CAN BUS
USED FOR DATA ACQUISITION SYSTEM CONTROLS
(AUTOMOTIVE SOLUTION FOR AIRCRAFT PROBLEM)

Bruce Johnson
NAWCAD Patuxent River, MD

And

John Smith
NAWCAD Patuxent River, MD

ABSTRACT

This paper discusses using the CAN (Control Area Network) Bus protocol for control and status of flight test data acquisition systems. The application of the CAN (Control Area Network) on an F/A-18 aircraft will be discussed in detail.

KEYWORDS

CAN (Control Area Network), EMI (Electromagnetic Interference), COTS (Commercial Off The Shelf), Printed Circuit Board (PCB)

INTRODUCTION

A substantial amount of resources are required to define, and implement how a flight test data acquisition system will be controlled in the aircraft. Due to unyielding requirements on safety and reliability dedicated flight test control wiring is typically installed to support these functions. Typically in flight test, aircraft modification time is very limited. Spare production wires can be used for discrete control/status signals. The drawback to this approach is that the number of spare wires, if any, limits the ability to control the system and quality of feedback provided to the aircrew. Installing shielded twisted pair wires to support RS-232 or RS-485 type serial protocols is another option. However due to the constraints for the F/A18 aircraft (listed below) another option for the F/A-18 data acquisition system control/status was needed. This paper will discuss the utilization of the CAN bus to meet the specific requirements for control and status of the data acquisition system on a production representative F/A-18 aircraft.

The constraints on the F/A-18 were as follows:

1. No flight test wiring installation on fleet aircraft will be permitted.
2. Not enough spare production wires were available to control the system with discretes.
3. The spare wires that are available are not twisted pair and are unshielded, (i.e. susceptible to EMI noise).
4. The chosen solution must be inexpensive (< $500 per system)
5. The chosen solution must be smaller than 9 cubic inches volume.
6. The chosen solution must be supportable using COTS (Commercial Off The Shelf) components and test tools.
5. The system must operate using 28 VDC @ < 1Amp.

Technical Issues:

1. The spare production wiring available is not twisted pair or shielded wire. Using wiring of this type violates the physical layer specification of serial communication protocols. It also makes the system extremely vulnerable to EMI noise present in the F/A-18 operating environment.
2. The serial communication protocol will need to be operational as soon as power is applied to the aircraft. Cockpit Control Panel to Data Acquisition System communication must be established to provide the pilot the ability to power and control the system as well as receive status information.

Various serial communication protocols were researched to determine if they were able to meet these requirements. The (Controller Area Network) bus or CAN bus was selected because it provides proven reliability and high noise tolerance. The CAN bus is widely used in the automotive industry, where the EMI environment in the engine compartment of a car/truck is quite ”dirty”. In addition CAN is a widely accepted standard with many COTS components to design, test, and build a very scaleable system.

Control Area Network (CAN) is a common small area network solution that supports distributed product and distributed architectures. The CAN bus is used to interconnect a network of nodes or modules. Typically a two wire, twisted pair is used for the network interconnection.

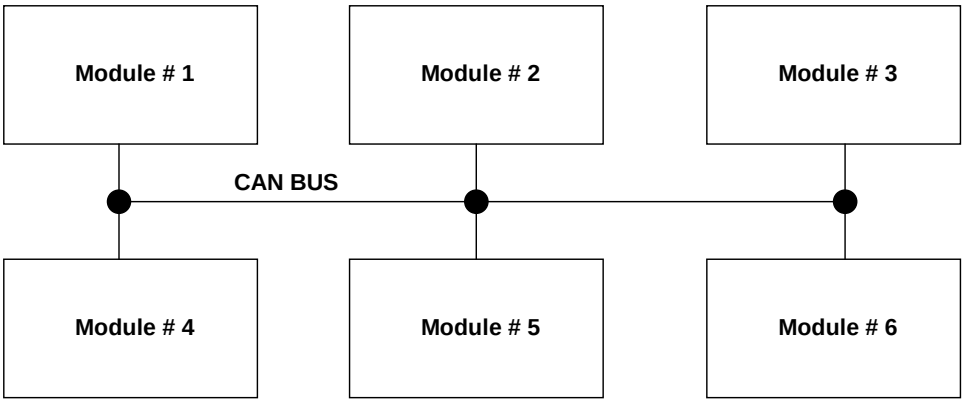

Figure 1.0 Typical CAN bus Implementation

WHY USE CAN?

CAN is a widely accepted standard in the automotive industry, as well as heavy trucks, busses, construction equipment, factory automation, appliances, medical, and marine industry.

