# LHC GCS
# A framework for the production of 23 homogeneous control systems

G.Thomas[1], K. Azarov[1], R. Barillère[1], S. Cabaret[2], N. Kulman[1], X.Pons[1], J. Rochez[1],
*[1]CERN, Geneva, Switzerland, [2]UPJV / LMBE-ESIEE, Amiens, France – CERN, Geneva, Switzerland.*

## Problem Statement

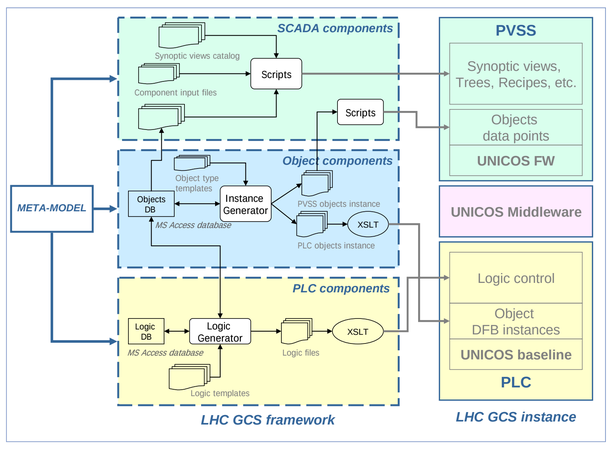
The LHC experiments' Gas Control System (LHC GCS) aims to provide the LHC experiments with homogeneous control systems (supervision and process control layers) for their 23 gas systems. To ease the production of these control systems, it has been decided to develop a library of components, the LHC GCS framework, and to adopt a model-driven automatic code generation approach.

The LHC GCS instances are turn-key control applications which provide end-users with a consistent look and feel. As they must be developed and maintained with a small team, it has been decided to first produce libraries and tools, the LHC GCS framework, and then to develop all instances from this framework.

## Strategy

The strategy consisted of selecting industrial tools and technologies for the implementation of all layers of the LHC GCS instances: A Supervision Control And Data Acquisition (SCADA) system for the supervision layer (ETM's PVSSII), Schneider Programmable Logic Controllers (PLC) for the process control, standard protocols for the middleware and field buses for the access to the devices.

UNICOS , a framework to develop industrial applications, has been identified as a tool offering solutions to most of the LHC GCS requirements.




## Architecture

The core of the LHC GCS framework is the Object components which are based on a common UNICOS tool (Instance Generator) to which LHC GCS types have been added. These types have been designed according to the UNICOS patterns.

The LHC GCS framework SCADA components ease the development of PVSS features such as synoptic and trend views, trees of views, etc. They depend on the Object components. They consist of inputs files which describe in detail the pieces of information to handle and scripts which create the PVSS displays and data points from these files. The input files can be produced manually (with text or XML editors) or can be extracted automatically from databases.

The LHC GCS framework PLC specific components are based on the UNICOS tool (Logic Generator), they help in the production of the application-specific PLC code (e.g. closed loop controls, interlock handling, etc.). They consist of generic source files which are pre-compiled for the PLCs.

## Conclusion

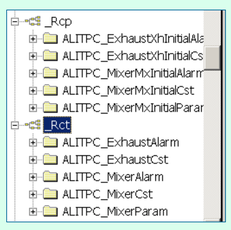
A first version of the LHC GCS framework implementing all necessary functionality has been released. This version has been used to produce the first of the 23 LHC GCS instances. The LHC GCS framework allows time saving in the generation process of a control application. In addition it guarantees homogeneity of the code produced for all LHC GCS instances. By replacing a lot of interactive configuration phases, it reduces the number of configuration errors. Although they are used in combination with a model-driven approach for the LCH GCS project, most of the framework components can be re-used in other UNICOS-based control projects.

