An innovative telescope control system architecture for SST-GATE telescopes at the CTA Observatory

Gilles Fasola[*a], Shan Mignot[a], Philippe Laporte[a],
Abdel Abchiche[b], Gilles Buchholtz[b] and Isabelle Jégouzo[a]

[a]Galaxies, Étoiles, Physique et Instrumentation (GEPI) Observatoire de Paris – CNRS – Université Paris-Diderot, 5 place Jules Janssen, 92190 Meudon, France; [b]Division Technique de l'INSU (DT-INSU), CNRS, 1 place Aristide Briand, 92190 Meudon, France

ABSTRACT

SST-GATE (Small Size Telescope - GAmma-ray Telescope Elements) is a 4-metre telescope designed as a prototype for the Small Size Telescopes (SST) of the Cherenkov Telescope Array (CTA), a major facility for the Very High Energy gamma-ray astronomy of the next three decades. In this 100-telescope array there will be 70 SSTs, involving a design with an industrial view aiming at long-term service, low maintenance effort and reduced costs. More than a prototype, SST-GATE is also a fully functional telescope that shall be usable by scientists and students at the Observatoire de Meudon for 30 years.

The Telescope Control System (TCS) is designed to work either as an element of a large array driven by an array controller or in a stand alone mode with a remote workstation. Hence it is built to be versatile and autonomous; as an example, pointing and tracking – the main functions of the telescope – are managed on-board, including astronomical transformations, geometrical transformations (e.g. telescope bending model) and drive control. The core hardware is a Compact-RIO (cRIO) featuring a real-time operating system and an FPGA.

In this paper, we present an overview of the current status of the TCS. We especially focus on three items: the pointing computation implemented in the FPGA of the cRIO – using CORDIC algorithms – since it enables an optimisation of the hardware resources; data flow management based on OPC UA with its specific implementation on the cRIO; and the use of an EtherCAT field-bus for its ability to provide real-time data exchanges with the sensors and actuators distributed throughout the telescope.

Keywords: TCS, FPGA, CORDIC, OPC UA, EtherCAT, cRIO, CTA, SST-GATE, 4-metre telescope

1. INTRODUCTION

SST-GATE, part of the GATE project, is a prototype for the SSTs of CTA. Its purpose is to evaluate the performances of the previously untried Schwarzschild-Couder mirror configuration. We have decided to take the advantage of this opportunity to experiment several options in the TCS design.

For this design, several types of constraints have to be taken into account. Firstly, we have to build a TCS with all the required functions to fulfil the primary goal of the prototype, i.e. evaluate the new opto-mechanical configuration. Secondly, we have to ensure that the TCS will fit the local operating requirements in Meudon, for technical interfacing and as an outreach facility as well. Finally, we have to design a TCS that is going to fit easily into the CTA array control scheme.

We describe, in section 2, an overall presentation of the TCS: the hardware setup, the communications inside and outside the telescope and the software characteristics. The major innovative feature stands in the pointing computation implemented in an FPGA; we present this aspect in section 3. In collaboration with the other teams involved in the design of the telescopes of the CTA project – Medium Size Telescopes (MSTs) and Large Size Telescopes (LSTs) –, we have chosen OPC UA as the hardware connectivity solution; we develop this part in section 4. All the field communications rely on the EtherCAT bus; we detail this in section 5.

* E-mail: gilles.fasola@obspm.fr

2. OVERALL DESIGN OF THE TCS

SST-GATE will host a scientific camera built by the Compact High Energy Camera (CHEC) development project. The TCS will provide the CHEC camera mainly with hardware support; a dedicated fibre bundle will convey the actual data transfer and control required by the camera to the operator workstation located in the control room.

Therefore, the primary function of the TCS is driving the telescope axes for pointing and tracking purposes. The drive subsystem gathers all the drive related software modules and hardware devices; it receives remote high-level orders – from a workstation or the array controller – and all computations are made on-board.

