

Developers documentation of the electrometer library

Creation date : November 2006

Last modification : 21/11/2006 05:55:00 PM

Version courante du document : 1.1

Modification history

Date	Revision	Description	Auteur	Relecteur
11/2006	1.0	Version initiale	X.Elattaoui	A. Buteau
07/2007	1.1	Minor change	X. Elattaoui	A. Buteau

Table of contents

1 Introduction.....	3
2 Context of this development and goals to achieve	3
2.1 Context of the development for SOLEIL	3
2.2 Aim of the library.....	3
3 Principles of measurements done with electrometers.....	3
3.1 Objectives of pico ammeters acquisitions.....	3
3.2 Keithley Models	4
3.3 Novelec models	4
4 Analysis	4
4.1 Methodology	4
4.2 Summary of functions and protocols for all instruments to interface ...	4
4.3 Another view of the problem	6
4.4 UML analysis model	7
5 Base Classes and responsibilities.....	7
5.1 AbstractElectrometer class	7
5.2 ElectrometerProtocol class	7
5.3 CommunicationLink class	8
5.4 Examples of collaboration diagram: how all that is instanciaded.....	8
5.4.1 For a Keithley 485	8
5.4.2 For a Keithley 6514	8
5.4.3 For a Novelec MCCE-2	8
5.4.4 Note on Instantiation mechanism	8
6 Detailed Library design	8
6.1 Preliminary note.....	8
6.2 AbstractElectrometer class	9
6.2.1 Implementation of default behavior	9
6.2.2 Functions supported only by one protocol.....	9
6.2.3 Particular cases	9
6.2.3.1 auto_range_off method:.....	9
6.2.3.2 GetElectroMeterGain(est utile)	9
6.3 ElectrometerProtocol Class	10
6.4 CommunicationLink class	10

1 Introduction

This document is a guide to help developers in exploring and updating the Electrometers library. This document describes:

- why we developed this library and the goals we wanted to achieve
- the problem analysis
- the software design of the library

2 Context of this development and goals to achieve

2.1 Context of the development for SOLEIL

At SOLEIL, beam lines are under construction and will be running for the beginning of 2007. This implies different users with their own context of measurement and instruments.

Till now, our software policy to interface measurement instrument to TANGO, was to develop one DeviceServer per instrument. Moreover, very often, these development were taken in charge by different developers. Drawbacks of this approach are the following :

- Quality :
 - different developers means different implementations (*each on with its own bugs*)
 - it's difficult to have a common Tango interface between all instruments (which may lead to different interfaces for the user)
- Workload :
 - for each new instrument to interface, a complete software development cycle (*analysis, design, code and test*) has to be done, which is of course time consuming.
 - Maintenance is also a concern, as there are many different software projects to deal with.

2.2 Aim of the library

The goal of this library for our software development group is :

- to decrease the peak of work of SOLEIL's software development team
- to have a reusable and maintainable source code for all Electrometers models
- to have as less Tango devices as possible to maintain

For our beam line users, followings advantages are expected :

- to deal with a single Tango interface
- Configuration is simplified and homogenous for all electrometers

3 Principles of measurements done with electrometers

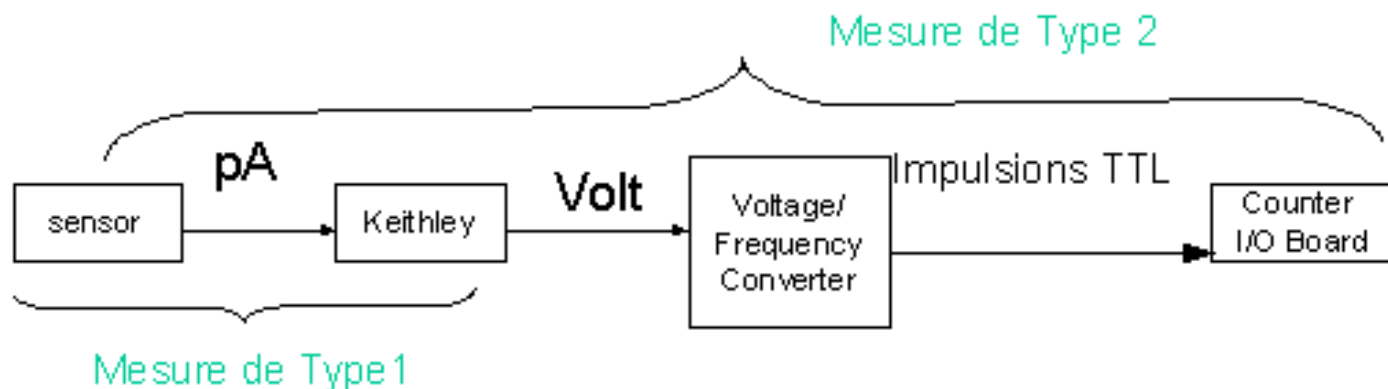
3.1 Objectives of pico ammeters acquisitions