CAN is very robust, has good error detection capabilities, simple to configure, low cost components, excellent two-wire fault tolerance capabilities (i.e. either of the two wires of the bus is broken, shorted to power, or shorted to ground.)

Many sources of software and hardware support are available. These include CAN bus analyzers & development tools, and large international users group that offer conferences and training.

BASIC OPERATION OF CAN

CAN is a serial communication protocol that may be used to transfer up to 8 data bytes with in a single message. CAN supports data transfers between multiple peers. No master controller is needed to supervise the network conversation. The CAN message is bit-oriented, always begins with a “start of message” indication, includes an address (called identifier), may contain data, includes a Cycle Redundancy Check (CRC), requires an acknowledgement from all network members, and is converted to the appropriate signal. The signal is then placed onto the physical layer. The physical layer consists of transceiver that interfaces to the microcontroller and the wires of the bus. The bus wires are typically twisted pair shielded wires, however the CAN bus performs very well with non-twisted pair shielded wires as well.

Key Components of the CAN message:

SOF	Start of Frame- signals the beginning of a CAN message
Identifier	Indicates a message number (prioritized from high to low)
Control	Indicates the message data size and special information
Data	Contains message data (0 to 8 Bytes)
CRC	Provides data transfer reliability
ACK	Signals message acceptance by all network members

CAN provides a simple mechanism, called Bitwise Arbitration, to eliminate competing transmitters from colliding into each other during message initiation. This feature provides an added measure of transmission efficiency by using "dominant-recessive logic". Bitwise Arbitration priority is based on the user's configuration of the message number identifier field of the CAN message.

Bitwise Arbitration Rules

1. A "dominant" bit always wins over a "recessive" bit
2. A transmitter always compares its transmit bit with its receive bit
3. If a transmitter detects that the receive bit is not the same a the transmit bit, then the transmitter will immediately stop its transmission

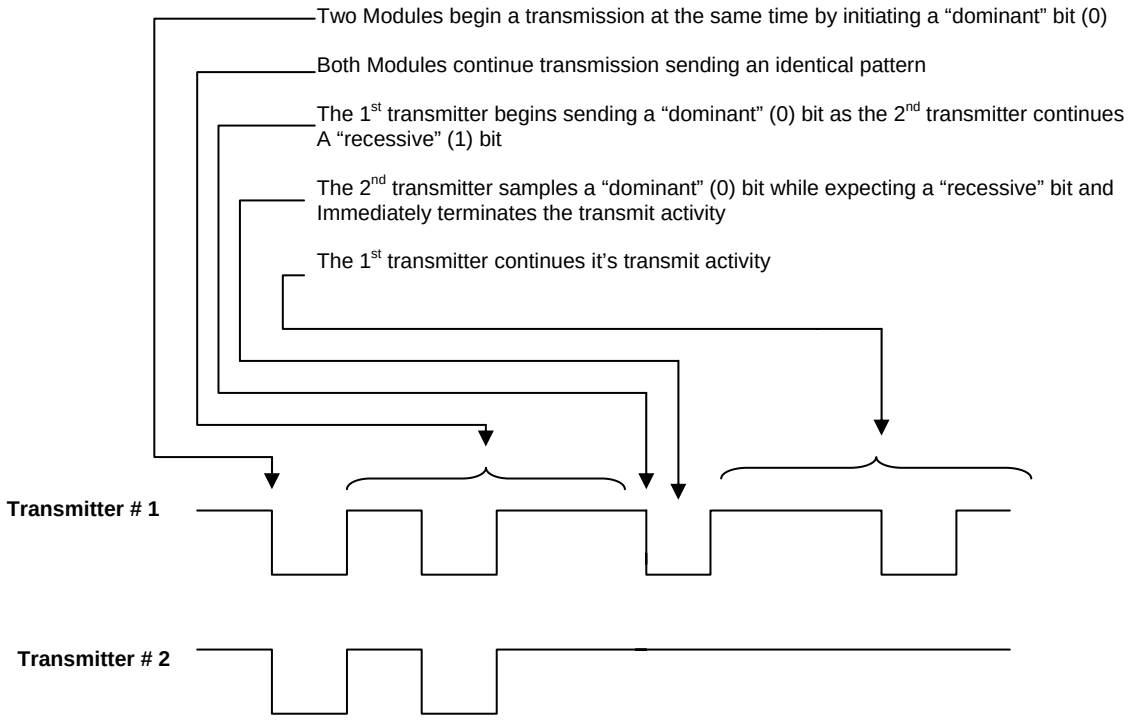

Figure 2.0 Example of Bitwise Arbitration

DETECTING AND SIGNALING ERRORS

Unlike other bus systems, the CAN protocol does not use acknowledgement messages but instead signals errors immediately as they occur. For error detection the CAN protocol implements the following mechanisms at the message level:

- Cyclic Redundancy Check (CRC): the CRC safeguards the information in the frame by adding a frame check sequence (FCS) at the transmission end. At the receiver this FCS is re-computed and tested against the FCS. If they do not match, there has been a CRC error.
- Frame Check: This mechanism verifies the structure of the transmitted frame by checking the bit fields against the fixed format and the frame size. Errors detected by frame checks are designated “format errors”.
- ACK Errors: Receivers of a message acknowledge the received frames. If the transmitter does not receive an acknowledgement an ACK error is indicated.

The CAN protocol also implements two mechanisms for error detection at the bit level:

- Monitoring: The ability of the transmitter to detect errors is based on the monitoring of bus signals. Each transmit station that transmits also observes the bus level and thus detects differences between the bit sent and the bit received.
- Bit Stuffing: The coding of individual bits is tested at the bit level. The bit representation used by the CAN is “Non Return to Zero (NRZ)” coding. The synchronization edges are generated by a means of bit stuffing. That means after five consecutive equal bits the transmitter inserts a stuff bit into the bit stream. This stuff bit has a complementary value, which is removed by the receivers.

HANDLING OF ERRORS

If one or more errors are discovered by at least one node using the above mechanisms, the current transmission is aborted by sending an error frame. This prevents other stations from accepting the message and thus ensures the consistency of data throughout the network. After transmission of an erroneous message that has been aborted, the sender automatically re-attempts transmission (automatic re-transmission).

F/A-18 CAN BUS IMPLEMENTATION

The following section will discuss the details on how the CAN bus was utilized to provide control and status for the F/A-18 flight test data acquisition system.

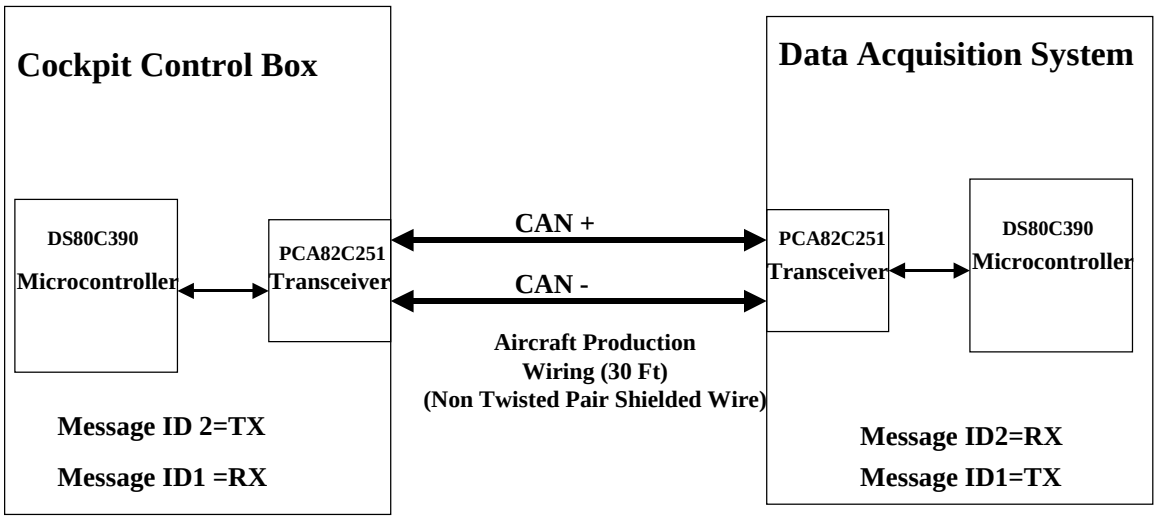

Figure 3.0 F/A-18 Data Acquisition System CAN Bus Block Diagram

MICROCONTROLLER

We chose the Dallas Semiconductor DS80C390 microcontroller for our CAN bus control system. The DS80C390 has two full-function 2.0B controllers with the status and control incorporated in the special function registers (SFR) and internal memory. In general, software that is written using the standard 8051 instruction set is portable to this microcontroller. The exception is the available higher clock speed of this processor requires the user to be mindful of timing issues; however, the default timing setting is the original 12 clocks-per-machine cycle used in the 8051. This microcontroller is widely used, well known, and will be supported for the foreseeable future. Future availability was a strong motivating factor in our decision to go with the DS80C390, due to the amount of effort that was required to setup reliable software for this specific embedded system. Additionally, we wanted a microcontroller that would be useful for other projects within our division. This microcontroller can run at up to 40 MHz if required or at very slow rate for low power consumption projects. The embedded system is a typical Intel 8051 layout, which our group has used in previous applications.