Many other functions are required to run the telescope in a proper way; these are collected in several subsystems (see Figure 1), hence each of these is composed of either hardware and associated software modules, or software modules alone.

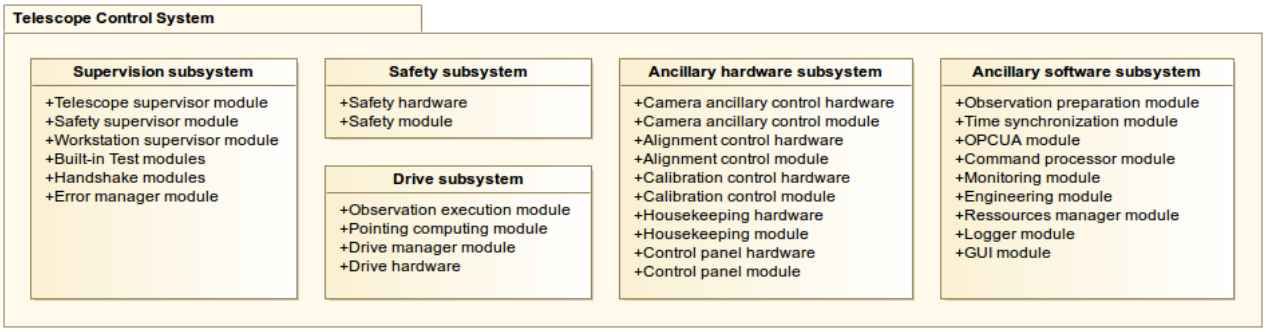

Figure 1. The five subsystems of the TCS: supervision, safety, drive, ancillary hardware and ancillary software

2.1 Hardware outline

To control the prototype, the TCS software is distributed amongst the hardware on board the telescope and the workstations of the control room (see Figure 2).

We use two PLCs within the telescope: the *main PLC* and the *safety PLC*. The safety PLC is in charge of the automation concerning safety, mainly to prevent hazards due to the motion of the telescope. The main PLC is the core device and controls all other functions.

The safety hardware setup is based on a TwinSAFE system by Beckhoff. The hardware modules, sensors and actuators used to perform the safety functions are distributed throughout the telescope.

The main PLC is a CompactRIO from National Instruments (NI); it features a real-time operating system (OS) and an FPGA. This FPGA is used to manage the connected I/Os – as a back-plane controller – and is also available for user-specific computation. Besides CompactRIOs have a set of slots to plug I/O modules directly into their back-plane, they can handle daisy-chained *extensions* via an Ethernet-based bus (Ethernet, EtherCAT or MXI-Express); this is a key feature to distribute the I/Os throughout the telescope with only one network cable. Each extension features an FPGA to achieve both the network communications and the back-plane functions; still, a lot of room is available for user computations.

This distribution of the I/Os via a network enables a much more efficient cabling system by limiting the analog part of the cabling to its minimum; this results in a better electromagnetic immunity, an easier cabling design – fewer cables leading to narrow cable paths –, and saves weight and money.

The choice of a cRIO as main PLC is driven by several points, among which:

- real-time operating system
- integrated FPGA
- versatility and accuracy in measurement
- low-power system
- long term support of National Instruments
- common software environment (LabVIEW, C or C++)
- harsh operating conditions

As previously said in the introduction, we experiment several options in the development of the TCS; most of them are related to the features of cRIOs and LabVIEW – real-time, FPGA, EtherCAT field-bus master, OPC UA. Beyond these, the main axis is to optimise the use of the hardware resources and figure out what is the best we can get from cRIOs. Seeking such an optimisation aims to fit the needs of a "mass production" – with dozens of SSTs in the CTA array – of telescopes: low unit cost, low power requirements... A consequence is the need to select, for the main PLC, the best candidate between the cRIO9074 and the cRIO9068.

2.2 Communication outline

For all local communications, we have chosen the EtherCAT field-bus, making a coherent package with the Beckhoff modules and the CompactRIOs' extensions with the EtherCAT flavour. As previously mentioned, this is described in section 5.

