Outline

- Status
  - initialization of global parameters / objects
  - disadvantage

- Proposal
  - global initialization task(s)
  - introduction of a typesafe dictionary with job live time to store and get access to global objects (parameters)

Status of Global Initialization

- Lots of objects need access to subsystem specific event or job based information, e.g. (EMC)
  - EmcMapper -> TwoCoordinateIndex, etc.
  - branch addresses
  - calibration constants & algorithms
  - geometry & alignment constants
  - . . .
- Initialization done at different places and partly several times
  - root macros
  - individual tasks
  - hard coded within several classes

Disadvantage of Present Status

- Initialization in root macros
  - user has to take care of the initialization
- Hard coded within several classes
  - error prone
  - once something has been changed, all relevant classes have to be modified
- Initialization at one well defined place with global access to the information would make the life easier

Proposal: Task for Global Initialization

- Initialization task for the individual subsystems
- Global initialization task containing the individual subsystem initializations
- Advantages
  - user has not to take care of it
  - initialization will be done only once and at a well defined place
  - maintenance of the code much easier

Proposal: Global Dictionary

- Global dictionary class would help to get an easy access to all global information
  - (global) static pointer to this dictionary class
  - contains subsystem specific dictionary classes
- Initialization via global initialization task
- Some information can change from event to event
  - access to database via dictionary
    - client might ask for some infos but not necessarily
    - deferment of data until the time that the user requests it
    - improvement of the performance
    - Event time stamp as key to query database

Proposal: Global Dictionary

```
class AbsEvt
{
public:
...
virtual EmcEnv* getEmc ();
virtual SttEnv*   getStt();
...

virtual void setEmc ( EmcEnv* envPointer);
virtual void setStt (SttEnv* envPointer);
...
};

// global pointer to env
AbsEnv* gblEnv;
```