SAE The Engineering Society For Advancing Mobility Land Sea Air and Space®

A Product of the
Cooperative Engineering Program

# SAE J1567 AUG87

# Collision Detection Serial Data Communications Multiplex Bus

SAE Information Report
Issued August 1987

SAE The Engineering Society For Advancing Mobility Land Sea Air and Space
400 COMMONWEALTH DRIVE, WARRENDALE, PA 15096

# HIGHWAY VEHICLE REPORT

SAE J1567

Issued August 1987

## COLLISION DETECTION SERIAL DATA COMMUNICATIONS MULTIPLEX BUS

1. INTRODUCTION: The serial communications bus and bus interface special function integrated circuit defined herein was intended to provide a simple, yet reliable, data communications link between members of a distributed processing vehicle multiplex system. The communications protocol chosen minimizes the software support overhead requirement of the modules on the multiplex bus. Appendix A contains a detailed technical summary of the Differential Serial Communications Bus protocol and Interface Integrated Circuit.

2. PROPOSED PROTOCOL: The network access method that best meets the requirements is known as CSMA (Carrier Sense Multi Access). A deterministic priority access method of resolving contention, by non-destructive bit-by-bit arbitration, was chosen instead of the nondeterministic random backoff procedure associated with classic collision detection. The message format chosen is shown below and will support a number of higher level protocols. Take note that idle periods are allowed between each byte of data. This permits the use of firmware control and direct connection to the host microcomputer's asynchronous serial I/O port.

   (SOM), (id 1), (id 2), (data 1), (data 2) ... (data n), (EOM)

   a - SOM, defines start of message
   b - id 1, 8 bit firmware addressing scheme
   c - id 2, optional identifier
   d - Data, may take any form, that is, data value, CRC, checksum, number of bytes in message, acknowledgement, etc.
   e - EOM, defines end of message

2. (Continued):

Standard byte synchronization NRZ was chosen as the bit encoding modulation technique. This data byte synchronizing method also provides message integrity by framing error detection. Three bit encoding methods NRZ, PWM, and Bi Phase were originally considered. The choice to use byte sync NRZ encoding as the modulation method was because it is believed to have the potential of providing the best compromise in bandwidth efficiency and data recovery in a noisy vehicle environment while maintaining the required high network data rates without generating excessive EMI.

The first thing considered was the data recovery performance in a noisy vehicle environment. The best performing data decoders should digitally approximate an integration over the bit period. In a noisy environment up to 1/2 of the samples can be corrupted for the NRZ encoding method before data is lost. PWM encoding method suffers a significantly smaller fraction dependent on the duty factor. This reduction in noisy environment performance cannot be compensated by just reducing the data rate.

The second thing considered was the EMI generated by the different data bit encoding method. Some empirical data was collected in a vehicle experiment. The data, although not completely conclusive, indicated that NRZ could operate at a significantly higher bit rate than PWM or Bi Phase encoding methods. It was also felt that the main advantage to using PWM or Bi Phase encoding is that the data bit rate is also encoded in the signal and, therefore, operation can be maintained using a larger clock tolerance. The NRZ requirement for a more precision clock was not a large advantage because all module applications already demanded a crystal or ceramic resonator.

The need for an open network so that minimal effect would result when nodes are added or removed, led to the selection of a broadcast type system. The identification (ID) byte of the message data content is unique and is broadcast across the network. The collision detection hardware within the interface IC determines which transmitting ID's wins without losing bus time when contention occurs. This action satisfies the need for establishing access priority and, therefore, gaining control of the latency for important messages. The message format and the data integrity method is determined by firmware. Maximum flexibility is the result and, thus, the software developer's creativity is not restricted for future applications.

3. FUNCTIONAL DESCRIPTION: Interfacing between a microcomputer and the bus is fairly simple. There are basically three methods; Asynchronous serial, synchronous serial, and parallel to the address and data I/O port. The cost of the parallel interface was avoided by development of a low cost serial interface IC. To make it universally interfaceable to most microcomputers three modes of operation are supported in one device; (1) SCI, (2) SPI, and (3) Buffered SPI.