At SOLEIL, Keithley electrometers are used on beamlines for 2 different kinds of measurements :

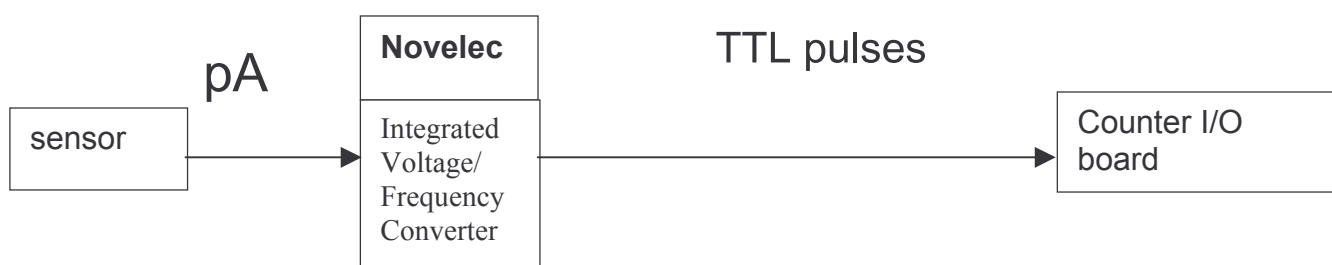
Type 1 measurement : often in the optical hutch, pico ammeters are used to locate the beam, and a raw instantaneous read of the current on the instrument provides the required information.

Type 2 measurement : for acquisitions to be done on the sample, an integrated measurement is mandatory to get the required precision and resolution. In this case, a voltage/frequency converter and a counter board provides the time integration mechanism (*as explained on the following schemes*)

3.2 Keithley Models



3.3 Novelec models



On a few SOLEIL beamlines, Novelec electrometers are used for pico current acquisitions. The main difference is that all models integrate a voltage/frequency converter.

[Demander à YMA le contexte de la mesure sur DIFFABS](#)

4 Analysis

4.1 Methodology

We have established a list of the various electrometers models that will be used on the first beamlines at SOLEIL. *(we do not pretend to cover all Keithley models but we were only focused on those used at SOLEIL.)*

Then we read the technical manuals of these instruments, to list the functions and measurements modes supported by each model, and the controls specifications *(like communication bus and protocols)*. This study has lead to the following synthesis, which is the key to understand the software design .

