ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

EXPLORING ALTERNATIVES AND DESIGNING THE NEXT GENERATION OF REAL-TIME CONTROL SYSTEM FOR ASTRONOMICAL OBSERVATORIES

T. C. Shen*, J. Sepulveda, ALMA Obervatory, Santiago, Chile
P. Galeas, F. Huenupan, S. Carrasco, R. Augsburger, R. Seguel, U. de La Frontera, Temuco, Chile

Abstract

The ALMA Observatory was inaugurated in 2013; after the first eight years of successful operation, obsolescence emerged in different areas. One of the most critical areas is the control bus of the hardware devices located in antennas, based on a customized version of CAN bus. Initial studies were performed to explore alternatives, and one of the candidates can be a solution based on EtherCAT technology. This paper compares the current architecture with a new proposal compatible with the existing hardware devices, providing the foundation for new subsystems associated with ALMA 2030 initiatives. The progress of a proof of concept is reported, which explores the possibility of embedding the existing ALMA monitor and control data structure into EtherCAT frames, using EtherCAT as the primary communication protocol to monitor and control hardware devices of ALMA telescope subsystems.

INTRODUCTION

The ALMA Observatory was inaugurated in 2013; after the first eight years of successful operation, obsolescence emerged in different areas. One of the most critical areas is the control bus of the hardware devices located in antennas, based on a customized version of CAN bus. Initial studies were performed to explore alternatives, and one of the candidates can be a solution based on EtherCAT [1] technology. This paper compares the current architecture with a new proposal not only compatible with the existing hardware devices, but also provides the foundation for new subsystems associated with ALMA 2030 initiatives. This document reports the progress achieved in a proof of concept that explores the possibility to embed the ALMA monitor & control protocol into a EtherCAT protocol. The main goal of this phase is to obtain the technical assessment of the feasibility to implement the EtherCAT as the communication protocol to monitor and control hardware devices/controller in all the subsystems that comprises the ALMA telescope. This is a collaboration project between ALMA Observatory and the Universidad de La Frontera in the context of a QUIMAL fund (QUIMAL190009) [2], which is sponsored by Chilean National Agency for Research and Development (ANID). The main objective is to design, implement and evaluate an possible alternative of the existing antenna real time computer.

* tshen@alma.cl

THE ALMA DISTRIBUTED CONTROL SYSTEM

The ALMA control software is a distributed system that is divided into several subsystems, each focusing on different stages of the observation process. The subsystems provide software interfaces to transfer communication messages in a coordinated manner between them. Likewise, the control system is also responsible for coordinating, by means of events and/or command messages, all the activities involved in the different observation steps.

The main subsystems are Control, Correlator, Scheduling, Telescope Calibration, Executive and Archive. The software for each subsystem is implemented in one (or more) programming languages (C++, Java, Python) that support the ALMA common software (ACS) [3], CORBA-based framework. The official operating system is Red Hat Enterprise Linux Server release 7.6.

The Antenna Bus Master (ABM)

The ABM is a dedicated real-time computer to monitor and control the antenna hardware devices. The purpose of this computer is to process low level messages from all antenna devices, using a particular implementation of the CAN communication protocol [4]. The scheme of monitor and control conducted by the ABM computer is accomplished with adoption of the ALMA Monitor Bus (AMB) specification. It is a particular ALMA protocol, based on a CAN bus, to communicate with hardware elements, which defines a unique master connected with several slaves in the same bus. The AMB specification promotes a configuration that converts the transaction of CAN messages in a deterministic communication of the command control messages. The master, in a timely manner, is the only agent on the bus that can sends messages and wait responses of the other elements involved in the CAN bus. Similarly, the ABM make uses of five independent ALMA Monitor Bus channels to communicate with the devices spread out in the antenna. In every channel the ABM real-time computer acts as the CAN master and antenna hardware devices are the slaves on the CAN bus.

