Nuclear Instruments and Methods in Physics Research A 815 (2016) 83–90

Contents lists available at ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

Testing flight software on the ground: Introducing the hardware-in-the-loop simulation method to the Alpha Magnetic Spectrometer on the International Space Station

Wenhao Sun [a,*], Xudong Cai [b], Qiao Meng [a]

[a] Southeast University, Nanjing 210096, China
[b] Massachusetts Institute of Technology, MA 02139-4307, USA

ARTICLE INFO

Article history:
Received 15 January 2016
Accepted 15 January 2016
Available online 21 January 2016

Keywords:
Alpha Magnetic Spectrometer (AMS)
Hardware-in-the-Loop Simulation
Embedded Software Test

ABSTRACT

Complex automatic protection functions are being added to the onboard software of the Alpha Magnetic Spectrometer. A hardware-in-the-loop simulation method has been introduced to overcome the difficulties of ground testing that are brought by hardware and environmental limitations. We invented a time-saving approach by reusing the flight data as the data source of the simulation system instead of mathematical models. This is easy to implement and it works efficiently. This paper presents the system framework, implementation details and some application examples.

1. Introduction

The *Alpha Magnetic Spectrometer* (*AMS*) is a particle physics experiment mounted on the *International Space Station* (*ISS*) [1,2]. It was designed to measure antimatter in cosmic rays and search for evidence of dark matter.

During the continuous operation of AMS, scientists of the AMS collaboration monitor the instrument status 24/7. Real-time monitoring allows issue resolution, such as communication anomalies or hardware failures, as soon as possible.

However, ground monitoring is not 100% reliable due to communication outages. To prevent potential sudden damage, more and more automatic protection mechanisms are being added into the onboard software as operational experience grows.

Automatic protection functions define some conditions, and once the conditions are triggered, the control system would take corresponding actions.

Consider the following scenario: a function is designed to monitor a group of temperature sensors, and it is supposed to do something if temperature alarms are detected on certain sensors. If the trigger logic is simple, e.g. only 1 sensor is taken into account, the functions would be easy to implement; otherwise, e.g. a configurable set of sensors are involved, the implementation would not be that obvious and the validity needs to be tested rigorously.

To perform such a test, we need to generate the triggering conditions that might be some sensors getting abnormal temperatures, but we have to face the fact that not all devices are installed in the ground test system. Furthermore, it is not easy to raise or lower the temperature whenever we want.

In order to overcome the difficulties brought by hardware and environmental limitations, we introduced the hardware-in-the-loop (HIL) method. *Hardware-in-the-loop simulation* is a technique that is widely used in the development and test of real-time control systems [3]. It connects the real controller in the simulation loop and virtual devices which interact with the controller.

In this research, we applied a time-saving approach and built the HIL system for AMS flight software test: based on the knowledge of AMS operations (Section 2), we built a general behavior model for the virtual devices and invented a language to describe different devices (Section 3); instead of building mathematical models for the simulated environment, we set up a data flow to feed the simulation system with the real-time flight data of AMS and made it possible to reproduce space environment on the ground (Sections 3 and 4); customized data derived from the flight data generates various conditions as expected (Sections 3 and 4); we compared our approach with typical HIL schemes, and summarized how to take advantage of our idea and build HIL systems to assist maintenance of complex systems (Section 5).

2. AMS operations

AMS consists of a series of detectors that are used to determine various characteristics of the radiation and particles as they pass

* Corresponding author. Tel.: +86 159 5051 3326.
E-mail address: wenhao_sun@126.com (W. Sun).

http://dx.doi.org/10.1016/j.nima.2016.01.042
0168-9002/

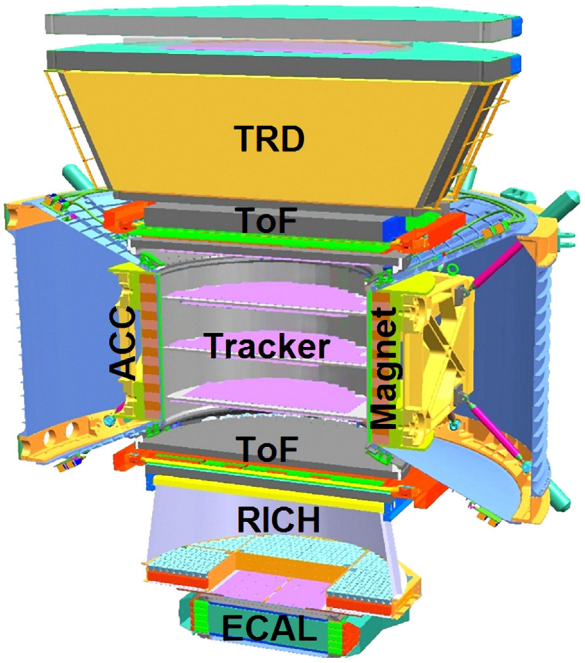

Fig. 1. Structure of AMS.

through. Fig. 1 shows the main components: Transition Radiation Detector (TRD), Time of Flight counter (ToF), silicon tracker, permanent magnet, Anti-Coincidence Counter (ACC), Ring Imaging CHerenkov detector (RICH), and Electromagnetic CALorimeter (ECAL).

