CANBUS AND MICROCONTROLLER USE IN THE BABAR DETECTOR AT SLAC

H. B. Crawley, P.-A. Fischer, R. L. McKay, and W. T. Meyer, Department of Physics and Astronomy, Iowa State University, Ames, Iowa, 50011, and P. L. Anthony, Stanford Linear Accelerator Center, Stanford, California 94309

Abstract

The BaBar collaboration has chosen the Controller Area Network (CAN) [1] bus for its controls and monitoring field bus. In addition, the Motorola MC68HC705X32 [2] microcontroller, which has a CAN interface, was chosen for the standard intelligent device for monitoring boards. This paper describes the CAN system used by BaBar and the embedded software that supports it, using the General Monitoring Board (GMB) as a specific example. The GMB is a CAN module that digitizes 32 differential analog signals and has eight bits of bi-directional I/O.

1. INTRODUCTION

The BaBar detector at the Stanford Linear Accelerator Center's PEP-II collider is designed to study CP violation by accumulating a large number of pairs of neutral mesons containing a b quark. The key to obtaining a large number of these events is efficient operation for a long time, and a critical element in this "factory mode" of operation is reliable monitoring of the entire detector. Early detection of out-of-range conditions can prevent costly down time due to hardware failures.

The heart of the monitoring system is the Experimental Physics and Industrial Control System (EPICS) [3]. This suite of tools uses a VME single board computer both to interface to the monitoring hardware and to serve the data to client processes running on a TCP/IP network. Clients can be such processes as operator displays, data archivers, and alarm handlers.

The BaBar field bus between the sensors and the control system is CAN. This bus follows an open standard that was developed for use in the automobile industry and enjoys widespread use, making parts easily available and inexpensive. Reliability was a primary consideration in its design. EPICS-supported VME hardware exists in the form of an industry pack (IP) carrier board from Greenspring Corporation [4] and a CAN driver IP module from TEWS [5].

Many microcontrollers with a built-in CAN interface exist and it is also possible to use an external CAN interface with most other microcontrollers. The BaBar collaboration chose to standardize on the Motorola MC68HC705X32 microcontroller, which has an embedded CAN interface.

In terms of the OSI reference model for communications, CAN defines the lowest two layers, the physical and datalink layers. EPICS provides the higher layers, although the middle layers are very thin; EPICS appears to CANbus mostly as an application layer. Figure 1 illustrates this structure.

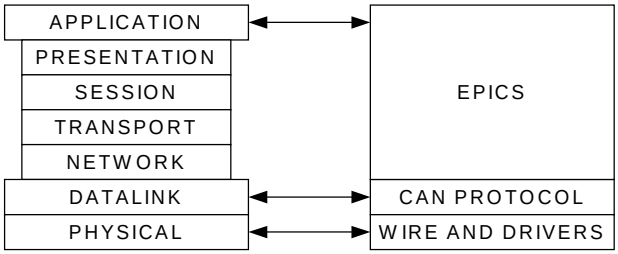

Figure 1: The OSI Reference Model for CAN

BaBar monitors over 50,000 quantities, distributed among six detector subsystems and a general infrastructure system. Monitored quantities include temperatures, radiation levels, power supply parameters, humidities, gas systems, and front-end electronics.

2. BABAR CANBUS STANDARDS

2.1 CANbus description

CANbus has international recognition as standards ISO 11898:1993 and ISO 11519-1:1994. It is a daisy-chained bus in which two wires carry the signal information. Other wires in the bus carry a ground reference, an optional +5V line for driving interface circuitry, and a current return for the +5 V line. While other physical implementations are possible, BaBar uses cables of twisted pairs of copper wire.

The two signal lines are labeled CAN_H and CAN_L. In the quiescent mode, both lines are at the midpoint of their range, about 2.5 V. During data transmission these signals go above or below their dormant state by at least 700 mV. When active, the bus is in either a dominant state or a recessive state. The dominant state has CAN_H high and CAN_L low. If multiple modules on the same

This work was supported in part by the U.S. Department of Energy, Division of High Energy Physics, under contracts No. DE-FG02-94ER40817 and DE-AC03-76SF00515.