WATCHDOG TIMER

The DS80C390 microcontroller incorporates an onboard watchdog timer that will reset the microcontroller. A watchdog reset could be due to software locking up the processor

(e.g., infinite loop, deadlock state) or due to a power glitch corrupting the program counter, stack pointer, or RAM data. The high electrical noise environment of an aircraft requires a high level of protection and mitigation from power transients and spurious noise. If noise or transients corrupt the normal functioning of the processor the watchdog timer will timeout and reset the processor. The CAN protocol is robust enough to allow for this resetting interruption. The onboard watchdog timer is not unique to the DS80C390 as most microcontrollers incorporate a watchdog timer, although not necessarily on the microcontroller. An onboard watchdog timer has the advantage of fewer components and a smaller PCB board footprint. Our cockpit control box had very tight space requirements and the small PCB board footprint was another major requirement that the DS80C390 fit.

TRANSCEIVER

General Overview

The transceiver provides a very effective buffer and gateway between the CAN bus and the microcontroller controller. The transmitter in the transceiver provides power to drive the CAN bus, which decreases the amount of load on the microcontroller chip. The receiver will protect against voltage spikes and DC voltages that would damage most TTL circuits. Additional protection includes bus line short circuit protection, overload protection due to grounding of the CAN bus, and continued bus operation with a broken (open) line. It should be noted the transceiver is not the termination of the bus; the bus should be terminated with resistors that minimize signal reflections. This is very similar to how a MIL-STD 1553 bus is physically setup.

PCA82C251

We specifically chose the PCA82C251 CAN transceiver to meet the ISO 11898 standard or fault-tolerant CAN standard. This transceiver provides protection from transient voltages on the CAN bus of up to +/- 200V and a maximum DC voltage of +/- 40V. The PCA82C251 provides protection against short-circuit conditions on the bus along with thermal overload protection. The PCA82C251 will shutoff in the event of thermal overload. Additional protection such as galvanic isolation with optocouplers can be added, however this increases the loop delay due to the signal having to pass through the optocouplers twice, once for receiving data and once for transmitting data. This delay will decrease the maximum bus length that can be achieved at a preset bit rate. Although we would have liked to use optocouplers in our application, the space constraints of the cockpit controller precluded their inclusion.

Transceiver Modes

The PCA82C251 transceiver can operate in multiple modes depending on the resistance value applied to pin Rs. The first mode is for high speed and/or longer bus lengths. The second mode is Slope Mode Control mode, which is used when unshielded wire is used as the bus and the transceiver is required to deal with an electrically noisier environment. Adjusting the slew rate of the bus via an external resistor decreases the electrical noise susceptibility of the transceiver. Decreasing the slew rate by this method will also increase the bus node loop delay and therefore require either a decrease in the length of

the bus or a lowering of the bit rate of the bus. The third mode is stand-by mode that is used in applications where lower power consumption is a requirement. In stand-by mode a transmission reactivates the bus from a dormant state.
Our bit rate is 125kbit/s, which is the ISO 11898-3 fault-tolerant maximum specified bit rate. At that bit rate the maximum bus length for the high-speed transceiver mode of operation is 500 meters. The bus wire run in our F/A-18 using the production wiring is less than 50 feet. If we would have experienced problems with electromagnetic interference during testing we could have used up to the maximum slew rate modification with minimal concern for the decrease line length due to the increased bus node loop delay. Other implementation issues, discussed later, diverted our attention from this additional feature of decreasing the slew rate. The F/A-18 implementation did not require this additional EMI mitigation. This again shows the robust nature of the CAN bus system. Future applications in higher EMI environments might require the addition of slew rate adjustments. This would be accomplished by adding resisters to the present transceiver on the PCB board. Again the CAN bus protocol has the flexibility to allow physical modifications that are transparent to the microcontroller and code. The transceiver chip provides the pathway for the data from the microcontroller to the bus, whether the CAN bus uses aircraft production wires, power lines, or fiber optic networks. Other transceiver chips are available that meet the ISO 11898-3 fault-tolerant CAN bus specification. Key features of these transceivers are transparent at the microcontroller level and include, automatic bus error detection and asymmetrical signal transmission due to the failure of one of the two lines of the CAN bus.

