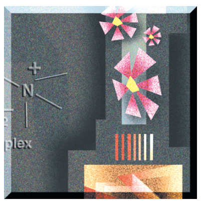


A Controller Area Network–Based Telemetry and Command System for Small Space Experiments

F. Charles Dumont, Joseph J. Suter, and Paul D. Schwartz

Controller area networks (CANs) have been used since the mid-1980s to connect microcontrollers and allow them to exchange data in an error-resistant manner. Until recently, their use has been limited to industrial and automotive applications. The Polymer Battery Experiment (PBEX) is a small spaceflight system developed at APL that will use a CAN to interface its telemetry and command system to the spacecraft computer. This type of network is attractive for space applications because of its built-in error detection and because it is a two-wire network, which helps to reduce spacecraft weight. This article describes the PBEX CAN and discusses the implementation issues that were encountered during the design of the experiment. (Keywords: Controller area network, Spacecraft systems, Telemetry.)

INTRODUCTION

A telemetry and command system on an instrument or experiment serves a dual purpose: it gathers and formats output data and it executes uplinked command messages. Telemetry must be converted to digital form, and formatted for transmittal and/or storage. Commands must be checked for validity, decoded, and implemented. Verification of command execution may also be required. In the past, these functions were often centralized within the spacecraft computer. Now, however, it is more common to see spacecraft in which each experiment performs its own command and telemetry handling while the central computer acts as a scheduler and coordinator. This has been made possible by advances in electronics miniaturization and the advent of small and powerful microprocessors.

The two main advantages of placing these functions within each experiment are that the processing power of the spacecraft computer is freed for more demanding applications, such as attitude determination and autonavigation, and that the size of the spacecraft electrical harness is significantly reduced. In a centralized system, at least one wire is required in the harness for each telemetry item and for each command. In a complex spacecraft, this can make for a very large and heavy harness. When the control is distributed, these wires typically become traces on a printed circuit board, which is a much more compact medium than a cable harness. The interconnection between the spacecraft computer and the experiments can now be reduced to a few control and data lines. This simplicity does come at a price, however. The experiments must become more complex, and protocols must be developed to communicate with the spacecraft computer. The differences between the two architectures are made apparent in Figs. 1a and b.

One protocol for data communications is the controller area network (CAN) developed by the German company Robert Bosch[1] in the mid-1980s and updated by Philips Semiconductor.[2] This protocol is a two-wire serial data communications specification that allows microcontrollers distributed throughout a system to exchange data over cable lengths of up to 40 m with a maximum transmission speed of approximately 1 Mbit/s.[2] Longer lengths are attainable by decreasing the data rate. The protocol was originally designed for microcontrollers and smart sensors within automobiles to communicate with each other, but it has since become widely used in industrial control applications.

A small spaceflight experiment developed at APL, the Polymer Battery Experiment (PBEX),[3] is using a CAN to communicate with the spacecraft computer. The experiment will test the performance of advanced rechargeable polymer batteries in the space environment and will be launched onboard the PicoSat spacecraft being built at Surrey Satellite Technology, Ltd., in Great Britain. This article discusses the hardware and software design issues that were involved in implementing the experiment's telemetry and command interface.

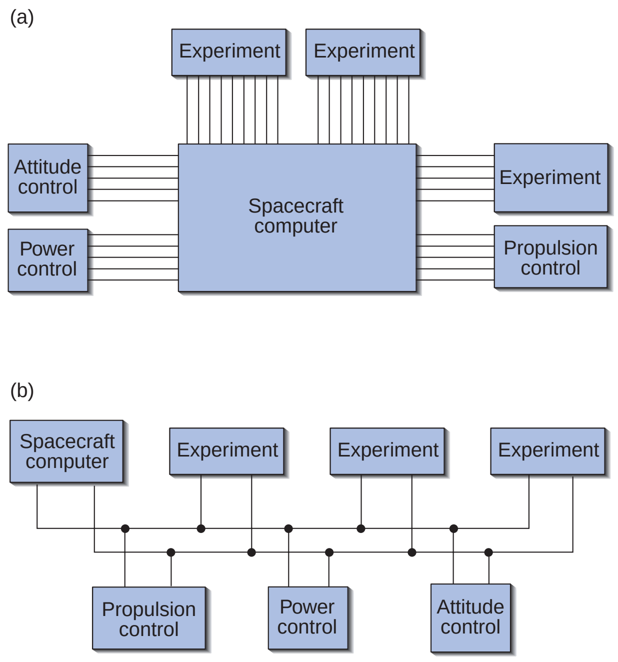


Figure 1. Spacecraft telemetry and command system architectures: (a) centralized, (b) distributed.