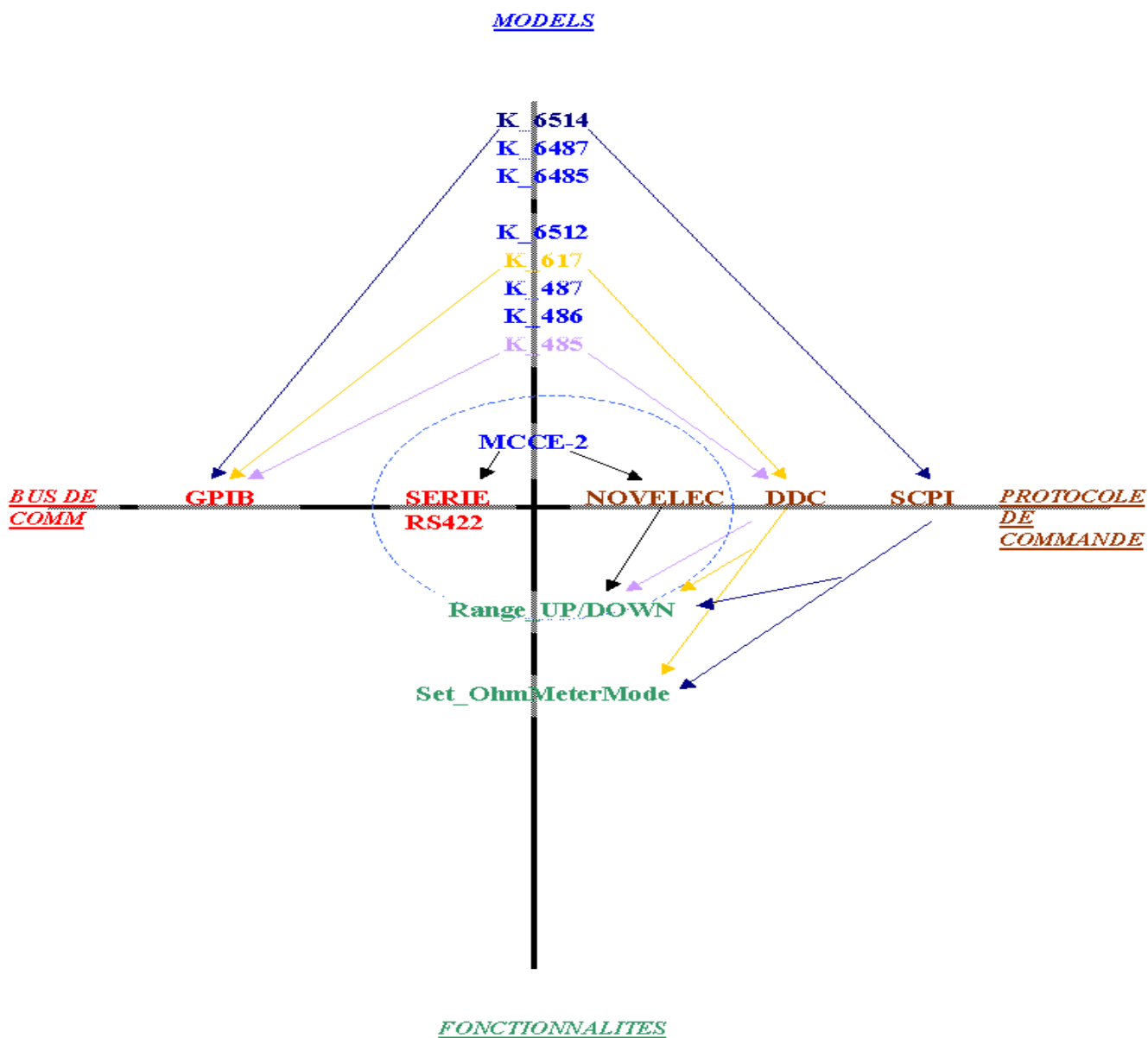
4.2 Summary of functions and protocols for all instruments to interface

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Model	Standards de	Bus de	Fonctions de mesures													
2		Protocole de commandes	communication	AnaOutPut	VoltageSrc	AutoZero	DIO		PicoAmpere	Voltmeter	OhmMeter	CoulombMeter	ExternalFeedback	FiltersType	Triggers	Fonctions supplémentaires	
3	Keithley																
4	485	DDC	GPIO	No	No	No	No	Yes	No	No	No	No	No	No	T0 T5		
5	486	DDC	GPIO	+2V	?	No	No	Yes	No	No	No	No	No	Digital Analog	T0 T9		
6	487	DDC	GPIO	+2V	?	No	No	Yes	No	No	No	No	No	Digital Analog	T0 T9		
7	617	DDC	GPIO	?	+102V	No	No	Yes	Yes	Yes	Yes	Yes	No	No	T0 T7		
8	6485	SCPI	GPIO	+2V	?	Yes	No	Yes	No	No	No	No	No	Median Digital +Adv	IN OUT		
9	6487	SCPI	GPIO	?	+505V	Yes	Yes	Yes	No	Yes	No	No	No	Median Digital	IN OUT	Sweep Voltage	
10	6512	DDC	GPIO	+2V	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	T0 T7		
11	6514	SCPI	GPIO	+2V	?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Median Digital	IMM TLINK		
12																	
13	Novelec																
14	MOCE2	Novelec	RS422	No	No	No	?	Yes	No	No	No	No	No	Freq coupure filtre	No	Polarité, Gain	
15																	

Figure 1 : Models with supported functions and protocols

4.3 Another view of the problem

This view shows the difficulty to have an homogenous interface. Each model has its “Commands Protocol” on a specific “Communication Bus” and supports different commands.

