ALMA-US Draft Computing Memo

ALMA Monitor and Control Bus

Methods of Ensuring Isolation Among Devices

Sharing One ALMA Monitor and Control Bus

Mick Brooks
2000-10-20

Revision History

Date	Revision	Description	Contributors
2000-08-03	1.0	First public release	Mick Brooks
2000-10-23	1.1	Revised due to comments from D'Addario and Tan	Mick Brooks

Table of Contents:

1. Introduction

This document briefly outlines a proposal for managing the development of interfaces to the ALMA Monitor and Control Bus (AMB). It is in response to some requirements defined by the Systems Group, and is an attempt to provoke discussion on several gray areas, such as the maintenance responsibility for embedded code.

2. Requirements

The ALMA Systems Group has charged the Computing Group with explaining how it intends to make the M&C bus behavior “bulletproof”. More specifically, this is taken to mean that the action of a single device on the bus must be such that it is unable to hang other devices or prevent them from accessing the bus.

In addition to this requirement, there is a strong suggestion that the interface hardware be standardized.

It should be noted that this proposal and ensuing discussion should not delay or jeopardize development work which is already in progress for the Test Interferometer.

3. Bus Master Nodes

It is the responsibility of the ALMA Computing Group to design, implement and test all code developed for bus master platforms. The method of ensuring correct bus operation for the master(s) is thorough testing and tight control of code and coding methodology [Software Process, Methodology and Tools, Draft ALMA Computing Memo, G. Chiozzi, R. Karban, P. Sivera, Issue 1.0, 2000-07-28].

It is required that on any single AMB there be only a single bus master node. Bus master nodes have been coded and tested for the following hardware and software combinations:

- O/S: VxWorks. SBC: MVME2700, MVME1603, MVME2604. CAN Interface: Tews Datentechnik TP816 Dual and Single Port PMC CAN modules (available in the US from SBS Greenspring). The ABM and ARTM will be MVME2700 SBCs.
- O/S: Windows NT. Any PC platform with PCI. CAN Interface: National Instruments Dual Port PCI CAN board. These are intended for use in laboratory situations and not in the operational array.

4. Bus Slave Nodes

All slave node implementations must comply with the bus specification detailed in ALMA Computing Memo #7, which requires that no slave node initiate transactions on the bus unless polled by a bus master. It is intended that there be two prime slave node interface implementations, the AMB Standard Interface (AMBSI) and the Zero-Additional Software Interface (ZASI). All designers of hardware interfacing to the AMB should use one of these two; new slave node implementations may be considered where sufficient justification can be provided.

4.1 AMBSI

The purpose of the AMBSI is to provide a highly flexible interface board with on-board CAN and device-side I/O consisting of parallel, serial and bit-wise ports. The board would be delivered to hardware designers with a standard firmware package supporting basic CAN access to the I/O ports. The micro-controller on the AMBSI will have sufficient spare processing capacity to run additional device-specific code, such as closing fast control loops.

Where device-specific code is required, the ALMA Computing Group may develop the code under the direction of the hardware developer or the hardware developer may develop the code in conjunction with the Computing Group. In both cases, the device-specific code is subject to the coding rules described below, and the ALMA Computing Group will be responsible for integration testing the product to ensure that the device-specific code does not interfere with the CAN slave code. Such testing is outlined in Section 5. All device-specific code will be archived by the ALMA Computing Group.

All AMBSI boards should be similar in hardware, and identical in size and connector arrangement. Code may be loaded into on-board flash memory by means of the CAN bus or the local RS232 port.

The CAN bus servicing code is written so as to be entirely interrupt-driven. It runs only in response to interrupts from the CAN controller, and those interrupts are set to the highest possible priority (in the Infineon C167, this is interrupt level 15, group level 3). When not servicing an interrupt, the processor executes an idle loop. Device-specific code may include a function to execute from the main idle loop, or it may simply consist of the callbacks for CAN message reception interrupts. Callback functions are used by the CAN interrupt service routine to dispatch device specific functions in response to received monitor or control requests.