OVERVIEW OF THE CONTROLLER AREA NETWORK PROTOCOL

CANs are composed of a number of nodes connected together in a daisy-chain fashion by a pair of shielded or twisted wires. Logic levels are derived from these two wires differentially. That is, logic 1 is represented by a positive voltage difference between the signals on the two wires, while logic 0 is given by a negative voltage difference. CAN messages are transmitted serially over the bus from one node to all other nodes. Each message contains a header with an identifier that specifies the contents of the message (e.g., input voltage, state of on/off switch, switch batteries to "on," etc.). Using this identifier, a node will filter out all messages except those that are intended for it. The way a node filters out messages depends on the type of CAN protocol implemented. Two different types exist: basic CAN and full CAN.

In basic CAN, a node is interrupted every time it receives a message. It must then examine the message identifier to determine if the message is relevant to the node. In a full CAN implementation, hardware filters all messages at the front end of each node screen and only interrupts the node's microcontroller when the identifier matches one of those to which the node responds. This implementation of the protocol is much preferred over basic CAN because it simplifies software programming for the nodes. Most commercially available CAN microcontrollers now support full CAN. The identifier also helps eliminate bus contention, which occurs when two nodes try to transmit data over the bus simultaneously. In a standard serial bus, this can cause damage to electronic components. With a CAN system, though, the message with the lowest identifier has the highest priority. Therefore, if two nodes try to transmit a message at the same time, the message with the lowest identifier will take control of the bus and prevent the other message from being transmitted.

Another advantage of CAN is that error detection is built into the protocol. There are four types of error flagging. Cyclic redundancy codes are computed for each message and transmitted as part of the message. The receiving node also computes this code and detects an error if there is a mismatch. Additionally, formatting bits within the message must have certain values or the message will be discarded as erroneous. Stuffing bits are also inserted after any five consecutive bits having the same value. Therefore, an error will be raised if a receiver detects six or more identical bits in a row. Finally, a send-and-acknowledge scheme performs further error checking. When a node receives a message, it must acknowledge its successful receipt by driving the bus low for a specified amount of time. The transmitting node will continue to transmit the message until it receives this acknowledgment or until a specified amount of time has passed. If a message is found to be in error, instead of acknowledging it, the node that flagged the error will send an error message to all other nodes, including the original transmitter. This will abort transmission of the message and will alert all of the nodes that the message is erroneous. The data and formatting fields of a standard CAN message frame are shown in Fig. 2.

POLYMER BATTERY EXPERIMENT TELEMETRY AND COMMAND SYSTEM

Controller Area Network Implementation

PBEX implements the full version of the CAN protocol. However, the identifier of a message has a slightly different meaning in the PicoSat network than that defined in the CAN specification. Instead of a unique identifier being assigned to each of the possible messages in the system, in the PicoSat implementation, the identifier represents the node, or experiment, for which the message is intended. PBEX is just one of the nodes on the network. Further formatting within the data field of the message describes the source, type, and content of the message. As would be expected, the message type specifier identifies whether the message contains a command or a telemetry item. For each message that is transmitted over the CAN bus, a matching return message is sent in the opposite direction to either acknowledge the message or to indicate that it is invalid. Thus, six types of messages are defined for PBEX:

- Telemetry request
- Telemetry request acknowledgment
- Telemetry request error
- Command
- Command acknowledgment
- Command error

Hardware Description

At the heart of the PBEX telemetry and command interface is a 16-bit microcontroller, the Siemens SABC167CR. This microcontroller has an integrated CAN protocol module that is configured to transmit data at 1053.257 kbps, which is right at the limit of achievable data rates. Data sent over the CAN bus are received by this module, checked for errors, and checked for a matching identifier. This is done entirely in hardware. If no errors are found and the message identifier is valid for PBEX, an interrupt is generated to alert the microcontroller that a new message has arrived. The CAN module has 15 message buffers that can be configured as either receive or transmit buffers. Because all messages sent to PBEX will have the same identifier, only one receive buffer is enabled. A second message buffer is configured as a transmit buffer for responding to all messages sent by the spacecraft computer. Figure 3 shows a functional block diagram of the PBEX telemetry and command interface.