All remote communications are handled via an Ethernet connexion. On the logical point of view, the hardware connectivity is managed with a set of OPC UA servers and clients (see section 4 for more details).

Figure 2 shows the only two physical communication links required between the telescope and the control room: the Ethernet cable and the fibre bundle for the CHEC camera.

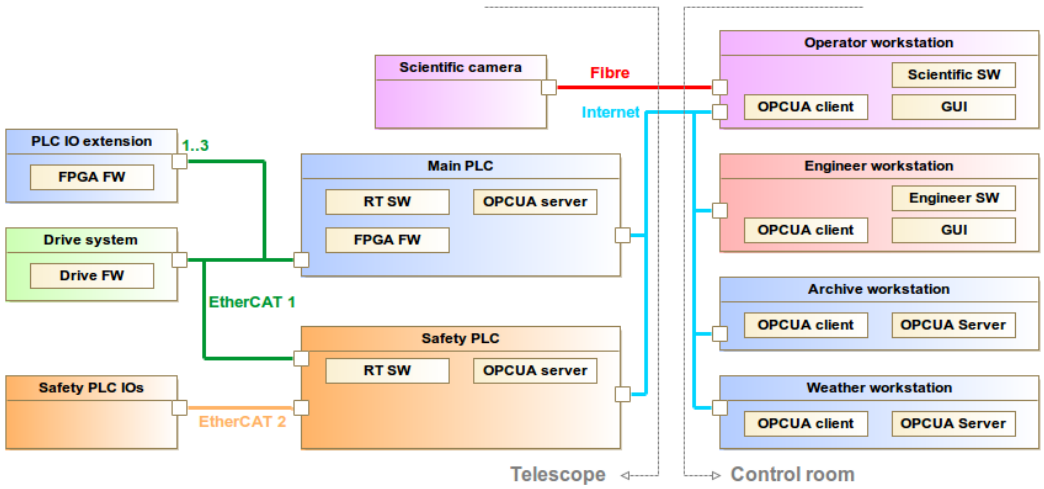

Figure 2. An overview of the software and the communication layout of the TCS. There are only two physical links between the telescope and the control room for communication purposes. A dedicated EtherCAT field-bus is used for safety. The logical connectivity is managed by OPC UA clients and servers. HW stands for hardware, SW for software and FW for firmware.

2.3 Software outline

To run the telescope, the software has to be distributed throughout local and remote locations to address all the required functional purposes. Different kinds of software are developed for the TCS. These depend on the target itself (a PLC, a workstation, a server), its OS and the type of execution (real-time, asynchronous, etc.).

A real-time software – an isochronal system in our case – is required for the tracking purpose of the telescope. The main PLC, hosting the pointing and tracking computation, is run with a real-time OS. The cRIO9074 is shipped with a VxWorks OS, whereas the new cRIO9068 comes with a Real-time Linux-based distribution prepared by NI. The LabVIEW programming environment allows the integration of C language via specific *nodes* in the LabVIEW code. Moreover, the LabVIEW environment is not required anymore to work with a cRIO9068; it is also possible to use any C/C++ software development environment. This allows a significant flexibility while fitting the habits of the programmer's team with the main programming language; this difference is a major criterion between the cRIO9074 and the cRIO9068.

A firmware can be designed and uploaded with the LabVIEW environment to run the available FPGA on cRIOs. As with the real-time part, the use of a foreign language is possible: VHDL in that case. However, it has to be noted that the LabVIEW FPGA module is required for both the considered cRIOs.

TwinCAT, the Beckhoff's environment, is used to design the safety software.

3. POINTING

The purpose of the SST-GATE telescope is to collect the Cherenkov photons emitted by the shower of secondary particles produced when a high energy particle enters the atmosphere. This requires, as for all telescopes, the pointing and the tracking of the emitting source to be observed.

3.1 Pointing computations

To achieve pointing, the equatorial coordinates of a source have to be converted into altitude-azimuth – or horizontal – coordinates, taking into account the observatory location and the date and time. For tracking, the conversions have to be computed in real-time to allow accurate scientific observations.

