The SAMBA reference book

Summary

I. Introduction

SAMBA is not only a readout loop to store detector data in files, but is, mainly, a full application to manage the DAQ hardware and the data taking, from debugging to physics runs, including data quality, monitoring and various status reports. Other important characteristic is that it is widely configurable, either for all that concern the hardware (detectors, digitizers, cabling, processes) or by allowing user scripts to be executed at various key steps. All the setup is defined in text files and can be modified, either directly, or through the SAMBA application (eventually graphically, e.g. the hardware is managed via icons in an open window). For these reasons, SAMBA is the software now in use for experiments as various as cryogenic crystals based (EDELWEISS, CUPID) or all the gaseous spherical experiments (NEWS network) and test benches.

Before going into the enumeration of the SAMBA's functionalities, the objects in use have to be detailed – in the next chapter. After that, we will see how to manage such objects, and finally review a number of implemented actions that help to efficiently run the SAMBA application.

II. Main concepts

A piece of history

SAMBA was born to ensure the data taking of the experiment EDELWEISS, version II.

At its origin, the specifications for SAMBA were rather simple: all detectors were identical in their definition, each detector had its own ADC box, and finally the path from which the data were retrieved were perfectly defined – and fixed. The fit between the detectors (which were actually bolometers) and their ADC box was such, that these boxes were called also “bolometer”, which was a bit confusing in the discussions. I began then to introduce the rule “one name for a given object, one object for a given name”: the ADC boxes became the “Bolometer Boxes”, or “BB”.

After one year or two of data taking, new detectors were to be tested. Following the way they were electrically read, a trick consisting in defining pseudo-detectors assembled by 2 (or 3) to represent each new ones, was added to SAMBA.

After that, again, a new version of the “Bolometer Box” was produced, much more versatile and with no more trivial connectivity with the old or future detectors.

Starting at this time, the EDELWEISS specific names were disused, and a generic terminology is now used all along the software – including about the file names the end-user has to deal with.

All this said, to justify this chapter which defines precisely the concepts and related terminology in use in SAMBA.

First level definitions

The hardware chain which translates and transfers the physical state of the detection device, up to the numerical data storage, is described and managed in SAMBA using precise definitions.

This hardware chain can be shown as follows:

Even if other (eventually better) names could be used, or some names look equivalent, only the present names have to be used in the SAMBA framework. Each object has some useful properties like its particular user-defined name, links to the rest of the chain, individual characteristics, … however in the following, we shall give the definition of each object, not only in the real life, but from the SAMBA point of view, which basically defines each kind of object.

The above shown objects are generally defined as a list of lower-level SAMBA objects (emphasized in italics), which will be defined right after.

Detectors

- ***function***

  Obviously, a detector is the location of physical events (expected, or not). In High Energy Physics, it converts the occurrence of a particle into some macroscopic quantities.

- ***definition***

  A detector is defined by a list of *channels* and a list of *settings*.

Digitizers

- ***function***

  A digitizer converts (with *ADC*s) electrical analog quantities (the ones delivered through the cabling) into numerical values.
  It may, at the opposite, deliver analog voltages or digital commands towards the detectors, when it is equipped by DACs. More generally, a digitizer may contain a number of memory locations (registers) which allow to do more than DAC conversions, e.g. some slow control on the electronics; all memory locations – DACs and registers - are called *resources*.

- ***definition***

  A digitizer is a list of *ADC*s and a list of *resources*.

Dispatchers

- ***function***

  A dispatcher collects the numerical data from the digitizers, and send them to the computers.
  It may process the data during its transfer, e.g. select events (to lower the amount of transmitted events).
  It may also transmit user values from the computer to the digitizers resources, which in turn may apply voltages (or so) on the detectors.

- ***definitions***

  A dispatcher is defined by its transmission technology (USB, Ethernet, ..) and a list of the *connections* it has access to.

Cabling

- ***function***

  The cabling defines which sensor is connected to which ADC of which digitizer, and which DAC is connected to which detector bias.