Hardware Device Interface (AMBSI)

The ALMA Monitor and Control Bus Interface (AMBSI) is a standard interface that defines, on the one side the physical connection of nodes on the bus, and on the other side, the application level protocol that nodes must conform to be monitored and controlled by a software control system. The AMBSI specification outlines that each ALMA device has

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

a node address (0-2030), a unique 64 bit serial number, the ALMA M&C bus (AMB) is a master/slave multi-drop serial bus. The ALMA specification is implemented in two standard interfaces, known as the ALMA Monitor and Control Standard Interface 1 & 2 (AMBSI1, AMBSI2). In addition, the standard also established that the low level protocol is Controller Area Network (CAN) serial bus operated in a master/slave mode by a dedicated bus master. The remote frame transmission request (RTR) is not used by the bus master to gather monitor data.

Timing Event

The ALMA software infrastructure synchronizes the activities with the antennas by a common time base [5]. The time base definition is based on an electronic pulse signal with a period of 48 milliseconds, denominated as Time Event (TE) and absolute Gregorian time associated with each TE. The time system is provided by group of specialized hardware elements and software component denominated the Array Time. It is the main reference time for all machines connected in the ALMA system environment. This is a special coordination mechanism of the time that enables interactions between participants, in a time synchronized manner, according to a unique universal time provided by this dedicated time scheme. The coordination between different participants is performed by means of a time events.

The activities of monitor/control of hardware devices are completed in synchronization with the Time Events (TE). The Fig. 1 below shows a TE-related control/monitor command, the processing of this type control command may begin at the starts of every TE pulse. Any slave (hardware device) that receives a control command from the CAN master (ABM) must processes it in the first 24 ms of TE. In the same way, monitor commands are processed in a second window of 20 milliseconds of the 40 ms TE-window.

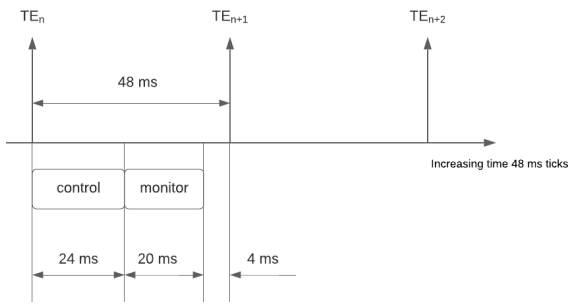

Figure 1: Time Event linked with control and monitor commands.

THE PROPOSED DESIGN

The design take into account different parameters of open standards in control bus. The Table 1 shows the attributes of EtherCAT, CAN-Bus (CANOpen), RS-232, RS485 bus standards.

Different scenarios were analyzed for the observatory in the coming years, and these are the most relevant requirements and uses cases in the new control system:

- monitor and control of the existing hardware devices of the ALMA telescope.
- monitor and control of new hardware devices to be introduced mainly due to obsolescence problems.
- monitor and control of new hardware devices to be introduced by improvement and automation in the antenna remote recovery activities.
- monitor and control of new hardware devices of the ALMA 2030 developments.

Given the aforementioned scenarios, it was decided to explore a solution that could combine the EtherCAT, CANOpen and OPC UA protocols. The new design must be capable to combine, in the distributed control system, the different flavors of protocols and interfaces of existing design and also guarantee the future extension of the this new design. The resulting architecture must minimize the impact in the higher control software layer based on ACS. Additionally, the new design should provide a way to allow hardware engineers to be able to access the control bus for maintenance purposes, without the need to have the entire stack of control software up and running. This is an important aspect, because it prevent hardware and software engineers to work in parallel in different areas during the scarcely available technical time.

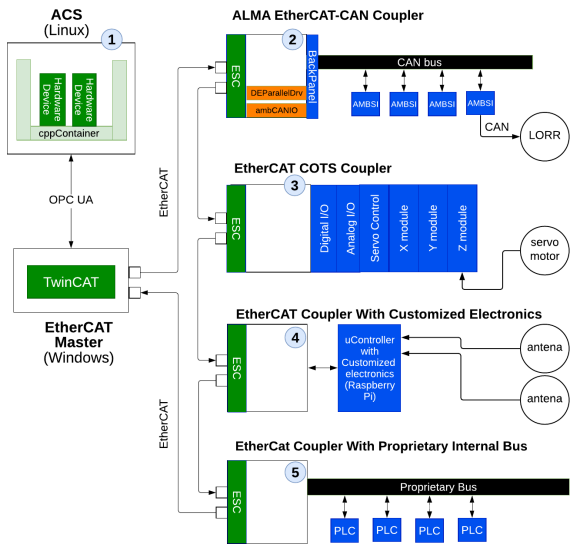

Figure 2: Control bus use cases in the coming years in ALMA.