There are two kinds of transformations: the astrometrical ones, to take into account the celestial motions, and the geometrical ones, to consider the deformation of the telescope. All these share the use of trigonometric functions like sine, cosine, and their inverse, i.e. arc sine and arc cosine.

3.2 Hardware considerations

In the seek of finding an optimised real-time system, we have decided to use the available FPGA on cRIOs as a coprocessor. Besides FPGAs are obvious components for real-time and contribute significantly to low-power requirements, this way offers other benefits as shifting computation from the CPU to the FPGA releases the CPU charge and the memory use. The extra cost is an additional interface to fit the CPU rates and variables to the FPGA ones.

3.2.1 CORDIC algorithms

FPGAs are low-level electronic components: they handle zeros and ones more easily than π. However, in 1959, Jack E. Volder invented the COordinate Rotation DIgital Computer[1] (or CORDIC) for real-time airborne computation. CORDIC is based on a simple algorithm of shifting and adding binary numbers. Amongst the numerous computing tasks it can perform, there is the calculation of trigonometric functions, and their inverse as well. Nowadays, the base algorithm has been enhanced in several ways, especially since the appearance of the FPGAs.

The LabVIEW FPGA module provides "High throughput math functions"; amongst which sine, cosine and inverse tangent – all CORDIC-based. Unfortunately, arc sine and arc cosine are not available in this palette.

3.2.2 Implementation strategy

To get round the lack of these functions, we have decided to build a CORDIC firmware and implement it into the FPGA of the cRIO. A VHDL model, based on the algorithm described in Lang's article,[2] has been first written, simulated and optimised.