- ***definition***

  A cabling is defined by a *detector location*, one or more *digitizer connectors*, and an ordered list of *detector sensors* associated to an ordered list of *digitizer ADC*s. The word “cabling” is reserved to these analog links.

Connections

- ***function***

  The connections links the digitizers to the dispatchers. They are digital links.

- ***definition***

  A connection has a single user name, which is referred to by the digitizers on one side, and the dispatchers on the other side.

Second level definitions

Channels

- ***function***

  Macroscopic changes (which occur in a detector) are accessible via sensors: they convert the physical quantity into, at the end, electricity, typically charges to be processed by a charge amplifier. Since the sensor state is to be accessed by the acquisition, the presence of a sensor implies the existence of a *channel*.

- ***definitions***

  A channel has mainly a user-defined name and is characterized by a *physics*, which is the macroscopic quantity measured.
  The physics names are also user-defined, and allow for physics-dependent data processing.

Settings

- ***function***

  The settings are either the physical quantities applied to or into the detector (voltage, gas pressure, …), or the individual parameters of the data processing.

- ***definition***

  A setting has a user-defined name, a category (from a predefined list) and a value given in the most convenient user unit (Volts, mb, …). A given setting is affected to a particular channel – if not, it will concern globally the detector.

ADCs and resources

- ***function***

  An ADC converts an external voltage to a numerical value, which can be processed later by the software.
  A resource is any addressable register, including the DACs (which convert a numerical value into voltage).

- ***definitions***

  An ADC is defined by its polarization (in volts, signed or not) and its width (the bits number).
  A resource has a number of characteristics (detailed later), among them can be mentioned its *local address* inside the device and the user-to-digital *conversion* rules.

Detector location

- ***function***

  If the experiment uses a number of detectors, a location is to be reserved for each one. The cabling may stay in place even when a detector is changed for another one. So, in SAMBA, each cable is connected to a particular *detector location*, and a given detector may be declared as placed into this location.

- ***definition***

  A detector location is named with a symbol containing 1, 2 or 3 characters representing a 1-, 2- or 3-dimensionnal array.

- ***note***

  Probably, you think right now that the location should include the geometrical 3D actual position. Unfortunately, it appeared so far useless at the acquisition level. Unless at some time somebody want to reconstruct, online, a track over several detectors? Not the case at the moment ☹

Digitizer connector

- ***function***

  If the experiment uses a number of digitizers, a place is to be reserved for each one. The cabling may stay in place even when a digitizer is changed for another one. So, in SAMBA, each cable is connected to a particular *digitizer connector*, and a given digitizer may be declared as plugged into this connector.

- ***definition***

  A digitizer connector is named with a symbol containing 1, or 2 characters representing a 1- or 2-dimensionnal array (implies tags to stamp on the digitizer chassis).

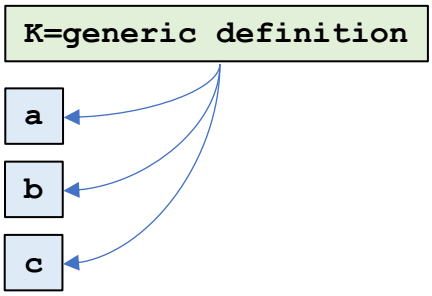
III. Models

If the experiment houses a rather large number of detectors, it may be interesting to factorize the common part of the definitions. In a mathematical formalism,

D = aK + bK + cK + … => D = K . (a + b + c + …)