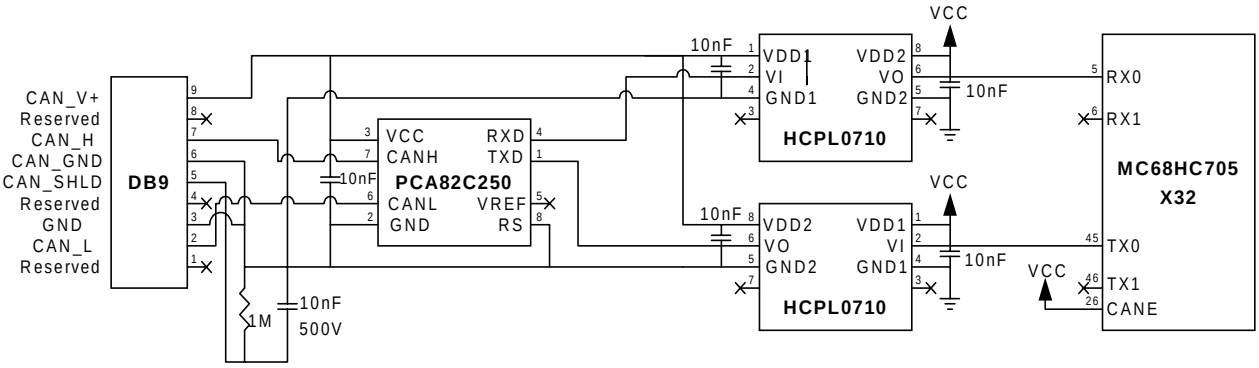

Figure 2: The Input Network

bus are trying simultaneously to assert dominant and recessive states, the bus will be in the dominant state.The dominant state represents the value "0" and the recessive state the value "1". Because CAN_H and CAN_L independently carry the information, the bus can function if one of the signal lines is broken, albeit with lower noise rejection.

In general, CAN modules do not need identifiers. Instead, each message carries a message identifier, which also sets the priority, and modules look for messages that they have been told to accept. Thus multiple modules may read a message from the bus and the sending module does not know or care how many modules are reading it. A lower message ID takes priority over a higher one. The CAN standard supports both 11-bit and 29-bit message IDs; BaBar has chosen to use 11 bits.

If a module wishes to use the bus, it begins by sending the message ID while simultaneously monitoring the bus. If at any point, it asserts a recessive state and sees the bus in a dominant state it knows that another module is trying to send a higher priority message and it aborts the transmission. Because the bus is always in the correct state for the highest priority message there is no inefficiency in bus arbitration. At the end of the message, modules that have pending transmissions try again.

Error detection using a 15-bit cyclic redundancy check is built into the CAN hardware, as is a standard error response protocol.

In general, CANbus modules are all equal; there is no intrinsic master/slave relationship.

2.2 BaBar CANbus Usage

BaBar has chosen to restrict the generality of the CAN standard in two ways. First, because we use EPICS and the VME CANbus interface to collect the data, we have what is essentially a master/slave structure. Second, we assign station numbers to each module on the bus and embed it into the message ID. Each module is then told to accept only messages with its own number in the appropriate bits of the identifier.

The BaBar standard CAN message identifier has the following structure:

Bit 10: A priority bit. It exists to allow users to force messages to a higher priority, if needed. In practice, this has been rarely done.

Bit 9: A direction bit. If this bit is zero, the message is from EPICS to the module specified in the module field. If it is one, it is a reply from the module to EPICS.

Bits 8-4 (5 bits): The module field. In general, up to 32 modules can exist on one bus, but for compatibility with a commercial standard used elsewhere in BaBar, we usually drop five of the possible values, leaving a maximum of 27 modules on the bus.

Bits 3-0 (4 bits): The command field. Together with the priority bit this field allows each module to accept up to 32 commands. Eight of the commands are BaBar-wide standards and the remaining 24 are available to users. By restricting eight commands for standard use we were able to develop tools that work on all boards following the BaBar standard. These tools are described in section 4.

2.3 Microcontroller Description

Each standard module has a Motorola MC68HC705X32 microcontroller (MCU) on it. This MCU has a CAN interface, four eight-bit bi-directional data ports, an internal watchdog circuit, and has EPROM, EEPROM, and RAM memory. The EEPROM memory is particularly useful for storing nonvolatile data such as station numbers and CAN register settings.

2.4 Interface Circuit

A standard circuit permits all BaBar modules to have a common interface between CAN and the MCU. A key feature of the circuit is the use of optical isolation to decouple the ground on the CANbus from the ground on the module. This prevents ground loops, which can interfere with monitoring and control.

Figure 2 shows a schematic diagram of this interface circuit. The PCA82C250 is a CAN interface chip and the HCPL710 is an optical isolator.

3. MONITORING BOARDS

3.1 General Description

A variety of monitoring boards has been developed to meet the needs of BaBar. Each of the six detector subsystems has developed at least one to meet its special needs and several boards exist which are used by more than one subsystem.

As the most widely used example of a BaBar monitoring board, we present a description of the General Monitoring Board (GMB). BaBar uses more than one hundred of these, and they are used by all six subsystems.