3.2.3 A CORDIC VI

TBDevelopped

3.3 Overall architecture

TBDevelopped

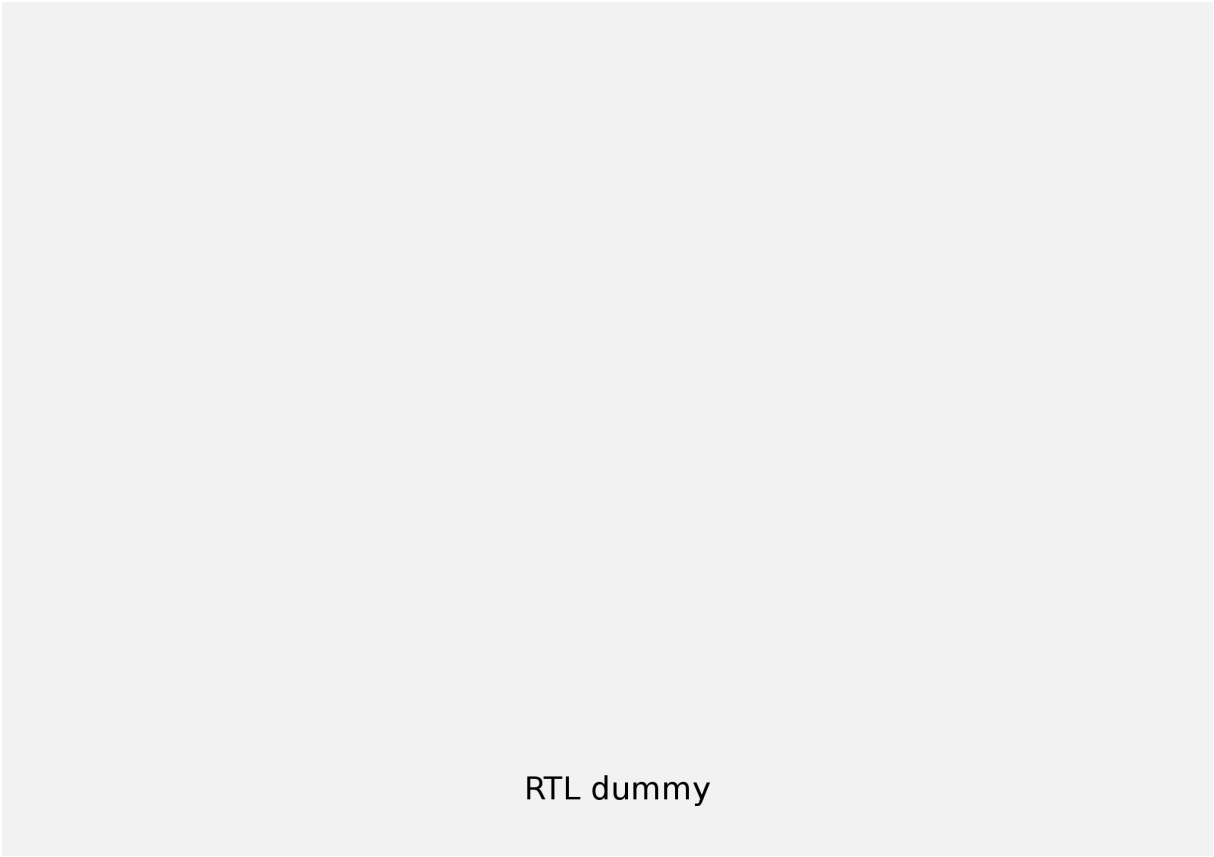

Figure 3. The dummy RTL image...

4. OPC UNIFIED ARCHITECTURE

OPC United Architecture (OPC UA) is an object-oriented platform-independent publish/subscribe protocol commonly implemented with a client-server pattern. Its main feature is *Data Access* with time-stamping and data quality tagging; it may also provides a system with services such as secured transmissions, *Historical Access* or *Alarms and Conditions*.

The base object in OPC UA is called a *node*. As shown in Figure 4, a node has mandatory and optional attributes. Some nodes are used to create links – or *references* – between other nodes; the whole resulting hierarchized structure is called the *address space*.

Detailed information on OPC UA is available on the Internet site of the OPC foundation.[3]

4.1 OPC UA in the CTA project

The numerous devices of the CTA telescopes have to be controlled and monitored; moreover, they are of various types. OPC UA has been chosen by CTA as the software communication layer standard to release the constraints related to the physical layer during the selection process of the sensors and actuators.

At the telescope array control level, the main software environment will be managed with the ALMA Control Software (ACS) and OPC UA will be used for low-level device communication. Therefore, only the *Data Access* service of OPC UA will be used in the CTA project. Later, secured transmissions may also be implemented.

4.2 OPC UA in the SST-GATE prototype

As shown in Figure 2 we have two local OPC UA servers in the telescope: one dedicated to safety and the other one for all other purposes. Some remote OPC UA servers are implemented in the workstations of the control