# SCADA Components

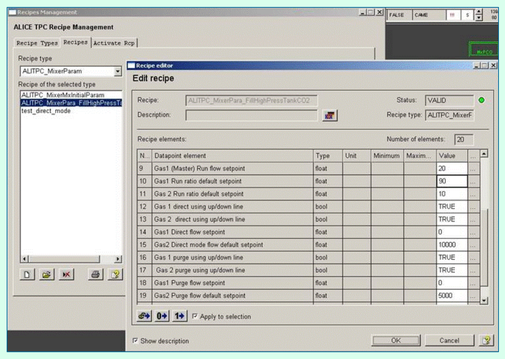
## Recipe component

```
ModuleType=Mixer;GCSPrefix=ALITPC;RecipeType=Param;RecipeName=Mx]
RecipeElement=Mx_L1FillUp|1;RecipeElement=Mx_L2FillUp|1;RecipeEle
ModuleType=Mixer;GCSPrefix=ALITPC;RecipeType=Alarm;RecipeName=Mx]
RecipeElement=Mx_L1NotRelHH|105.00;RecipeElement=Mx_L1NotRelH|103
RecipeElement=Mx_L1NotRelLL|0.50;RecipeElement=Mx_L1NotRelDt|30;R
RecipeElement=Mx_L2NotRelH|103;RecipeElement=Mx_L2NotRelL|0.80;Re
RecipeElement=Mx_L2NotRelDt|30;RecipeElement=Mx_L1PressL|1;Recipe
ModuleType=Mixer;GCSPrefix=ALITPC;RecipeType=Cst;RecipeName=MxIn
#ModuleType=Pump;GCSPrefix=ALITPC;RecipeType=Param;RecipeName=Pp]
```

Input file –Recipe parameters of a LHC GCS instance (Alice TPC)




Recipes instantiated in PVSS DB




Recipe Management panels

The component has been implemented using PVSS standard technologies. For the automatic instantiation of the recipe elements in PVSS, a configuration panel has been implemented to gather the input file (file describing the recipes of a LHC GCS instance) and scripts written in the PVSS scripting language to process the input file and create the data point elements in the PVSS database. The input file is coded in ASCII CSV format. The file can be written manually or automatically extracted from a database.

The PVSS panels provided have been used for recipe editing. The PVSS activation panels have been customized to deal with the xPAR download mechanism and to display feedback information. The download mechanism has been written in the PVSS script language.

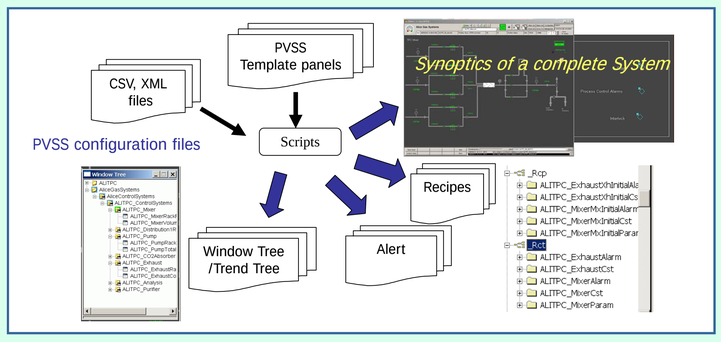
## Purpose

The purpose of the SCADA components is to replace the interactive configuration phase of the supervision layer.

A typical LHC instance such as Alice TPC (6 gas modules) contains ~20 synoptic views, 15 Anomaly views, 3 Recipe types per module (with ~ 100 parameters per type), 1 Window/trend tree (7 nodes), and 1 Alert summary configuration per PCO.

## Principle

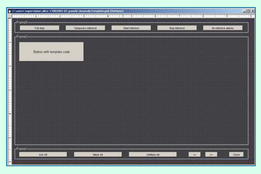
All components are built upon a similar principle. They consist of scripts which produce displays and/or data points in the PVSS DB from LHC GCS instance-specific input files (e.g. synoptic view templates, objects description files). The input files are created manually or can be extracted from a database.



