```
#include "StXxxCollection.h"
ClassImp(StXxxCollection)

StXxxCollection: StXxxCollection() {}
StXxxCollection: ~StXxxCollection() {}
int StXxxCollection:getADC(int ch){
   return mADC[ch];
}
void StXxxCollection:setADC(int ch, int val){
   if(ch<0 || ch>=mNchannel){
      if(mDebug) { cout << “error!” << endl;}
   }else{
     mADC[ch]=val;
   }
}
int StXxxCollection:getADCSum(){
   int sum=0;
```

For function name:

First word start with non-capital

Later words are all start with capital

4) Adding root specific stuff

StXxxCollection.h

```
#ifndef StXxxCollection_hh
#define StXxxCollection_hh
#include "StObject.h"

class
pub
   S
   v

   int getADC(int ch);
   void setADC(int ch, int val);

   int getADCSum();
   void setDebug(bool b) {mDebug=b;}
protected:
   enum {mNchannel=10;};
   int mADC[mNchannel];
   int mTDC[mNchannel];
   bool mDebug; //!
    ClassDef(StXxxCollection, 1)
}
#endif
```

2 macros for root CINT

ClassDef(name,version#)

ClassImp(name)

StXxxCollection.cxx

```
#include "StXxxCollection.h"
ClassImp(StXxxCollection)

StXxxCollection: StXxxCollection() {}
StXxxCollection: ~StXxxCollection() {}
int StXxxCollection:getADC(int ch){
   return mADC[ch];
}
void StXxxCollection:setADC(int ch, int val){
   if(ch<0 || ch>=mNchannel){
      if(mDebug) { cout << "error!" << endl;}
   }else{
     mADC[ch]=val;
   }
}
int StXxxCollection:getADCSum(){
   int sum=0;
   for(int i=0; i<mNchannel; i++){
      sum+=mADC[i];
   return sum;
}
```

"//!" means "do not save to file"

5) Adding more advanced access functions

StXxxCollection.h

```
#ifndef StXxxCollection_hh
#define StXxxCollection_hh
#include "StObject.h"

class StXxxCollection : public StObject {
public:
   StXxxCollection();
   virtual ~StXxxCollection();

   int getADC(int ch);
   void setADC(int ch, int val);
   int getNChannel() {return mNchannel;}
   int getADCSum();
   void setDebug(bool b) {mDebug=b;}
protected:
   enum {mNchannel=10;};
   int mADC[mNchannel];
   int mTDC[mNchannel];
   bool mDebug; //!
    ClassDef(StXxxCollection, 1)
}
#endif
```

StXxxCollection.cxx

```
#include "StXxxCollection.h"
ClassImp(StXxxCollection)

StXxxCollection: StXxxCollection() {}
StXxxCollection: ~StXxxCollection() {}
int StXxxCollection:getADC(int ch){
   return mADC[ch];
}
void StXxxCollection:setADC(int ch, int val){
   if(ch<0 || ch>=mNchannel){
      if(mDebug) { cout << “error!” << endl;}
   }else{
     mADC[ch]=val;
   }
}
int StXxxCollection:getADCSum(){
   int sum=0;
   for(int i=0; i<mNchannel; i++){
      sum+=mADC[i];
    }
    return sum;
}
```

5) Adding data member : **schema evolution should take care it**

StXxxCollection.h

```
#ifndef StXxxCollection_hh
#define StXxxCollection_hh
#include "StObject.h"

class StXxxCollection : public StObj
public:
   StXxxCollection();
   virtual ~StXxxCollection();

   int getADC(int ch);
   void setADC(int ch, int val);
   int getNChannel() {return mNchar
   int getADCSum();
   void setDebug(bool b) {mDebug=b;}
protected:
   enum {mNchannel=10;};
   int mADC[mNchannel];
   int mTDC[mNchannel];
   bool mDebug; //!
    ClassDef(StXxxCollection, 2)
}
#endif
```

StXxxCollection.cxx

```
#include "StXxxCollection.h"
ClassImp(StXxxCollection)

                                  () {}
                                  n() {}
                                  ){


                                  ch, int val){

                                  " << endl;}
      mADC[ch]=val;
   }
}
int StXxxCollection:getADCSum(){
   int sum=0;
   for(int i=0; i<mNchannel; i++){
      sum+=mADC[i];
    }
    return sum;
}
```