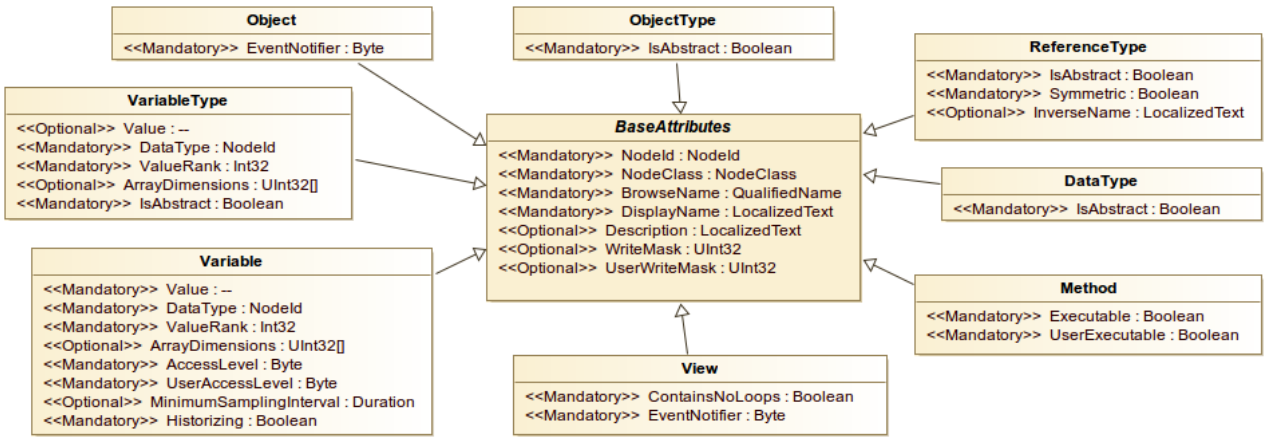

Figure 4. Several types of OPC UA nodes are defined; all sharing the general definition of the *BaseAttributes* node.

room to allow the user (the *Engineer* or the *Operator*) to consult, when the telescope is not in an operating state, previously archived data or the current weather conditions, for instance. On regular operating conditions, these workstations will behave as clients to provide the telescope's servers with up-to-date values or archive the monitored nodes.

On the SST-GATE prototype in Meudon, Data Access is the only OPC UA service implemented.

4.2.1 The main OPC UA server

An OPC UA software development kit is shipped with two of the LabVIEW modules: the Data-logging and Supervisory Control (DSC) module and the Real-Time module. This kit provides a whole set of functions to implement the Data Access feature of a client-server OPC UA protocol.

These functions allow to:

- create a server
- create an address space with *folders*, *items* and *properties*
- start and stop a server
- read or write node values, as server, at a given rate with time-stamping and data quality tagging (*Status*)
- create a client
- create a –certified, if required– connection between a client and a server
- browse the address space of a server
- create a subscription
- add monitored nodes to a subscription
- read or write node values, as client, at a given rate with time-stamping and data quality tagging

It has to be noted that although the objects' classes as presented in LabVIEW differ (see Figures 4 and 5) from the OPC UA foundation's, the resulting server may be accessed, browsed, read and written with any client.

Before actually running a server, it has to be populated. We define the address space needed for the prototype in an *XML* file. Its structure is checked with a *schema* –a dedicated dictionary saved in an *XSD* file– to avoid any error while parsing it. In our case, the schema has to reflect the hierarchy and the attributes of the objects shown in Figure 5.

The OPC UA server is a major task of the main PLC. The required computing power is an additional criterion to evaluate the two cRIOs. The CPU charge of a cRIO9074 has been monitored versus several parameters (the clients' rate, the server's rate, the number of nodes or the number of clients) while running an OPC UA server. It has to be noted that for these tests, all nodes are refreshed at the server's publish rate.

A sample of the results is presented in Figure 6; with an estimate of 350 nodes and 4 running clients, it shows that a 9074 will have to handle a CPU charge between 35 and 40%. This number is rather high for only one running application, given that this PLC has to run other software such as the real-time part of the pointing

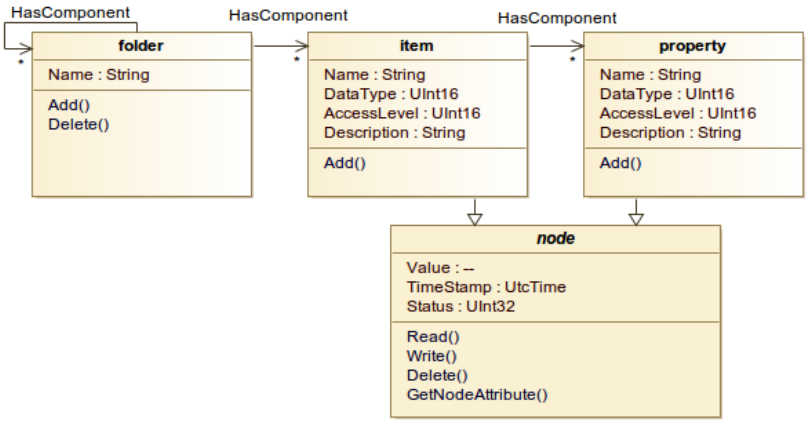

Figure 5. The OPC UA objects in the LabVIEW implementation. The hierarchy is defined with the *HasComponent* reference. Items and properties have common attributes as nodes.