The scenario or use cases displayed in the Fig. 2 are the following:

ALMA Device Controller

The use case 1: The existing ABM's functionalities are separated into two parts. The first part represents the user-space (no real-time) to be executed in a Linux server with ACS. The existing "Hardware Device" components are adapted to use a OPC UA client driver, which sends (via Ethernet) messages to the second part (real-time) of the proposed system. Finally, an application must be developed to run within the IPC on top of TwinCAT 3, in order to verify

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

Table 1: Control Bus Comparison

	EtherCAT	CAN-Bus	RS232	RS485
Speed/Cable Length	100 Mbps at 100 m between two nodes for full speed	1 Mbps at 40 m from the start to end nodes	19,200 bps at 15 m from the start to end nodes	1 Mbps at 120 m from the start to end nodes
Cable Topology	Daisy Chain, Point-to-Point, Combination of the above	Daisy Chain	Point-to-Point	Daisy Chain, Point-to-Point
Max Number of Nodes	65535	128	1	32
Open Source Software	Yes	Most of CANOpen is proprietary	No predetermined protocol	No predetermined protocol
Required Hardware at master	Regular Ethernet adapter	PCI, USB, IC-type interface	PCI, USB, IC-type interface	PCI, USB, IC-type interface

the reception of the OPC UA messages and dispatch the message to the EtherCAT bus.

ALMA Current CAN Bus Device

The use case 2: This is the most important scenario for ALMA, because it tackles the current obsolescence problem of ABM real-time computers. In this scenario, the ACS components deployed in containers of the Linux Server running ACS will be able to communicate via EtherCAT with hardware devices connected to the CAN bus. The commands sent by hardware device component are processed by the OPC UA server and translated into EtherCAT frames, once the message is received by an EtherCAT-CAN coupler, it will converts the messages into the CAN messages compatible with the current implementation of ALMA control bus.

COTS EtherCAT Coupler with Digital, Analog I/O

The use case 3: This differentiates from use case 2 because it includes functionalities to interact with hardware module which are connected to the E-bus of the EtherCAT master or another EtherCAT coupler (i.e: Beckhoff ELK1100), for example, relays, temperature sensors, and servo motor driver, and encoder modules. This use case is the native usage of EtherCAT with hardware of the Beckhoff vendor.

EtherCAT Coupler with Customized Electronic

The use case 4: this scenario incorporates a development of EtherCAT slave that allow to attach customized functionalities implemented in a micro-controller.

EtherCAT Coupler Interfacing with a Sophisticate Subsystem

Use case 5: This is a generic case, in principle, it would be used to adapt future instruments with proprietary protocols to the EtherCAT infrastructure.

IMPLEMENTATION

It was decided to focus our effort and resource to demonstrate the use case 0 and use case 2, because their are the most complicated and more likely scenario for the coming years. The use case 3 poses very low risks and is well demonstrated already in the industry. The use case 4 and use case 5 are variants of the use case 2.

The EtherCAT-CAN coupler will be implemented on top of a XMC4800 micro-controller [6], mainly because the XMC4800 has a MultiCAN controller that could support up to 5 independent CAN buses. Using the SDK of the XMC platform, the ALMA customized CAN protocol is implemented. The choice of the XMC4800 was heavily influenced by past experience of the team, but any micro-controller, FPGA with CAN transceivers could also be a possible alternative.

This design will cover all the different possible scenarios that ALMA may face in the next decade. It allows a coexistence of the different field bus protocols and a smooth transition of the existing architecture to a one that could also support ALMA2030 projects.