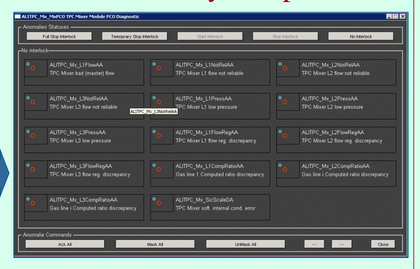

- The synoptic view components allow the production of similar but not identical human user interfaces.
- The trend view component produces all trend views of a LHC GCS instance.
- The view trees component configures the hierarchical organization of the synoptic and trend views of an LHC GCS instance.
- The recipe component provides tools for the configuration and operation of recipes of an application.
- The anomaly component is a solution to build synoptic views displaying the alarm objects which can raise the interlock of a given higher-level object of the control application.
- The alert summary component allows the configuration of PVSS alert summaries for these high level objects.

## Anomaly component

Input file -List of Anomaly Objects



Input -PVSS template file panel



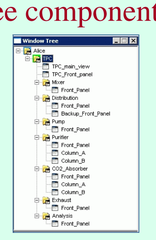

Output file – Alice TPC Mixer Anomaly view

This component is implemented by means of PVSS scripts which read a file in XML format listing the alarm objects and their relations with the PCOs. At configuration time the scripts add the widgets in the appropriate windows of the panel and adjust its size.

## Window Tree component

```
Alice|1|PANEL|Alice_frontpanel.pnl|
TPC|4|PANEL|ALITPC/ALITPC_SystemOverview.pnl|
TPC_main_view|0|PANEL|ALITPC/ALITPC_System.pnl|
TPC_Front_panel|0|PANEL|ALITPC/ALITPC_SystemRackFrontPanel.pnl|
Mixer|1|PANEL|ALITPC/ALITPC_mixer.pnl|
Front_Panel|0|PANEL|ALITPC/ALITPC_mixerRackFrontPanel.pnl|
Analysis|1|PANEL|ALITPC/ALITPC_AnalysisChain1.pnl|
Front_Panel|0|PANEL|ALITPC/ALITPC_AnalysisChain1FrontPanel.pnl|
```

Input file – Alice TPC Window Tree structure

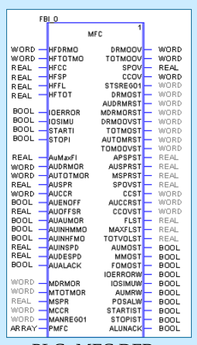



PVSS WT display

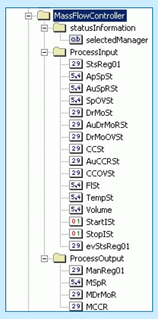
The principle relies on input files describing the tree hierarchies and scripts to translate the information of the input files and write it to the corresponding PVSS data points. The input file can be written manually or generated from a database and contains the information about the tree structure (node and children relationships) of a given LHC GCS instance. Then standard PVSS scripts and JCOP framework functions are used to parse the file and create the data points in the PVSS database. The standard UNICOS active X window is then used to display and navigate through the trees.

# Object Components

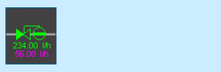
## Object class in PLC and PVSS



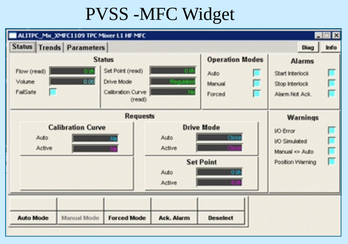

PLC -MFC DFB




PVSS -MFC DPT




PVSS -MFC Widget




PVSS -MFC Faceplate

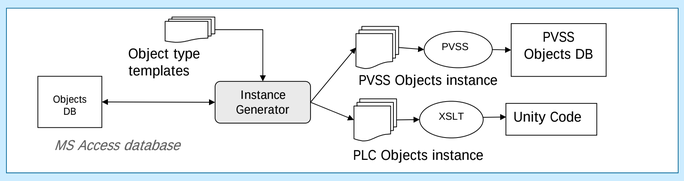
These proxies are represented by means of icons (widgets) and operated by means of dedicated displays (faceplates); they have been developed for each class of objects. A communication layer implements the exchange of object status and commands between PLCs and PVSS. In order to have consistent PLC and PVSS layers, they are generated from a unique database using a data-driven tool, the UNICOS Instance Generator.