and a margin of idle CPU time is always required. Considering that we can't use fewer clients – we may even require more of them–, to mitigate this value, it is possible to *a)* reduce the number of nodes for the prototype; *b)* reduce the rate – about 8% of charge less with a 2-second rate and 500 nodes; *c)* or use a more powerful CPU. In the latter case, the same tests performed with a cRIO9068 – featuring a 667 MHz dual core instead of a 400 MHz single core – should allow to run the PLC with plenty of margin.

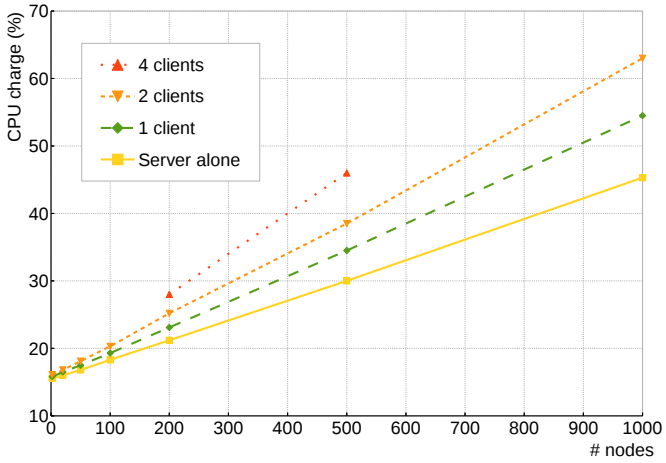

Figure 6. The CPU charge of a cRIO9074 while running an OPC UA server. The node values are published by the server and monitored by the clients every second.

4.2.2 The safety OPC UA server

Beckhoff provides with TwinCAT an embedded OPC UA server feature. The available services are *Data Access*, *Historical Access*, *Alarms and Conditioning* and *Security*.

This server is set up in the configuration manager of TwinCAT: amongst the automation variables managed by the safety PLC those chosen to be published are routed to the server.
TBDevelopped

5. THE ETHERCAT FIELD-BUS

5.1 EtherCAT with a CompactRIO

TBDevelopped

5.2 Safety over EtherCAT

TBDevelopped

6. CONCLUSIONS

The design of the TCS for SST-GATE has not come to his end yet. However, we can already report the first results of our experiments.

The try to manage real-time pointing with an FPGA is about to be converted. We have implemented the VHDL model of a CORDIC algorithm and have linked it to a real-time routine, all in a CompactRIO. The trigonometric functions being the main computation issues, making up the whole pointing algorithm is now just a matter of time.

The use of a CompactRIO as the core PLC of a telescope is looking good. We have checked separately the ability of this system to handle all the required functions of the TCS; to sum up: real-time telescope pointing, real-time access of the field devices, and remote management via an industrial protocol.

The decision between a NI9074 and a NI9068 is not made for the SST-GATE prototype: Achieving to manage locally the TCS with a NI9074 would be very cost effective; A NI9068 offers a more powerful computation ability, along with a great versatility with the possibility of a C-only software development.

ACKNOWLEDGMENTS

This work has been funded under the Convention 10022639 between the Région Île-de-France and the Observatoire de Paris. We gratefully acknowledge the Région Île-de-France, the CNRS (INSU and IN2P3), the CEA and the Observatoire de Paris for financial and technical support.
We gratefully acknowledge National Instruments France for technical support.

REFERENCES

[1] Volder, J. E., "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers* **EC-8**, 330–334 (1959).

[2] Lang, T. and Antelo, E., "CORDIC-based computation of ArcCos and ArcSin," in [*Application-Specific Systems, Architectures and Processors, 1997. Proceedings., IEEE International Conference on*], 132–143 (Jul 1997).

[3] "OPC Unified Architecture." <https://opcfoundation.org/about/opc-technologies/opc-ua/> (2014).