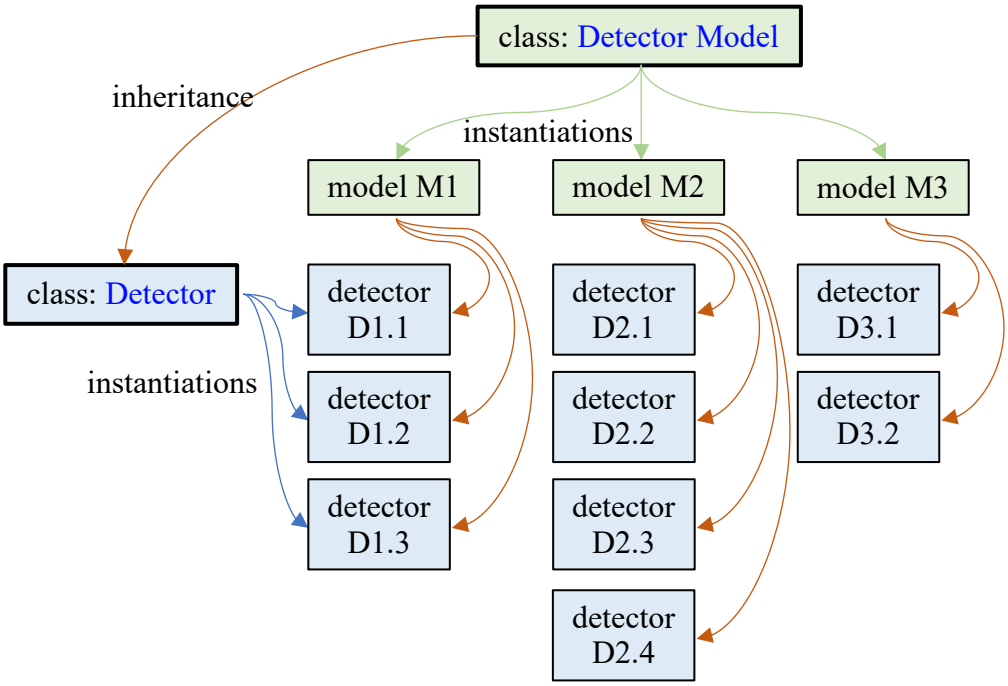
where D is the full detector setup, aK (and bK and cK …) are the fully described individual detectors. K is a definition which is common to all detectors, and a (and b and c …) are the particularities of each detector.

With an organigram point of view:



In an object-oriented language, K is a class and the detectors are instantiations of this class. Note that the class K is itself an instantiation of the generic class “detectors” (amongst the other generic classes as defined in the previous chapter).