The spacecraft requests eight analog telemetry items and one digital item from PBEX every 10 s. Since each telemetry item is 16 bits wide, this translates to a data rate of 152 KB per day. Another option would have been to let PBEX transmit these data without the need for a request, which would further decrease the load on the spacecraft processor. Although this was allowed, it would have required synchronizing the PBEX clock with the spacecraft clock and the added complexity was undesirable. The analog data items represent the voltages, currents, and temperatures of the two batteries located within the experiment. The digital item is made up of the status bits (on/off) for six relays. Analog telemetry items are converted to digital data by the microcontroller's integrated 10-bit analog-to-digital converter. Commands can be either relay commands or internal state commands. Relay commands cause the microcontroller to produce pulses, which are amplified to switch the state of electromechanical latching relays.

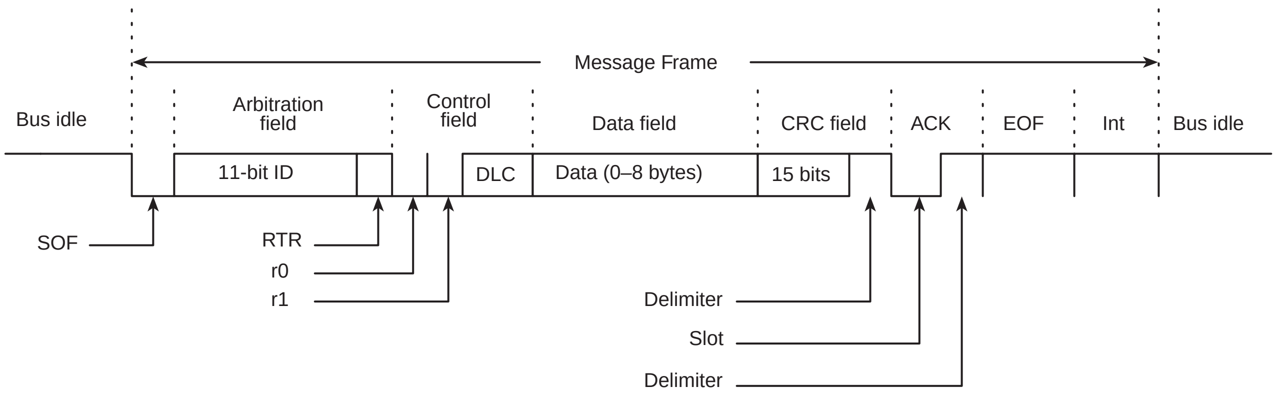


Figure 2. Controller area network standard message frame. (SOF = start of frame, RTR = remote transmission request bit, DLC = data length code, CRC = cyclic redundancy code, ACK = acknowledge bit, EOF = end of frame, Int = interval, and r0, r1 = reserved bits for future use.)

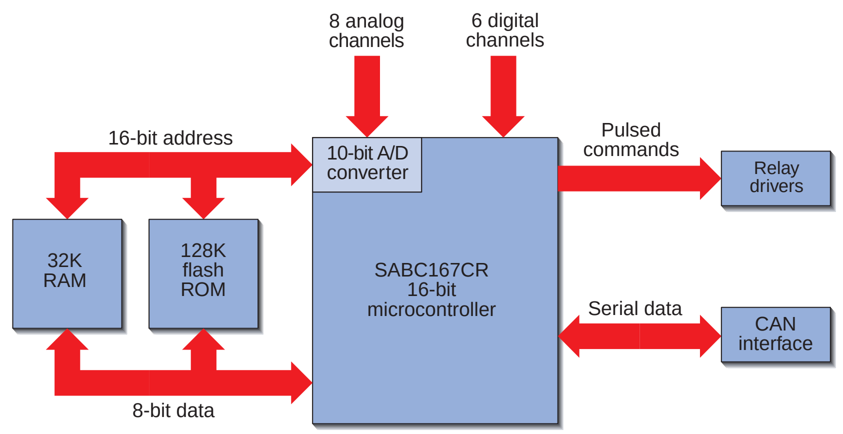


Figure 3. Polymer Battery Experiment telemetry and command interface.

Internal state commands change the value of variables in the experiment's software. These variables are parameters used in the experiment as part of its autonomous battery charging algorithms. A total of 31 different commands are defined for PBEX.

