Dataflow Calibration Scenario

A simple example for making a DIRC pedestal calibration run within the BaBar Dataflow system.

- Prerequisites
- Dataflow classes
- Calibration classes
- DIRC classes
- Config transition
- L1 Accept transition
- Summary

Prerequisites

- Calibration request queuing system
  - Subsystems individually request a specific calibration specifying:
    - What calibration type
    - What running conditions are required (IE beam off,...)
    - How 'urgent' the need is (can wait 1 hour,....)
  - Online system synchronizes requests to schedule calibration runs
    - Majority logic to define global urgency
    - Ability to force PEPII to stall when sufficiently urgent beam off calibration requests have accumulated
- Code downloading to the ROM
- Configuration data downloading to the ROM

Dataflow Classes

```
//
//  Simple implementation of Tagged Container
//
class MyTC : public dfTaggedContainer {
public:
  int Address();
  int Tag();
  dfObject* Insert(dfObject& item);
  dfTaggedContainer* Insert(dfTaggedContainer& item);
  dfTaggedContainer* Remove(dfTaggedContainer& item);
  dfObject* Object();
  dfObject* Object(dfObject &item);
  int Items();
};


//
//  Simple implementation of TC browser
//
class MyBrowser : public dfTcBrowser {
public:
  MyBrowser(dfTaggedContainer& hierarchy);
  dfTaggedContainer* Up(int levels);
  dfTaggedContainer* Down(int levels);
  dfTaggedContainer* Prev();
  dfTaggedContainer* Next();
  dfTaggedContainer* GetTc();
  dfTaggedContainer* Seek();
  dfTcBrowser& operator=(dfTcBrowser&);
};
```

Calibration DF Classes

```
//
//   Dataflow calibration channel base class,
//   used to accumulate statistics during
//   cycles.
//
class dfCalChan : public dfObject {
public:
  virtual Clear() = 0;
  virtual Increment(int value) = 0;
};

//
//   Specific implementation of a Dataflow
//   calibration channel class appropriate
//   for Pedestal calibration.  The Increment
//   function accumulates a running mean and
//   variance.
//
class PedChan : public dfCalChan {
public:
  PedChan();
  Clear();
  Increment(int value);
 private:
  int sum;
  int sum2;
  int numincrement;
};
```

DIRC Dataflow Classes

```
//
//  DIRC single channel raw data class
//
class DircData : public dfObject {
public:
    int ADCvalue();
    int TDCvalue();
    int ChipChannel();
private:
    int some_data;
};
```

Config Transition

```
//  Class to define Config transition action
class configAction : public dfAction {
public:
  configAction(int duration){_duration = duration;};
  int Duration(){return _duration;};
  void Fire(dfDatagram& input, dfDatagram& output);
private:
  int _duration;
};
//
//  Callback to Config transition for pedestal
//  calibration.
void configAction::Fire(dfDatagram& input,
                        dfDatagram& output)
{
// Download config data here (Someday).
//
// Build the calibration output. Someday this
// will be a single templated call.
//
  pedout = new dfMap();
  pedb = new MyBrowser(*pedout);
// Start with the top level (ROM) container
  MyTC *node = (MyTC*)pedb->GetTc();
  pedb->Down(1);     // Move to element level
  while(node =  (MyTC*)pedb->Next() ){
    pedb->Down(1);  // Move to chip level
    while(node =  (MyTC*)pedb->Next()){
      pedb->Down(1);// Move to channel level
       while(node =  (MyTC*)pedb->Next()){
         PedChan *pc = new PedChan();
          node->Insert(*pc);
      } pedb->Up(1);
    } pedb->Up(1);
  } pedb->Up(1);
}
```

L1 Accept Transition

```
// Class to define L1 Accept transition action
class L1Action : public dfAction {
public:
  L1Action(int duration){_duration = duration;};
  int Duration(){return _duration;};
  void Fire(dfDatagram& input, dfDatagram& output);
private:
  int _duration;
};
// Callback to L1 Accept transition for
// pedestal calibration.  This loops over the
// (sparse) input container and increments the
// corresponding PedChan for the ADC data.
// Will someday be encapsulated/templated.
void L1Action::Fire(dfDatagram& input,
                    dfDatagram& output)
{
  MyBrowser ib(*(dfTaggedContainer*)input.Object());
  MyTC *pnode;
  DircData *dd;
  PedChan *pc;
// ROM Tagged Container from input
  MyTC *inode =  (MyTC*)ib.GetTc();
  ib.Down(1);                         // Move to element level
  while(inode =  (MyTC*)ib.Next()){
    ib.Down(1);                       // Move to chip level
    while(inode =  (MyTC*)ib.Next()){
      ib.Down(1);                     // Move to channel level
      while(inode =  (MyTC*)ib.Next()){
       dd = (DircData*)inode->Object();
       *pedb = ib;  // Assign pedb to point to
                    // same channel as raw data
       pnode = (MyTC*) pedb->GetTc();
       pc = (PedChan*) pnode->Object();
       pc->Increment(dd->ADCvalue());
      } ib.Up(1);
    } ib.Up(1);
  } ib.Up(1);
}
```

Summary

- It is possible to construct useful scenarios using the Dataflow API
- Some rough edges have been (and are still being) smoothed out
- There are major pieces missing
    - Code download
    - Data download
    - Calibration queuing system
- There's a lot of code to write!