3.2 General Monitoring Board - Purpose

The GMB is used to monitor voltages, currents, and temperatures. A block diagram is shown in figure 3.

Up to 32 differential analog signals can be read by the GMB and digitized by a 12-bit ADC which covers the range of 0 V to +4 V in 1 mV steps. A passive network conditions the input signals and feeds them to an analog multiplexer (AMUX). The ADC is read by the MCU in a continuous background scan. When the MCU receives a CAN message requesting data, it sends the stored values from the most recent scan.

In addition, the eight signals from MCU I/O port C are brought to a 10-pin connector for such uses as driving relays, setting alarms, and reading valve positions. Care must be taken when using these signals because there is no buffering to protect the MCU and to prevent ground loops from the external connection.

An on-board DC-to-DC converter provides +9 V and -9 V bias voltages for the analog multiplexer and for biasing solid state temperature probes as described below.

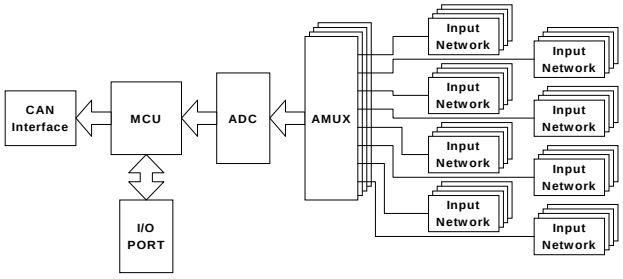

Figure 3: GMB Block Diagram

3.3 General Monitoring Board - Description

The module measures 120 mm by 100 mm. The CAN signals arrive on a DB-9M connector which is daisy-chained to a DB-9F connector for the CAN output. The 32 analog signals arrive on a 2x32 DIN connector. The port C signals are connected to a 10-pin low profile Amp connector.

A two-pin connector provides the power. Originally, the board used +5 V power but a later modification added a regulator to allow us to distribute +12 V power and reduce it to +5 V on the board. This eliminated problems associated with voltage drops over long power cables.

Front mounted LEDs display a steady green signal when power is on and an amber flash when a CANbus message is sent or received. Figure 4 shows a photograph of the GMB.

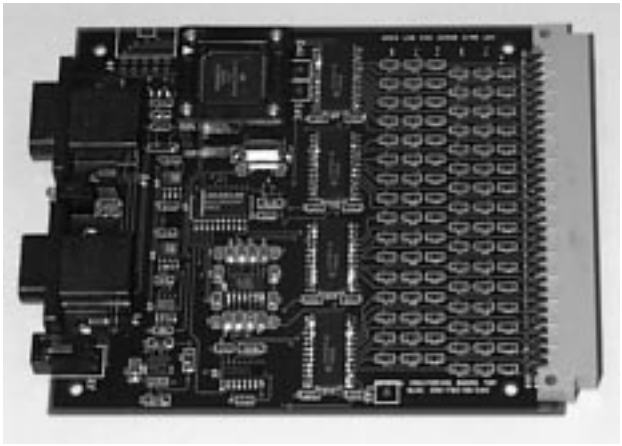

Figure 4: GMB Photograph

A passive input circuit on each channel conditions the input signal. This circuit can convert a current source into a voltage, it can scale an input voltage via a voltage divider, and it can provide a bias voltage for solid state temperature sensors and convert their returned current to a voltage. These are shown in figure 5. The first diagram in the figure shows the general circuit, which consists of two resistors, two jumpers and one optional filter capacitor to reduce high frequency noise.

The second circuit shows the configuration for use with AD592 temperature sensors from Analog Devices [6]. J1 is set to provide a nominal +9 V bias to the sensor, which produces 1 μA of current per degree Kelvin. This current flows across a 10 K resistor to convert it into a voltage with 10 mV per degree Kelvin.

The third circuit uses R_S and R_L to form a voltage divider to scale the input appropriately. Note that Sense_lo is not necessarily connected to the board's ground via J2.

3.4 General Monitoring Board - Serial Port Version

A modified version of the GMB brings out the serial port from the MCU to permit communication with RS-232 and RS-485 devices. The input signals have active filtering which permits a zero-offset adjustment on each