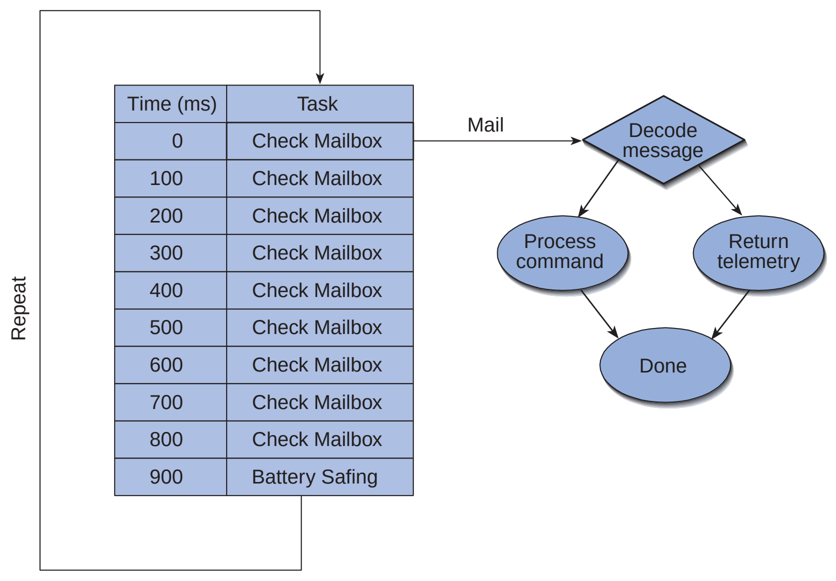
As mentioned earlier, a CAN-based telemetry and command system reduces the size of the electrical harness between the spacecraft computer and the experiments. If a centralized approach had been used instead, PBEX alone would have required 14 individual wires for its telemetry and 31 wires to receive commands. The two wires required in a CAN implementation represent a 96% savings in harness weight. Given that PBEX is the simplest of the subsystems on PicoSat, the overall weight reduction is even greater.

Software Description

The PBEX flight software is built around a main loop that repeats indefinitely. This loop is structured in a deterministic fashion to ensure that central processing unit resources will always be available for the current task. To do this, the loop is divided into 10 tiers of equal length. Each tier is guaranteed to last 100 ms by starting a timer upon entering the tier. If the task finishes before the timer expires, the tier waits until the time is up before passing control to the next tier. With 10 tiers, the main loop is guaranteed to repeat every second. This makes software debugging and timing analysis much simpler because the particular task being performed at a certain time is always known. It is not dependent on what interrupts have occurred in the system. This type of software architecture is relatively insensitive to changes in the system clock speed. The only change that would be required for a different clock would be the hexadecimal reload value of the 100-ms timer. Figure 4 illustrates the structure of the PBEX main loop.

Only two types of tiers exist in this structure: Check Mailbox and Battery Safing. Check Mailbox does just that; it checks to see if a new message has arrived on the CAN bus. This will be indicated by a flag set by the CAN interrupt. The interrupt service routine for the CAN module merely sets this flag and moves the message to a buffer. It does not process the message. Up to three messages can be waiting to be processed in the buffer. Any additional messages will be ignored. If a message is waiting, it is decoded to determine if it is a telemetry request or a command. Depending on the message type and content, an analog-to-digital conversion may be started, a relay may be pulsed, or some other action will occur. The appropriate response will then be sent back to the spacecraft computer over the CAN bus, and the tier will wait until the 100 ms are up.

The last tier in the main loop, Battery Safing, performs an analog-to-digital conversion on either the voltage or current for the battery being exercised. This



Time (ms)	Task
0	Check Mailbox
100	Check Mailbox
200	Check Mailbox
300	Check Mailbox
400	Check Mailbox
500	Check Mailbox
600	Check Mailbox
700	Check Mailbox
800	Check Mailbox
900	Battery Safing

Figure 4. Deterministic software loop for the Polymer Battery Experiment.

value is compared against a limit that has been set for each battery to determine if it is time to switch from charge to discharge mode or vice versa. The tier also checks the battery temperature to ensure that it does not exceed a safe limit. If it does, the tier disconnects the battery from its circuit. All of the limits for the batteries can be changed by command from the ground.