As shown in Fig. 3, the testing setup comprises the following equipment: i) a VM with ACS and TwinCAT development environment, ii) a Beckhoff CX2030 IPC EtherCAT master, iii) an Infineon XMC4800 microcontroler EtherCAT slave, iv) an ALMA AMBSI board representing hardware device of ALMA.

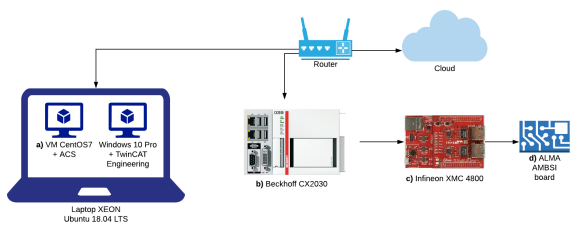

Figure 3: Physical connection of the test bentch.

The objective is to reproduce the existing communication between of an ABM and a hardware devices through the AMBSI interface, but instead sending the customized CAN frame, the data structure will be embedded into a EtherCAT Service Data Object (SDO) frame.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

The SDO protocol in EtherCAT allows to configure (write) and read back values of the Object Dictionary (OD) [7]. It establishes a client-server communication between the EtherCAT master (client) and a EtherCAT slave (server). The server (the XMC-4800 microcontroller) exposes an OD that defines the data structure and the related addresses for the SDO communication. If the amount of the entries of the OD requested by the client exceeds the maximum size, The SDO protocol will segment the data structures in multiples SDO frames. The simplified version of the OD was shown in Fig. 4.

5001:0	ALMA_CAN_Device1	RO	> 6 <
5001:01	AMBSI_SERIAL_NUMBER	RO	0x10e14f7e020800e7
5001:02	AMBIENT_TEMPERATURE	RO	0x10100038 (269484088)
5001:03	PROTOCOL_REV_LEVEL	RO	0x00000000 (0)
5001:04	CAN_ERROR	RO	0x000A0000 (655360)
5001:05	TRANS_NUM	RO	0x0000017F (383)
5001:06	SW_REV_LEVEL	RO	0x00020000 (131072)

Figure 4: Object dictionary.

In order to implement the use case 0 and use case 2, the data flow described in Fig. 5 was implemented, the main components are:

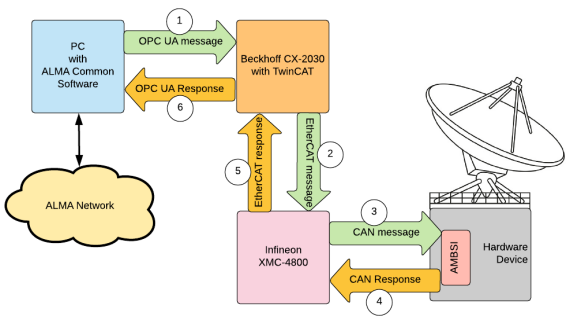

Figure 5: Data flow of the control bus.

OPC UA Device Driver

The existing ALMA software uses Hardware Device Driver approach, in which according to the type of the hardware controller, the corresponding device driver must be developed. From the time being, the most common drivers are the AMBDevice driver and Ethernet Device Driver. For the OPC UA Device driver, the Ethernet Device Driver was extended, and additional OPC UA functionalities were added. As described in Fig. 6 This driver in essentially a OPC UA client that reads/writes data structure defined in OPC nodes hosted by an OPC server. For each entry of the OD, there is a specific node in the OPC server. Within this node, a sub-node is defined for receiving requests from the client (AMBMessage), additional the client subscribes itself to a second sub-node in order to receive the eventual response (AMBResponse).

TwinCAT Application

A TwinCAT application was developed. This application is the EtherCAT master that runs inside of the Beckhoff CX2030. As shown in Fig. 7, the main objective of this application is to serves as a gateway between the OPC

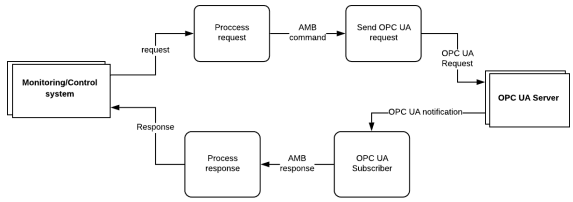

Figure 6: Data flow: ACS to OPC UA section.