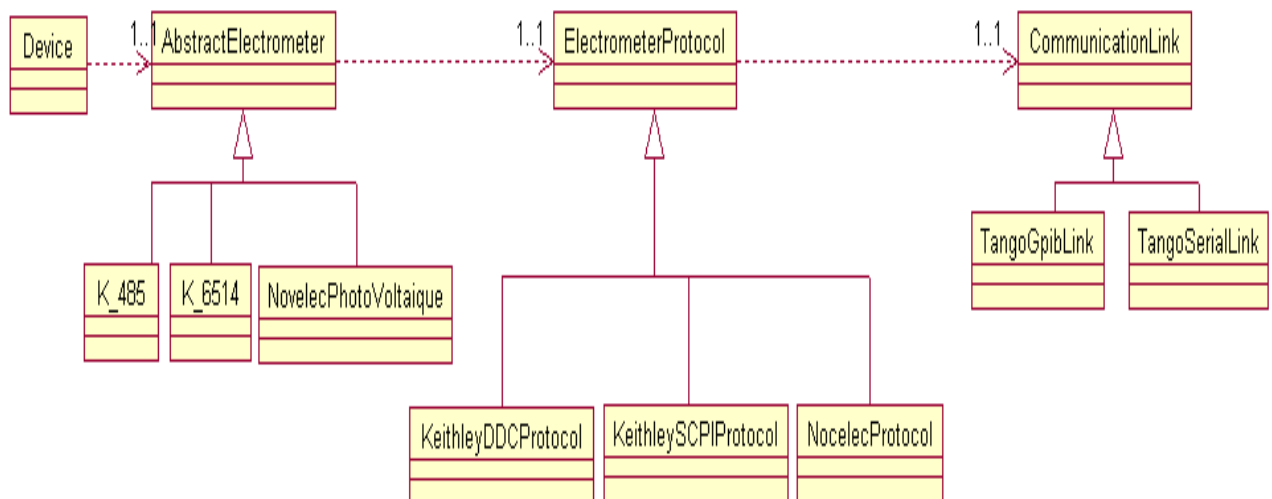


4.4 UML analysis model

The previous representations highlight

- the difficulty to share a maximum of functionalities between modes (*for instance, ohmmeter functions are supported only by a few models (K_617, K-6487, ..) and moreover with different protocols*).
- and in the same time shows up the main components of the library architecture:
 - **3 base classes** representing instrument model (*AbstractElectrometer class*), protocol (*ElectrometerProtocol class*) and communication bus (*CommunicationLink class*)
 - **one concrete** class per instrument model (*deriving from AbstractElectrometer class*)
 - **one concrete** class per command protocol (*deriving from ElectrometerProtocol class*)
 - **one concrete** class per communication bus (*deriving from CommunicationLink class*)

This architecture is illustrated by the following UML analysis model



5 Base Classes and responsibilities

5.1 AbstractElectrometer class

This class is the visible interface of the library for the TANGO device. This class provides a default implementation for each function supported by electrometers. There is one specific concrete subclass for each instrument model. (for instance *K_485* for Keithley 485 models) One object of this class is in relation with an *ElectrometerProtocol* object

5.2 ElectrometerProtocol class

Object of this class knows only the command protocol and nothing about the electrometer type !

One object *ElectrometerProtocol* is in relation with a *CommunicationLink* object

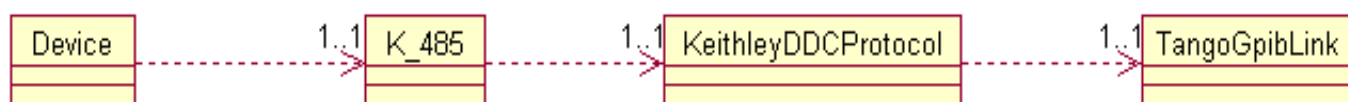
5.3 CommunicationLink class

This class is in charge of sending and receiving data through the specific communication bus. Supported buses are till now RS232, RS422 and GPIB

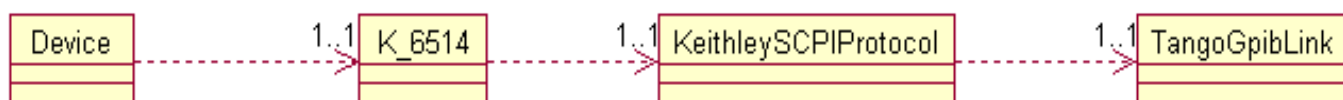
5.4 Examples of collaboration diagram: how all that is instanciated

5.4.1 For a Keithley 485

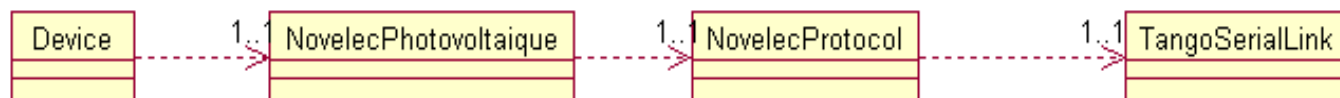
As shown in the above figure, a Keithley 485 uses the DDC protocols over a GPIB link, which leads to the following instantiation diagram.