Device-specific code must follow these rules:

- Device-specific code must not modify the interrupt table entry for the CAN controller (interrupt vector XP0INT) or the CAN interrupt priority registers. This means that interrupt vector table entry 0x40 is reserved for the CAN interrupt and the XP0IC register may only be altered by the CAN protocol code.
- Any interrupt handlers defined in device-specific code must use interrupt priorities at an interrupt level lower than 15, the level of the CAN interrupt. It is not sufficient to use an interrupt level of 15 and group level lower than 3. All other interrupt control registers must not set to any group level of interrupt level 15 which is reserved for CAN interface code use.
- Any interrupt handlers defined in device-specific code must be written with the knowledge that the CAN interrupt is of higher priority and may pre-empt the handler at any time.
- Device-specific code must not modify any registers associated with the CAN controller, nor any other registers reserved for use of the CAN servicing code. These registers are Special Function Registers at addresses 0xFEF00 through 0xFEFFF.
- Device-specific code may use any timer except the Timer 2 unit of the on-board General Purpose Timer 1. Timer unit 2 is used to control the interface to the Dallas Semiconductor 1-Wire interface.
- As the CAN interrupt service routine has the highest priority, it may preempt or delay other interrupt service routines when activated. The maximum preemption time of the CAN ISR is 150 μs; this is also the maximum latency for lower priority interrupts.

4.2 ZASI

This device would perform the function of a CAN bus to Serial Peripheral Interface (SPI) converter. The ZASI is designed for use in systems which either have their own microprocessors or need a very minimum amount of I/O to the CAN bus. It is currently proposed that the resulting subsystem would require +5 Volts at less than 2- ma and occupy a small daughter PCB of approximately 2 in by 1 in. The PCB would be either mounted to the main PB Board by several SIP headers which also serve as the I/O connectors for the subsystem.

The board would have a unique serial number, which would be programmed into the microprocessor or extracted from a Dallas Semiconductor silicon serial number device. There would be about 16 bytes of RAM which could be accessed by either the CAN bus or the SPI port. An ATN output would indicate when RAM contents had been altered by the CAN bus. The unit will have its own oscillator and reset circuitry, completely independent of the host microprocessor. All of the software in the subsystem would be developed by and be the responsibility of the software group.

In addition to providing the CAN interface, the unit would also provide the interface for the 48 ms TIMING pulses which come in through the AMB DB-9 connector or module wiring as RS-485 and are made available as TTL signals to the application circuitry. The AMB Global Reset signal would cause a reset of the ZASI circuitry only.

4.3 Other Slave Interfaces

In general, hardware designers are prohibited from developing alternative AMB slave interfaces to the AMBSI and ZASI. Two exceptions are noted here:

- The antenna vendor ACU implementations are not constrained by ALMA. Such interfaces will be extensively tested in the factory, during servo tests and during antenna acceptance at the site.
- An ACU simulator based on a Linux PC and using a Janz CAN-PCI/K2O board will be developed by the ALMA Computing Group. This implementation will not be used outside of lab test situations.

5. Testing

It is assumed that a control system test-bed similar to that operated by ESO in Garching will be used in the development of the ALMA project. Such a test bed could be located in Tucson, Socorro or Garching and possibly all three. One element of the test bed would be a complete AMB with master node and representative slave nodes.

It will be a requirement for all subsystems connected to the ALMA M&C system to be thoroughly tested on a test-bed in a fashion incorporating regression and integration tests. It is expected that slave subsystems incorporating the ZASI will require less testing of the physical interface than subsystems with device-specific code running in the AMBSI. Nevertheless, subsystems using either style of slave interface should be regression tested whenever a new software revision is released.

Such tests would involve but not be limited to the following:

- 10^9 transactions per test per subsystem
- All messages defined for slaves of all software revisions
- Saturated bus load
- Unloaded bus

Where device-specific code has been included in an AMBSI, the source code will also be inspected by the ALMA Computing Group for compliance with the coding rules described in Section 4.1. In addition, semantic and other code quality test software such as lint may also be employed.