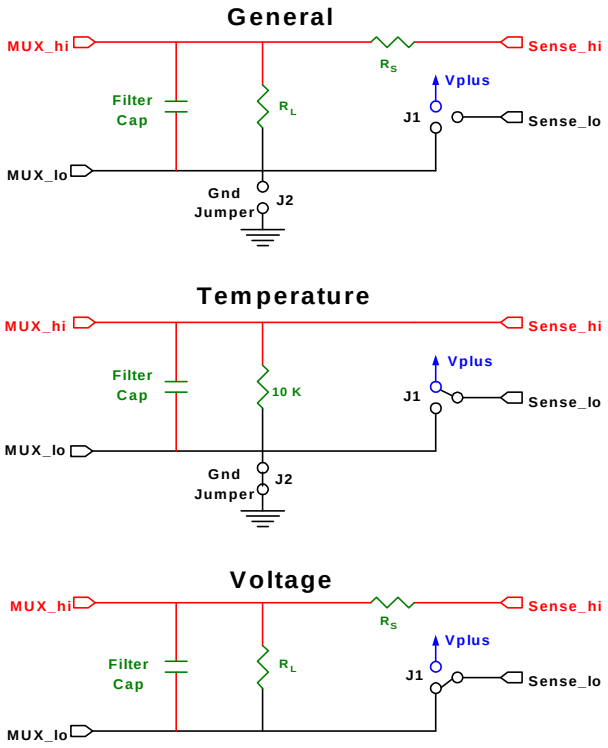

Figure 5: Input Networks

channel. This modified version is made on a 6u VME board.

3.5 General Monitoring Board, Version 2

A revised version of the GMB, called the GMB-2, is being designed. In addition to the capabilities already described, this board will contain a serial interface, allowing it to be read from standard serial ports. In addition, more display indicators are provided and more protection has been included to increase reliability. This version will be marketed by BiRa Systems of Albuquerque, NM [7].

4. SOFTWARE

4.1 Core Microcontroller Software

The MCU software is designed to have a set of core routines common to all BaBar boards. These routines call standard user routines for user-specific code. In the standard software distribution, the user routines are simple stubs.

Core routines are used for such tasks as reading the CAN interface and MCU registers, accessing RAM and EEPROM, and reporting the board status,

4.2 User Microcontroller Software

User-supplied software responds to CANbus commands not included in the set of eight reserved core commands. This software performs user-specified tasks in the background mode, and responds to all interrupts not handled by the core CAN interrupt service routine. This is where users install the unique functionality of their boards.

4.3 VxWorks Software

Utility software exists on the EPICS VME single board computer, which runs the VxWorks operating system from Wind River Systems [8]. Routines exist to scan an entire CAN bus for boards responding to the BaBar standards and to list their serial number and operating parameters. Other routines allow users to change a board's station number and to enable and disable the watchdog circuit on individual boards or all boards on a bus.

4.4 EPICS Applications

Several general tools exist at the EPICS level. These are client applications that provide graphical displays. The most important of them is called canProbe. It allows users to compose and receive arbitrary CAN messages, and it works on any CAN module, not just those following the BaBar standard. Other tools are board specific. Most notable is an application to retrieve and display all 32 channels from a GMB.

5. PERFORMANCE

After an extensive commissioning period using cosmic rays, the BaBar detector is now collecting data. CANbus modules are working in all subsystems and operate reliably. The use of common tools and common hardware has greatly reduced the work in building the monitoring and control system, and allows for a greater pool of expertise in running the detector.

6. ACKNOWLEDGEMENTS

Many people assisted in this effort. We particularly wish to thank Steve Lewis and Carl Lionberger for help with EPICS. Bill Thomas, Dave Nelson, Gunther Haller, and Angel Angelov provided engineering support. Technical assistance was received from Lee Harker and Mark Freytag. We also thank the many BaBar CANbus users who contributed useful comments and suggestions.

7. REFERENCES

[1] CAN was invented by Robert Bosch GmbH, Postfach 50, D-7000, Stuttgart, Germany. Additional information can be obtained from CAN in Automation (CiA) at Am Weichselgarten 26, D-91058 Erlangen, Germany.

[2] The MC68HC705X32 is manufactured by Motorola Corporation. Literature is available at Motorola Literature Distribution, P.O. Box 20912, Phoenix, AZ 85036 or at hppt://motserv.indirect.com.

[3] EPICS was originally developed at Los Alamos National Laboratory and Argonne National Laboratory.

Much support and development continues to come from its user community. More information can be found at the following web sites:

(i) http://mesa53.lanl.gov/lansce8/Epics/epics.htm.

(ii) http://epics.aps.anl.gov/asd/controls/epics/EpicsDocumentation/WWWPages.

[4] Greenspring Computers, 181 Constitution Drive, Menlo Park, CA, 94025, or at the web site http://www.greenspring.com.

[5] TEWS Datentechnik GmbH, Am Bahnhof 7, D-25469 Halstenbek, Germany.

[6] Analog Devices, One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106 and at the web site http://www.analog.com.

[7] BiRa Systems, 2404 Comanche N.E., Albuquerque, NM 87107, and at info@bira.com.

[8] Wind River Systems, Inc., 1010 Atlantic Ave., Alameda, CA 94501-1153.