## Purpose

The UNICOS framework Object components consist of libraries to implement many types of objects (e.g. I/O, field objects, and high level control objects). Each of the objects is evaluated in the PLCs and has a proxy in the PVSS layer.

## Principle

New classes of objects have been introduced for the LHC GCS applications. They deal with specific gas devices (e.g. Mass Flow Controllers) and with specific functionality (e.g. xPar objects for recipes).




## GCS object classes extension

- MFC (Mass Flow Controller)
- Pump
- AA/DA (Analog/Digital Alarm)
- xPar (Analog/Digital/Word Parameters)
- xS (Analog/Digital/Word Status)
- xC (Analog/Digital/Word Computed)

## Instance Generator template files

```
{Info(""MFC"")}
(* $Description$ *)(* A PLC init does always reset all structures, therefore we must initialize it again *)
{IIF (""$Process Input ParReg$"" = """", ' (* ParReg set by logic*)', '$Equipment$_$Location$_$Name$_PMfc.PArReg :=
$Process Input ParReg$;')}{IIF (""$Process Input PAlTiDe$"" = """", ' (* PalTiDe set by
logic*)','$Equipment$_$Location$_$Name$_PMfc.PAlTiDe := t#$Process Input PAlTiDe$s;')}
{IIF (""$Process Input PMInSpd$"" = """",' (* PMinSpd set by logic*)',
'$Equipment$_$Location$_$Name$_PMfc.PMinSpd := $Process Input PMInSpd$;')}
```

Instance Generator –MFC Unity template

```
MassFlowController;{21000 + $Record Number$};$Equipment$_$Location$_$Name$;$Description$;$Diagnostics$;
$Html Info$;$Synoptic$;$Domain$;$Nature$;$Instance Type$;$Setpoint Unit$;$Setpoint Format$;$Setpoint Range
Max$;$Setpoint Range Min$;$Flow Unit$;$Flow Format$;{IIF($SetPoint Flow Archive Active$,"Y;$SetPoint Flow Ar-
chive Sampling$","N;0")};$Volume Unit$;$Volume Format$;{IIF($Volume Archive Active$,"Y;$Volume Archive Sam-
pling$","N;0")};$Dead Band Value Scada$;$Dead Band Type Scada$;$Normal Pos$;$StsReg01$;$evStsReg01$;
```

Instance Generator –MFC PVSS template

**MassFlowController; ObjectNumber; Alias; Description; Diagnostic; Html; DefaultPanel; Domain; Nature; Widget**; setpointUnit; setPointFormat; setpointRangeMax; setpointRangeMin; FlowUnit; FlowFormat; VolumeUnit; VolumeFormat; Deadband; DeadbandType; ArchiveActive; ArchiveTimeFilter; NormalPosition; StsReg01; evStsReg01; ApSpSt; AuSpRSt; SpOVSt; DrMoSt; AuDrMoRSt; DrMoOVSt; CCSt; AuCCRSt; CCOVSt; MaxFlSt;

PVSS –MFC CSV Format

In addition template files and libraries have been developed for each object class to enable their automatic generation by the Instance Generator and the PVSS scripts.

# PLC Components

## Object-include file relationships

| id | Application | Plc Nb | Equipment | Location | Name | Type | Master | File | Parameter1 |
|---|---|---|---|---|---|---|---|---|---|
| 64 | ALITPC | 0 | ALITPC | Di | SEQ | 14 | DiRack61PCO | DiRackiPCO/DiRackiPCO_SL.xfm | 61 |
| 65 | ALITPC | 0 | ALITPC | Di | GT | 15 | DiPCO | DiPCO/DiPCO_TL.xfm | 61 |
| 66 | ALITPC | 0 | ALITPC | Di | GT | 15 | DiRack61PCO | DiRackiPCO/DiRackiPCO_TL.xfm | 61 |
| 67 | ALITPC | 0 | ALITPC | Di | GL | 16 | DiPCO | DiPCO/DiPCO_GL.xfm | 61 |
| 68 | ALITPC | 0 | ALITPC | Di | GL | 16 | DiRack61PCO | DiRackiPCO/DiRackiPCO_GL.xfm | 61 |
| 69 | ALITPC | 0 | ALITPC | Di | CDOL | 17 | DiPCO | common/xx_CDOL.xfm | |