5.4.2 For a Keithley 6514



5.4.3 For a Novelec MCCE-2



5.4.4 Note on Instantiation mechanism

The instantiation process is well known and so simple, that we decide not to implement) for the moment a Factory (*in the design pattern meaning !*) . In fact each object instantiates what it needs (for example **K_485** creates the **KeithleyDDCProtocol** object it needs, which in turns instantiates the **TangoGPIBLink** it needs to communicate with hardware).

We do not need (till now !) to build, for example, a K_485 object communicating with a Serial interface (*even if it may be physically possible*). If such combination are required, a Factory should be implemented to manage this more complex instantiation process.

6 Detailed Library design

6.1 Preliminary note

Library is fully documented by Doxygen. Detailed description of functions and class should be found there.

6.2 AbstractElectrometer class

6.2.1 Implementation of default behavior

This is the visible interface of the Electrometers library for the TANGO device. For each function , it implements the default behaviour for the majority of the models:

- For instance, if **most** electrometers models/types **support** a functionality, this one is **implemented** in the **AbstractElectrometer** class and an **ElectrometerException(COMMAND_NOT_SUPPORTED)** has to be thrown by the few concrete electrometer subclass(es) which do not support this functionality
- In the other case , if a functionality is supported by a **minority** of electrometers the **ElectrometerException(COMMAND_NOT_SUPPORTED)** is thrown by the **AbstractElectrometer** class and the functionality has to be implemented in the **minority** concrete class(es) that support it.

Important note : defining what is the “default behavior” is often not intuitive because for many functionalities, 50% of electrometers support them and the 50% not. In this case, the adopted philosophy is to throw the exception in the **AbstractElectrometer** class and the implementation is done in the concrete classes that support it !

6.2.2 Functions supported only by one protocol

Some functions are supported only by one protocol and do not depend on the electrometer type. In this case were no logical switch has to be done depending on the electrometer model, **AbstractElectrometers** propagates the methods calls to the **ElectrometerProtocol** class.

6.2.3 Particular cases

6.2.3.1 auto_range_off method:

Even if K_486,K_487 K_617 & K_6512 use the DDC protocol, they use different commands (“R10” or “R12”) for this function and the K_485 does not support it, so the implementation as been done like this :

- in K_485 subclass there is no implementation, → **AbstractElectrometerClass** implementation is used
- **AbstractElectrometerClass** propagate the `auto_range_off()` call to the **KeithleyDDCProtocol** class
- K_486 and K_487 propagates the call to the `autoRange_OFF_forK486_487()` method of the **KeithleyDDCProtocol** class
- Same mechanism for K_617 & K_6512
- in **KeithleyDDCProtocol** class, `auto_range_off()` command throws an exception by default and there is a specific implementation for K_486 & K_487 (named ‘`autoRange_OFF_forK486_487()`’) and an other for K_617 & K_6512 (named ‘`autoRange_OFF_forK617_6512()`’).

6.2.3.2 GetElectroMeterGain

this command is only supported by Novelec PhotoConducteur type so :

- a) the concrete class **N_PhotoConducteur** class propagate the call to **NovelecProtocol** class
- b) for the other electrometers an exception is thrown in **ElectrometerProtocolClass**

6.3 ElectrometerProtocol Class

This class knows the three supported protocols (Gpib : SCPI & DDC and Novelec protocol) and sends/receives commands and device response through a the specific CommunicationLink object.

The device type is unknown here ; only the protocols specifics functionalities are implemented

If a functionality is not supported by a protocol, an ElectrometerException is thrown.

6.4 CommunicationLink class

This class manages the communication through the desired bus, GPIB bus or Serial bus.

There is a dependency with TANGO , which means that if you want to manage the communication with no dependency with TANGO you must replace the **TangoGpibLink** and **TangoSerialLink** with your own classes.

However, to replace these classes, it is mandatory to respect the **CommunicationLink** abstract class interface.