Signals from these detectors are processed by the electronics. Data is transmitted under a set of communication protocols. All processes are managed by real-time software onboard the controllers.

Manual control from the ground is possible only when we have good communications with the ISS. Hence communication outages pose potential risks.

A solution is to add automatic procedures in the onboard software, where the triggering conditions are undoubtedly essential and should be verified and validated.

2.1. AMS electronics

The AMS electronics has 650 boards and about 300,000 data channels. This large amount of electronics could be divided into 2 major parts, the *data acquisition* (*DAQ*) *system* and the *slow control system*, besides which the *main data computer* (*MDC*) deals with all commands and data on the top layer. Fig. 2 shows the basic topology of the system [4–6].

Node is an abstract concept here. In general, a node refers to a processor unit and the devices under its control. The processor could be a digital signal processor (DSP) in the DAQ system or a microcontroller unit (MCU) in the slow control system.

2.2. Communication protocols and data formats

Communications inside AMS follows the master–slave principle: high-layer nodes are masters and low-layer nodes are slaves accordingly. MDC is the master of the entire AMS electronics, in other words, all the other nodes are slaves of it.

Two main communication protocols are applied: AMSWire protocol (customized on the basis of the IEEE 1355 SpaceWire Standard) in the DAQ system and the Controller Area Network (CAN) protocol in the slow control system [7]. For each protocol, an encapsulation format, namely AMSWire packet and CAN

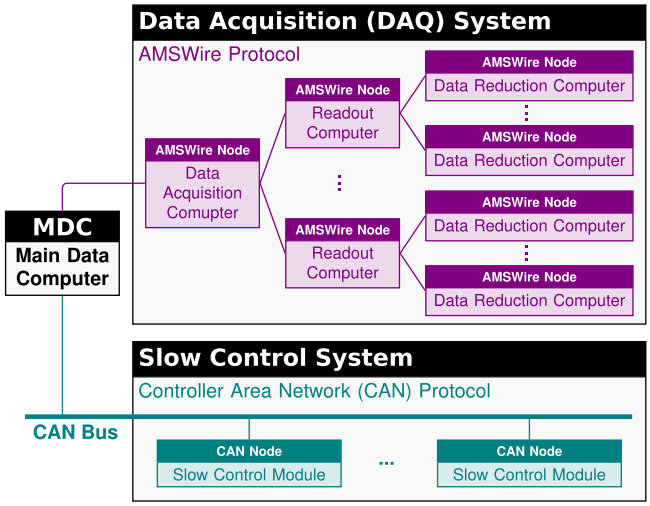

Fig. 2. Topology of AMS electronics.

packet, was designed to carry *AMSBlocks*, the standard format for AMS data processing.

Since AMS is a payload of the ISS, protocols and data formats used by the ISS are involved in the AMS data flow as well.

2.3. Flight operations

The MDC software automatically manages almost every aspect of regular operations: data acquisition, storage management, communications, monitoring, etc. If some new features are required for automatic control, the MDC software must be updated.

As slaves of the MDC, nodes in DAQ and slow control systems work in “slave” mode: get instructions, do something, and send replies. These nodes send control signals and collect data if and only if they get explicit instructions from their masters.

2.4. Ground operations and potential risks

MDC is unable to decide the execution time for every possible operation, while ground commands tell it what to do occasionally. However, not every command arrives in time.

On one hand, ground commands would be sent out only when the communication link works well; on the other hand, delay of telemetry data postpones the reactions more or less.

The inevitable delay of ground commands brings potential risks, especially when AMS is facing environmental changes in space. Consequently, we try to migrate more activities to MDC and reduce risks.

2.5. Software updates and ground tests

The AMS collaboration has accumulated a wealth of operational experience since the installation of AMS took place in May 2011. From time to time, people expect more features of the MDC software which would reduce the complexity of ground operations.

When a new version is ready to go, the software needs to be tested before we upload it to AMS. Real-time control software should be tested on certain hardware, and the *flight simulator*, a spare of the AMS electronics, provides the facilities.

The flight simulator is available in the lab. It is equipped with some nodes and sensors, but not all. On the flight simulator, we can update MDC software and test newly added functions.

The hardware and environmental limitations, however, make some tests difficult: if a function relies on either return values from a component that is missing on the flight simulator, or