CAN BUS SYSTEM

The CAN bus uses the broadcast communication mechanism where the message contents are defined versus defining nodes and node addresses. A benefit of a content-oriented addressing scheme is the ability to remove nodes without disrupting the other nodes left in the system. Nodes can be added without requiring any hardware or software modifications to the existing nodes, if the new nodes are receiving data from the existing system. The flexibility and modularity makes the CAN bus easy to upgrade or modify. We used the 11-bit identifier, also know as CAN base frame, for our system since we only have two nodes. CAN extended frame is available with 29 identifier bits if more node addresses are required beyond the 11 bit frame. The CAN bus assigns bus priority by the binary identifier setup for each message. The message with the lowest identifier value has the highest priority and conversely the message with the highest binary identifier value has the lowest priority. Each message is labeled with the binary identifier and each node checks the priority of the message to determine which one has the highest priority. Thus conflicts on bus priority are handled by predetermined prioritization setup by the system designer. Physically, this arbitration is handled by a bitwise ANDing of the message identifiers by the node. The lower priority node “waits” until the dominant messages are finished before attempting to gain control of the bus.

TOOLS

We chose the Keil compiler to develop our software and debug the system. This is a COTS software package that is highly recommended, however other compilers are available that work with CAN microcontrollers. The debugging system is very helpful and user friendly. The compiler must be setup to work with the DS80C390 and this can be a bit involved for the beginning user. Someone with a good background in Intel's 8051 microcontroller and general embedded system knowledge will find the Keil compiler has very useful debugging features. Another indispensable tool is the CAN bus emulator. We use the Dearborn Protocol Adapter (DPA), which is a serial communication adapter that connects a personal computer to the CAN bus via the computer's RS-232 plug. The DPA software allows you to monitor the bus traffic on the CAN bus. The user can determine which nodes are sending messages and the data that was sent. We are also able to test each node in our system with the DPA by sending messages from the PC onto the CAN bus as if we were the other node. This also allowed us to try different scenarios prior to developing working code in the sending node or the receiving node.

PROBLEMS ENCOUNTERED DURING DEVELOPMENT

1. One part of the coding that did give us trouble was the STARTUP.A51 file that the Keil compiler runs at system reset. Specifically, you must use the $NOMOD51 directive in the STARTUP.A51 file. This prevents the software Assembler from predefining the SFR to the 8051 microcontroller. We were then able to include the REG390.h file in the STARTUP.A51 file to tell the Assembler that the SFR should be setup specifically for the DS80C390. It was not well documented that the default state was for the Assembler to essentially ignore the included REG390.h file statement in the STARTUP.A51 file.

2. The data acquisition system that the CAN bus was interfaced to utilized a 115 VAC to 28 VDC transformer rectifier. On power up of the data acquisition system, a power spike was sent out that would intermittently scramble the contents of the E^2 PROM. Filtering and adding a time delay circuit resolved this issue.

3. We also experienced intermittent failures of the microcontroller. The microcontroller failures were traced back to noise spikes coupling from the data acquisition system to the microcontroller I/O ports. Isolating the I/O ports with relays resolved this problem

TESTING

It was cost prohibitive to flight test our system as the sole device under test, therefore we flew piggyback to another project after initial ground testing. Since the time on aircraft testing was limited the following risk reduction tests were done:

1. Lab test of prototype boards using emulator.

2. Lab test of prototype with 30 ft of non-twisted pair, non-shielded wire connecting the cockpit control box to the data acquisition system controller. Verified systems response to temporary errors (i.e. shorts/opens of CAN bus)
3. Ground test on aircraft, using prototype cockpit control box PCB and data acquisition system controller PCB and the production 30ft wire run in the aircraft. This allowed the systems communications to be verified with avionics running to simulate the EMI environment. Verified systems response to temporary errors (i.e. shorts/opens of CAN bus.
4. Lab test of prototype PCB boards packaged into the flyable cockpit control box, and data acquisition system controller.
5. Flight test of system on 3 flight tests, no control problems observed.

CONCLUSION

CAN is a proven, reliable method of bus communications. Due to the large-scale use of CAN microcontrollers, the cost and scalability, CAN is a very good choice for control and status of data acquisition systems. The F/A-18 implementation of the system has been very successful when operating in the harsh military aircraft environment. We have varied our implementation of the CAN bus on the F/A-18, to have several packaging configurations, depending on a specific projects needs while using the same printed circuit board layout. Minor changes to the microcontroller software will allow the flexibility to meet new specific project requirements.

REFERENCES

1. Vector-cantech.com
2. CAN 2.0B
3. Dallas Semiconductor
4. www.can-cia.org/can/protocol