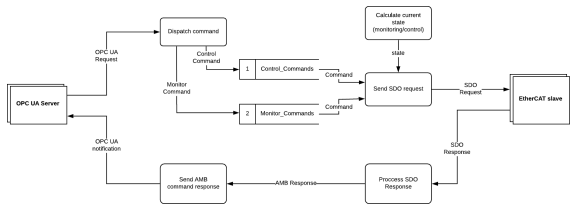

Figure 7: Data flow: TwinCAT to XMC section.

UA interface (external) and the EtherCAT bus (internal). This application implements the OPC UA server with a tree of nodes that represents the OD. Upon an reception of a OPC UA client request, the server queue the corresponding AMBMessage data structure in the control queue or monitor queue accordingly. Two messages processors (see Fig. 7) convert the AMBMessage into SDO data structure and dispatch this message in the EtherCAT bus. In addition, a state machine was implemented in order to separate the dispatch of messages in the queue according to the definition of the TE (see Fig. 1): in each windows of 48 ms, the first 24 ms are reserved for the control requests, and the next 20 ms are reserved for the monitor requests, then 4 ms of idle period follows. In each AMBMessage data structure, the address field maps the sub-index in the OD in a SDO frame. Once the EtherCAT slave (XMC4800) responds the SDO request, the data is extracted and filled inside of an AMBResponse structure which is updated in the corresponding OPC node in the tree of the OPC Server. The node trigger a notification to the OPC client (previously registered to this node) and the data is ready to be retrieved by the OPC client embedded in the OPC UA Device driver.

EtherCAT Slave (XMC4800)

Inside of the micro-controller XMC4800, as shown in Fig. 8, an firmware is implemented to forward each SDO [8] request to the AMBSI card. Upon arrival of an SDO request, the firmware extract the relevant parameters and create an ALMA CAN frame and dispatch it in the CAN bus synchronously, once the respond of the AMBSI card returns, the XMC updates the content in its OD defined its EtherCAT Slace Controller (ESC), and the next EtherCAT frame will take care to bring it back to the EtherCAT master.

RESULTS

The aforementioned implementation allow to generate an end-to-end communication between the logic layer inside of the ACS component and the hardware device located behind the AMBSI board by using COTS hardware and software.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

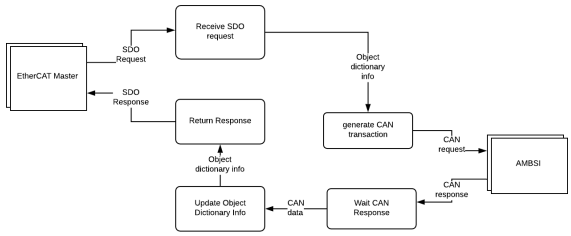

Figure 8: Data flow: XMC to AMBSI section.

The performance tests consist on sending 10000 messages of 6 possible monitor points (see Fig. 4).

As shown in Fig. 9, we built a histogram with the time-of-fly of EtherCAT messages associated with six monitoring points (sub-index 01 to 06) of a hardware device in an antenna. We used the Wireshark software to analyze the TCP packets traffic between the CX2030 and the XMC4800. The packets were processes on a Jupyter notebook to calculate the time-of-fly of each EtherCAT packet. Given the limitations of the sampling time, it was not possible to obtain the EtherCAT propagation times. However, we estimated that it is in the order of nanoseconds, which is negligible for our measurement as the most time consuming in the communication process is related to the CAN communication between the XMC4800 and the AMBSI card. The mean values and the standard deviation of the times of each monitoring point were also calculated and shown in the Fig. 9. The monitor points defined in the OD at the sub-index 1,3, and 6 took 200 µs in average, while the sub-index 2, 4, and 5 took 400 µs in average. The differences between these two groups are because the implementation done inside of the XMC.