Although these functions are relatively simple, it is easy to see that they would consume valuable spacecraft processing time if they had to be performed by the spacecraft computer. The advantages of a distributed telemetry and command system are evident. The reduction of the spacecraft computer's load offered by a distributed system is difficult to quantify since, for example, the process of getting a telemetry item is shared between PBEX and the spacecraft computer. But if we consider just the Battery Safing tier, with a centralized system the spacecraft computer would have to spend 100 ms out of every second checking the limits for the batteries and acting accordingly. This alone is a 10% savings in processing time. If it had to perform the same sort of tasks for the three other experiments on PicoSat, a 40% savings would be realized by a distributed system.

CONCLUSION

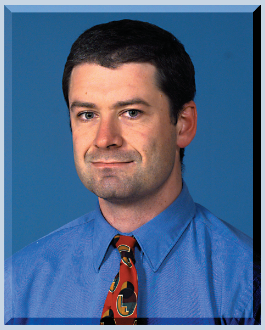
CANs are an attractive method to connect the experiments on a spacecraft because of their inherent error detection capabilities and because of the large degree of flexibility they offer for configuring a spacecraft's telemetry and command system. However, the one drawback to using them is that the available microcontrollers that incorporate a CAN protocol module are all commercial components. Their susceptibility to ionizing radiation may be a problem. PBEX will determine how much of a problem this is for the Siemens SABC167CR microcontroller. Unfortunately, chip manufacturers will not invest in radiation hardening of their components until there is a proven market for them. And this market will not be established unless the gains realized from using a CAN network are made apparent to spacecraft developers. It is our hope that the success of PBEX will demonstrate those gains.

REFERENCES

[1]Robert Bosch GmbH, *CAN Specification, Version 2.0*, Stuttgart, Germany (1991).
[2]Philips Semiconductors, *CAN Specification, Version 2.0*, Publication 5452, Sunnyvale, CA (1992).
[3]Dumont, F. C., Suter, J. J., Lew, A. L., Schwartz, P. D., Le, B. Q., et al., "Polymer Battery Experiment: Novel Power System for Space Applications," in *Proc. AIAA Defense and Civil Space Programs Conference and Exhibit*, Huntsville, AL (Oct 1998).

ACKNOWLEDGMENTS: We would like to thank the engineers at Surrey Satellite Technology, Ltd., for their invaluable assistance with the development of the PBEX CAN interface. Their patience and perseverance in helping us to understand the CAN protocol played a large role in the successful outcome of the PBEX project. We also gratefully acknowledge the contributions to PBEX from Rome Laboratory in Rome, New York.

THE AUTHORS



F. CHARLES DUMONT received his B.S. in electrical engineering from the University of Virginia in 1991 and his M.S. in aerospace engineering from the University of Colorado in 1997. Mr. Dumont joined the Space Department upon completion of his M.S. degree and has since been the lead engineer for the Polymer Battery Experiment. He also played a vital role in the development of the APEX visible and ultraviolet spectrographs. Currently, he is the lead design engineer for a system that will allow remote sensors to be monitored over a cellular phone link. His e-mail address is francis.dumont@jhuapl.edu.

JOSEPH J. SUTER received a B.S. degree in physics and mathematics from the Free University of Amsterdam, The Netherlands, in 1977, an M.S. degree in physics from Michigan State University in 1980, and an M.S.E.E. degree from the University of Maryland in 1983. In 1988, he was awarded a Ph.D. degree in materials science and engineering from The Johns Hopkins University. Dr. Suter joined APL in 1983 and is a Principal Professional Staff scientist and program manager in the Space Department. He also serves on the APL Internal Research and Development Advisory Council and chairs the Sensor System Thrust Committee. He is a member of IEEE, ISA, and SPIE, and has served on several external sensor technology committees. He is the (co)author of over 50 technical publications. Dr. Suter, together with his colleagues in the Space and Submarine Technology Departments and the Research and Technology Development Center, teaches a graduate sensor system course in The Johns Hopkins University Applied Physics Program. His e-mail address is joseph.suter@jhuapl.edu.

PAUL D. SCHWARTZ received his B.S. and M.Eng. degrees in electrical engineering from Cornell University. Since joining the APL Space Department in 1973, Mr. Schwartz has been the lead design engineer for the development of numerous spacecraft subsystems including the AMPTE Command System, the COBE Momentum Management Assembly, and the MSX Data Handling System. Mr. Schwartz was the lead hardware design engineer for the NEAR Command Telemetry Processor. He has designed power conditioning electronics for several space instruments including the Galileo EPD and the Ulysses LAN Experiment. He is currently the system engineer for the development of a miniaturized visible imager. His e-mail address is paul.schwartz@jhuapl.edu.