Logic Objects database

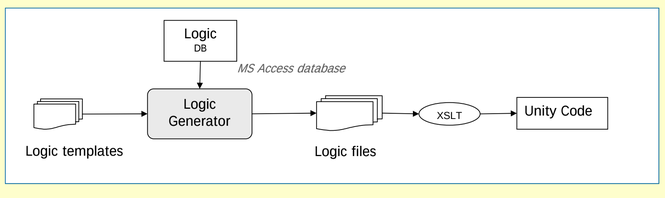
```
<IncludeFile>
        <sectionDesc name="$Equipment$_$Location$_$Master$_$Name$" FMName="$Equipment$_$Location$_$Master$"/>

        <program>
                <identProgram name="$Equipment$_$Location$_$Master$_$Name$" type="section" task="MAST"/>
        <comment>
        </comment>
        <STSource>


(* Invocations *)

$Equipment$_$Location$_TIME$Parameter1$WAIT05 (IN := StartIndPurge$Parameter1$05_$Equipment$_$Location$.x, (* BOOL *)
                PT := T#1000ms(* TIME in ms*));

$Equipment$_$Location$_TIME$Parameter1$WAIT15 (IN := StartComPurge$Parameter1$15_$Equipment$_$Location$.x, (* BOOL *)
                PT := T#1000ms(* TIME in ms*));

$Equipment$_$Location$_TIME$Parameter1$WAIT25 (IN := StartingRun$Parameter1$25_$Equipment$_$Location$.x, (* BOOL *)
                PT := T#1000ms(* TIME in ms*));
```

Logic template file (include file)

An include file typically contains variables to handle object names; it can then be used for several gas systems. For instance when a device is fitted for a given purpose (e.g. the pump bypass valve), in these gas systems, it is driven according to the same principle. When a unique file can not be used, one can produce several files and associate them appropriately with the objects.

## Purpose

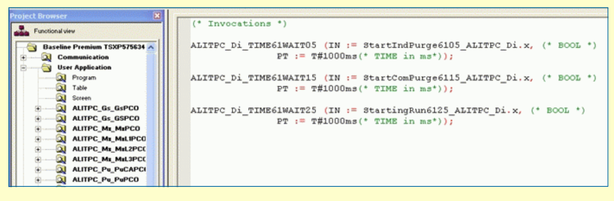
In addition to the libraries implementing the LHC GCS classes in the PLC, a component has been developed to ease the development of the application-specific code required for the closed-loop controls, the interlock detection, etc.




## Principle

The basic principle of the PLC logic component is then to produce a set of re-usable files, establish a relationship between these files and the field objects of a LHC GCS instance and let a tool gather them to build the full PLC code according to the UNICOS PLC code structure. A file contains the logic required to drive a low level object of one of the 7 routines of a PCO.

## LHC GCS instance logic code (Unity)




The component has been implemented using standard technology. The files are in XML format embedding PLC code (e.g. the standard Structure Text PLC language) and Visual Basic functions. The tool is based on a Microsoft Access database in which the objects of a LHC GCS instance are associated with files. The parameters represent the properties of the relationships.

This database can be populated manually or by high level generators. Visual Basic scripts pre-compile the files to produced intermediate XML files. The test conditions and queries are evaluated during this phase. Finally the files produced are combined into a single XML file using the XSLT transform which is then imported in the PLC development environment.

URL: http://itcofe.web.cern.ch/itcofe/Projects/LHC-GCS/welcome.html