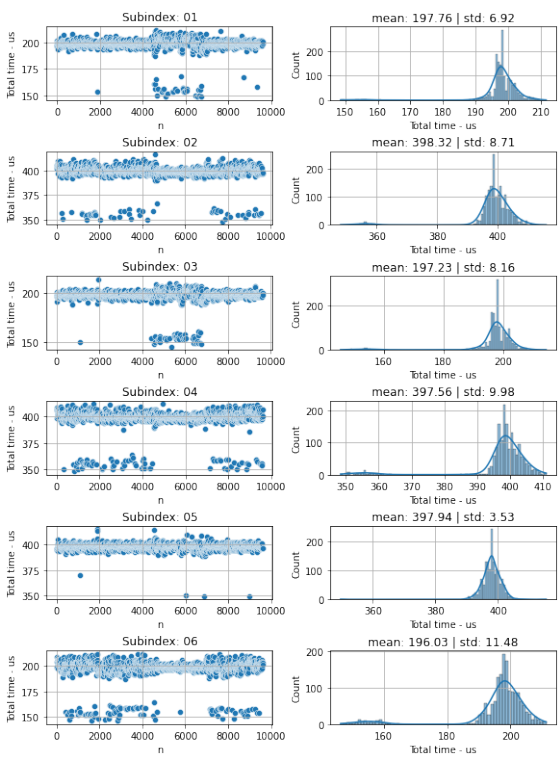

Figure 9: SDO traffic performance.

CONCLUSION AND FUTURE WORKS

The proposed design demonstrated to be an viable alternative to replace the existing control bus in ALMA. Based on a open standard such as EtherCAT, it offers a ready to use framework to design the control interface of new hardware of the ALMA2030 projects, these hardware will behave as EtherCAT slaves. During the transition period, it was expected to have a combination of existing CAN based hardware device and EtherCAT based device in operation. This scenario can be supported by the design proposed in this paper. The feature of having an OPC UA interface between the Hardware Device Driver and the EherCAT master (replacement of ABM) allows more efficient usage of the technical time between software engineers and hardware engineers in the observatory. The hardware engineers don't need to use the entire ALMA software for their specific hardware troubleshooting activities, which currently block the software testing activities of the software engineers in the observatory. Finally, the ALMA ICD definition fits perfectly in the concept of the OD dictated by the EtherCAT CoE, therefore minimal impact is expected in the Hardware Device layer.

Nevertheless, the SDO structure suits to the purposes of Monitoring and Control in the ALMA Control bus, we believe more performance can be achieved by using the PDO structure. The SDO can be reserved for setup commands, and low data rate monitoring points, such as temperature sensors. But the PDO structure can be reserved for high data rate monitoring points, such as, vibration sensors, encoders values, etc. The concept of using PDO is under analysis. Additionally, the TE infrastructure also needs to be improved in this design as same as the possibility to use the MultiCAN infrastructure of the XMC4800. Currently, only one of the possible six channels are being exploited.

REFERENCES

[1] EtherCAT Technolog Group, https://www.ethercat.org/default.htm

[2] Proyecto QUIMAL190009, https://cemcc.ufro.cl/quimal/

[3] G. Chiozzi, B. Jeram, H. Sommer, *et al.*, "The ALMA common software: a developer-friendly CORBA-based framework", in *Proc. SPIE*, vol. 5496, pp. 205–218, 2004. doi:10.1117/12.551943

[4] "CAN Bus Specification 2.0", Robert Bosch GmbH, 1991.

[5] R. Amestica, B. Gustafsson, and R. Marson, "Time synchronization within the ALMA software infrastructure", in *Proc. SPIE*, vol. 6274, pp. 338–346, 2006. doi:10.1117/12.670298

[6] XMC4800 32 bit Arm Cortex Microcontroller, https://www.infineon.com

[7] "EtherCAT Modular Device Profile", ETG.5001.6220 S (D) V1.0.5. https://www.ethercat.org/download/documents/ETG5001_6220_V1i0i5_S_R_IO-LinkMaster.pdf

[8] "EtherCAT Master Classes", ETG.1500 D (R) 1.0.2. https://www.ethercat.org/download/documents/ETG1500_V1i0i2_D_R_MasterClasses.pdf