Adding new valuables -OK

Adding new functions - always OK

Do NOT change order of valuables

Do NOT change the name of valuables

New version #

A) Now, write a Maker to fill it

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name="Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
          {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
          {return StMaker::Finish();};

  ClassDef(StXxxMaker,0)
}
#endif
```

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->XxxCollection();
  if(xxxCollection == 0){
    xxxCollection = new StXxxCollection();
    mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
  if(daqReaderDS == 0) return kStWarn;
  StDAQReader* daqReader = (StDAQReader*)(daqReaderDS->GetObject());

  if (daqReader == 0) return kStWarn;
  if (!(daqReader->XxxPresent())) return kStOK;
  StXxxReaderInterface* xxxReader
                = xxxReader->getXxxReader();
  if (xxxReader==0) return kStWarn;
  for (int i=0; i<xxxCollection->getNChannel(); i++){
    XxxCollection->setAdc(i,XxxReader->GetAdc(i));
  }
}
```

Derived from StMaker

Name will be used to access to this Maker from other Makers

B) Init, Make and Finish

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name="Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
          {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
          {return StMaker::Finish();};

  ClassDef(StXxxMaker,0)
}
#endif
```

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->XxxCollection();
  if(XxxCollection == 0) {
     xxxCollection = new StXxxCollection();
     mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
  if (daqRead
  StDAQRea

  if (daqRead
  if (!(daqRea
  StXxxReaderInterface* xxxReader
                  = xxxReader->getXxxReader();
   if (xxxReader==0) return kStWarn;
   for (int i=0; i<xxxCollection->getNChannel(); i++){
     XxxCollection->setAdc(i,XxxReader->GetAdc(i));
   }
}
```

Init (once at beginning)

Make (every event)

Finish (once at the end)

C) How to access to StEvent?

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name="Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
            {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
            {return StMaker::Finish();};

  ClassDef(StXxxMaker,0)
}
#endif
```

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->XxxCollection();
  if(XxxCollection == 0) {
     xxxCollection = new StXxxCollection();
     mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
  if (daqReaderD
  StDAQReader
                  =                    )(daqReaderDS->GetObject());
  if (daqReader==0) return kStWarn;
  if (!(daqReader->XxxPresent())) return kStOK;
  StXxxReaderInterface* xxxReader
                  = xxxReader->getXxxReader();
   if (xxxReader==0) return kStWarn;
   for (int i=0; i<xxxCollection->getNChannel(); i++){
     XxxCollection->setAdc(i,XxxReader->GetAdc(i));
   }
}
```

This is how you can obtain StEvent

D) Get, or create your class

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name="Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
            {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
            {return StMaker::Finish();};

  ClassDef(StXxxMaker,0)
}
#endif
```

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->xxxCollection();
  if(xxxCollection == 0) {
     xxxCollection = new StXxxCollection();
     mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
  if (daqReaderDS==0) return kStWarn;
  StDAQReader* daqReader
                  = (StDAQReader*)(daqRea
  if (daqReader==0) return kStWarn;
  i                esent())) return k
  S                xxxReader
                   er->getXxxReader();
  i                rn kStWarn;
  for (int i=0; i<xxxCollection->getNChannel(); i++){
     xxxCollection->setAdc(i,xxxReader->GetAdc(i));
  }
}
```

If it's empty, create one and add to StEvent

Try to get StXxxCollection from StEvent

E) How to obtain raw data?