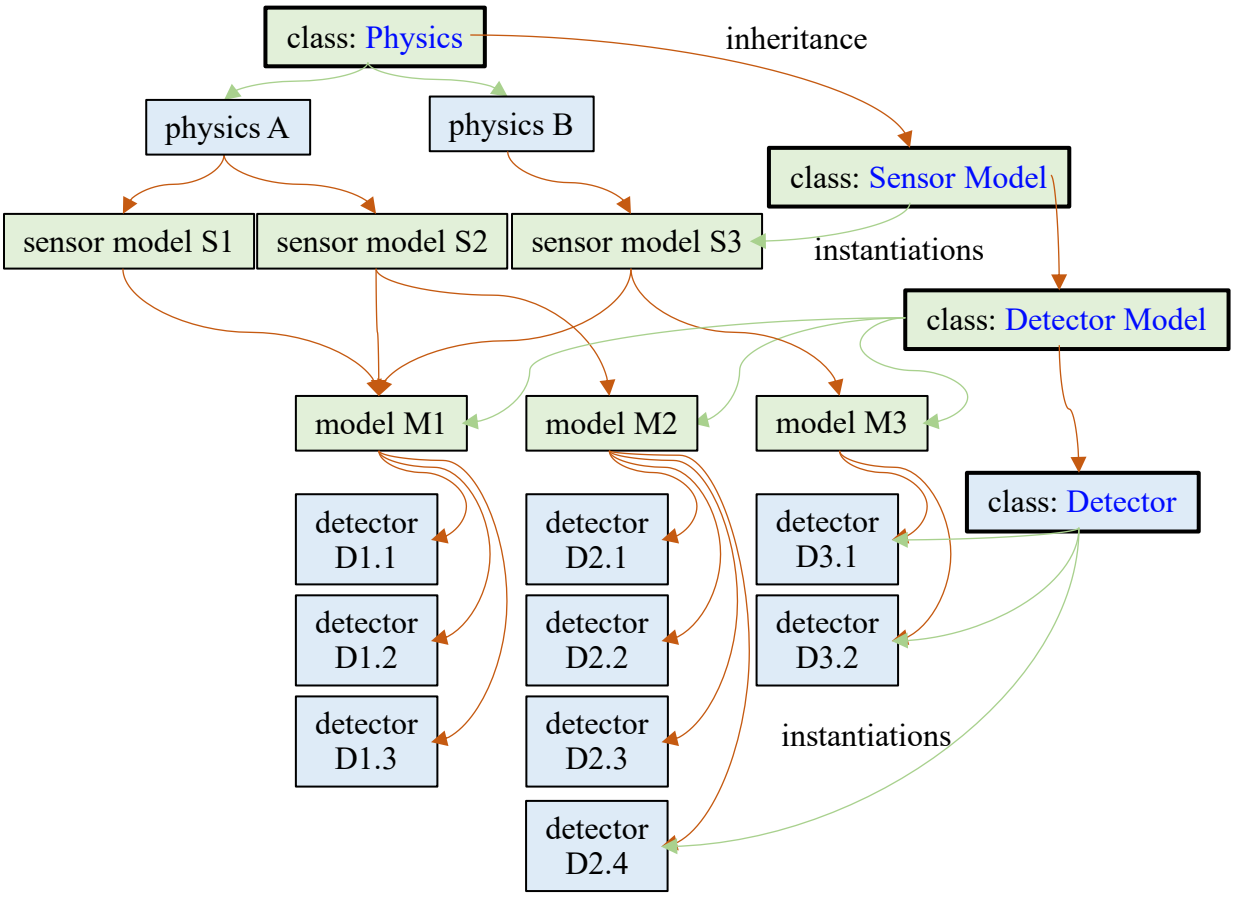
In SAMBA, the class K is called “Detector Model”. The full hierarchy is defined as follows:



Of course, the same figure can be shown for all the object of the previous chapter, that is, there are also models of digitizers, dispatchers, channels, cablings, etc…

Relationship between classes of the detection stage

Let's have a look on the following example, for detectors:



Detectors D1.1, D1.2 and D1.3 are of the same type (M1) and have a number of sensors taken among 3 kinds of sensors (S1, S2 and S3). Kinds S1 and S2 are somehow different but are sensitive to the same physics (A). Detectors D3.1 and D3.2 have also several sensors, but they are of the same kind (S3).

Relevance of Samba

All names and definitions are user-defined in text files. This means that Samba don't support any particular piece of hardware and relies only on external classes description. So, Samba can run any acquisition where the setup can enter in the above structures, and it can be seen that this structure covers probably *any* complex setups.

In particular, since the setup is fully user-defined in text files, the *user interfaces (inputs, outputs) are automatically adjusted* to show in an adapted way the various elements in use (channels, biases, any hardware and software conditions) and no change is needed in Samba despite the continuous changes and improvements of the detectors and the DAQ.

However, access to the digitizers includes a supplementary stage to distribute all data to/from the computers in charge of the data taking. SAMBA can use PCI or Ethernet connections to get the access to this stage, called "dispatchers". The general characteristics of such connections are also defined externally. But, due to the fact that generally dispatchers house some software that make the data packets highly diversified, **a restricted number of dispatcher types have been supported** (3 on PCI, 3 on Ethernet including the IPE4 crate). A remarkable advantage of Ethernet is that it makes the system widely expandable.

When the data format is simple, the concept of 'medium' is used, either through USB or Ethernet. Any complex dispatcher implement 2 media (*to* the dispatcher from the computer, and *from* the dispatcher to the computer), but not all media have the abilities of a dispatcher.

IV. User interface

V. Scripts