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name="Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
            {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
            {return StMaker::Finish();};

  ClassDef(StXxxMaker,0)
}
#endif
```

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->xxxCollection();
  if(xxxCollection == 0) {
     xxxCollection = new StXxxCollection();
     mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
   if (daqReaderDS==0) return kStWarn;
  StDAQReader* daqReader
                  = (StDAQReader*)(daqReaderDS->GetObject());
   if (daqReader==0) return kStWarn;
   if (!(daqReader->XxxPresent())) return kStOK;
  StXxxReaderInterface* xxxReader
                  = xxxReader->getXxxReader();
    if (xxxReader==0) return kStWarn;
   for (int i=0; i<xxxCollection->getNChannel(); i++){
      xxxCollection->setAdc(i,xxxReader->GetAdc(i));
   }
}
```

Get Raw Data reader from "StDAQReader"

F) Now, actually fill it

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name="Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
            {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
            {return StMaker::Finish();};

  ClassDef(StXxxMaker,0)
}
#endif
```

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->xxxCollection();
  if(xxxCollection == 0) {
     xxxCollection = new StXxxCollection();
     mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
  if (daqReaderDS==0) return kStWarn;
  StDAQReader* daqReader
                = (StDAQReader*)(daqReaderDS->GetObject());
  if (daqReader==0) return kStWarn;
  if (!(daqReader->XxxPresent())) return kStOK;
  StXxxReaderInterface* xxxReader
                = xxxReader->getXxxReader();
  if (xxxReader==0) return kStWarn;
  for (int i=0; i<xxxCollection->getNChannel(); i++){
     xxxCollection->setAdc(i,xxxReader->GetAdc(i));
  }
}
```

Fill StXxxCollection from raw data

G) Return values and some protections

Makers should NOT assume that input / other makers are there

" independence of makers"

#include "StMaker.h"

Return Values:

kStOk = Good

kStWarn = Warning, but continue

kStErr = Error, skip this event for this maker

kStFatal = Fatal, skip event (do not use in BFC!)

(kStEOF = End of File)

StXxxMaker.cxx

```
Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
  StXxxCollection* xxxCollection;
  StEvent *mEvent = (StEvent *) GetInputDS("StEvent");
  if(mEvent == 0) return kStWarn;
  xxxCollection = mEvent->xxxCollection();
  if(xxxCollection == 0) {
     xxxCollection = new StXxxCollection();
     mEvent->setXxxCollection(xxxCollection);
  }
  St_DataSet* daqReaderDS = GetDataSet("StDAQReader");
  if (daqReaderDS==0) return kStWarn;
  StDAQReader* daqReader
                  = (StDAQReader*)(daqReaderDS->GetObject());
  if (daqReader==0) return kStWarn;
  if (!(daqReader->XxxPresent())) return kStOK;
  StXxxReaderInterface* xxxReader
                  = xxxReader->getXxxReader();
   if (xxxReader==0) return kStWarn;
  for (int i=0; i<xxxCollection->getNChannel(); i++){
     xxxCollection->setAdc(i,xxxReader->GetAdc(i));
  }
return kStOK;}
```

H) Include files

StXxxMaker.h

```
#ifndef STAR_StXxxMaker
#define STAR_StXxxMaker
#include "StMaker.h"
class StEvent;

class StXxxMaker : public StMaker {
public:
  StXxxMaker(const char *name=”Xxx");
  virtual ~StXxxMaker() {};
  virtual Int_t Init()
            {return StMaker::Init();};
  virtual Int_t Make();
  virtual Int_t Finish()
            {return StMaker::Finish();};
  Int_t fillFrom(StEvent* evt);

  ClassDef(StXxxMaker,0)
}
#endif
```

Single include file for ALL classes in StEvent

StXxxMaker.cxx

```
#include “StXxxMaker.h”
#include “StEventTypes.h”
#include "StDaqLib/XXX/XXX_Reader.hh"
#include "StDAQMaker/StDAQReader.h"

Classimp(StXXXMaker)
StXxxMaker::StXxxMaker(const char *name):StMaker(name) {}
Int_t StXxxMaker::Make(){
…
return kStOK;
}
Int_t fillFrom(StEvent *evt){
…
}
```

These raw data reader need to be implemented by experts

Do not include in .h file unless absolutely needed
Instead, use class <class name> if this is enough

What's Next?

- Contact Thomas Ullrich for modifying mother StEvent classes to add your class
- Contact Jerome Lauret for adding your maker to BFChain
- One should really read real documents for deeper understanding on StEvent and Makers:
  - http://www.star.bnl.gov/STAR/comp/root/special_docs.html
  - http://www.star.bnl.gov/STAR/comp/train/tut/Maker-in-STAR/
- Once done, you should be able to run it with macro bfc.C