3. (Continued):

The circuits of the interface IC used when connected to the microcomputer SCI are basic to the operation of all the other modes of serial communications. Therefore, this mode will be explained first. The components of the device (See Fig. 1) include the following:

a - A contention permitting differential transceiver
b - A collision detector
c - An arbitration detector
d - A digital filter
e - A bus idle detector
f - A timing and synchronizing circuit

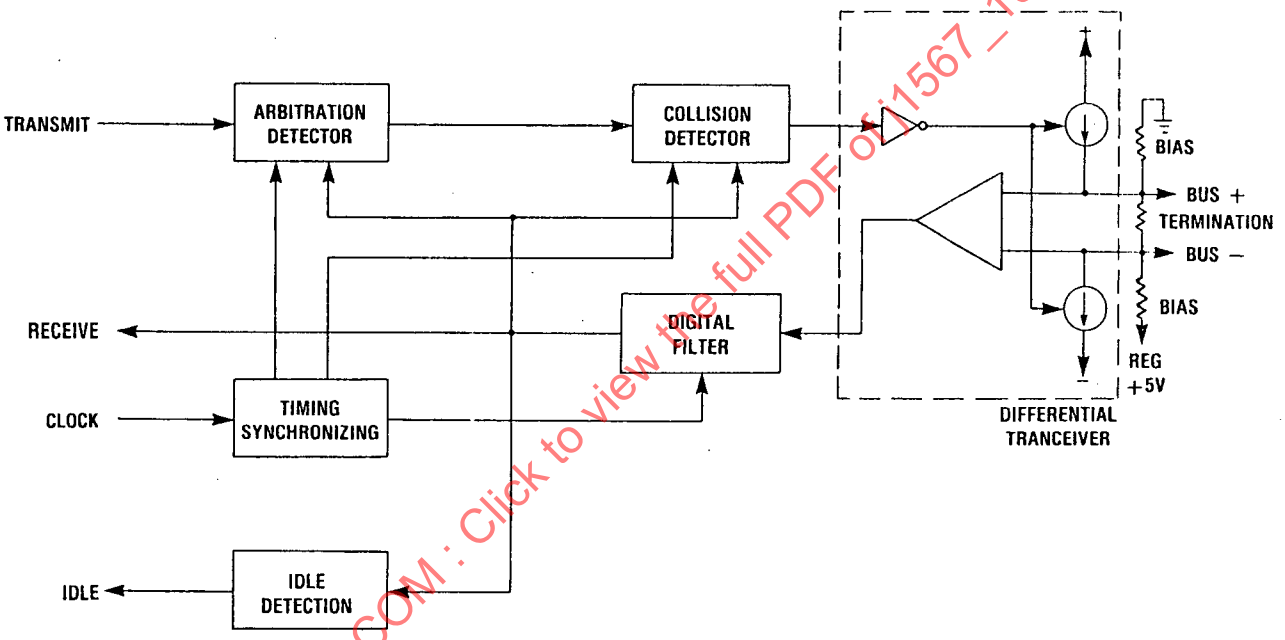



FIG. 1 - SIMPLIFIED BLOCK DIAGRAM OF BUS INTERFACE

3.1 Contention Permitting Differential Transceiver: The differential transceiver is a serial interface device which accepts digital signals and translates this information for transmission on a two-wire differential bus. The transmitter section, when transmitting, shall provide matched constant current sources to the bus "+" and bus "-" drivers (Ref. Fig. 1), sourcing and sinking current respectively. When a logic zero is supplied to the "transmit data" input, the differential amplifier shall cause the bus "+" driver to provide source current and the bus "-" driver to provide a matched current sink. A logic one at the "transmit data" input must cause the bus "+" and bus "-" drivers to simultaneously provide a high impedance state.

3.1 (Continued):

The wired "OR" action of the transmitting section allows more than one device to transmit at the same time thus permitting data collisions. The nonsymmetrical action of the bus drivers will allow a transmission of a logic zero, from one device, to overpower the transmission of a logic one from other devices. In this manner two or more devices can simultaneously transmit and contend for the bus, each using a unique message ID byte. The winner is determined by the value (or priority) of the ID byte, without losing bus time. A logic zero bit in one message ID byte has priority over a logic one bit in another message ID byte.

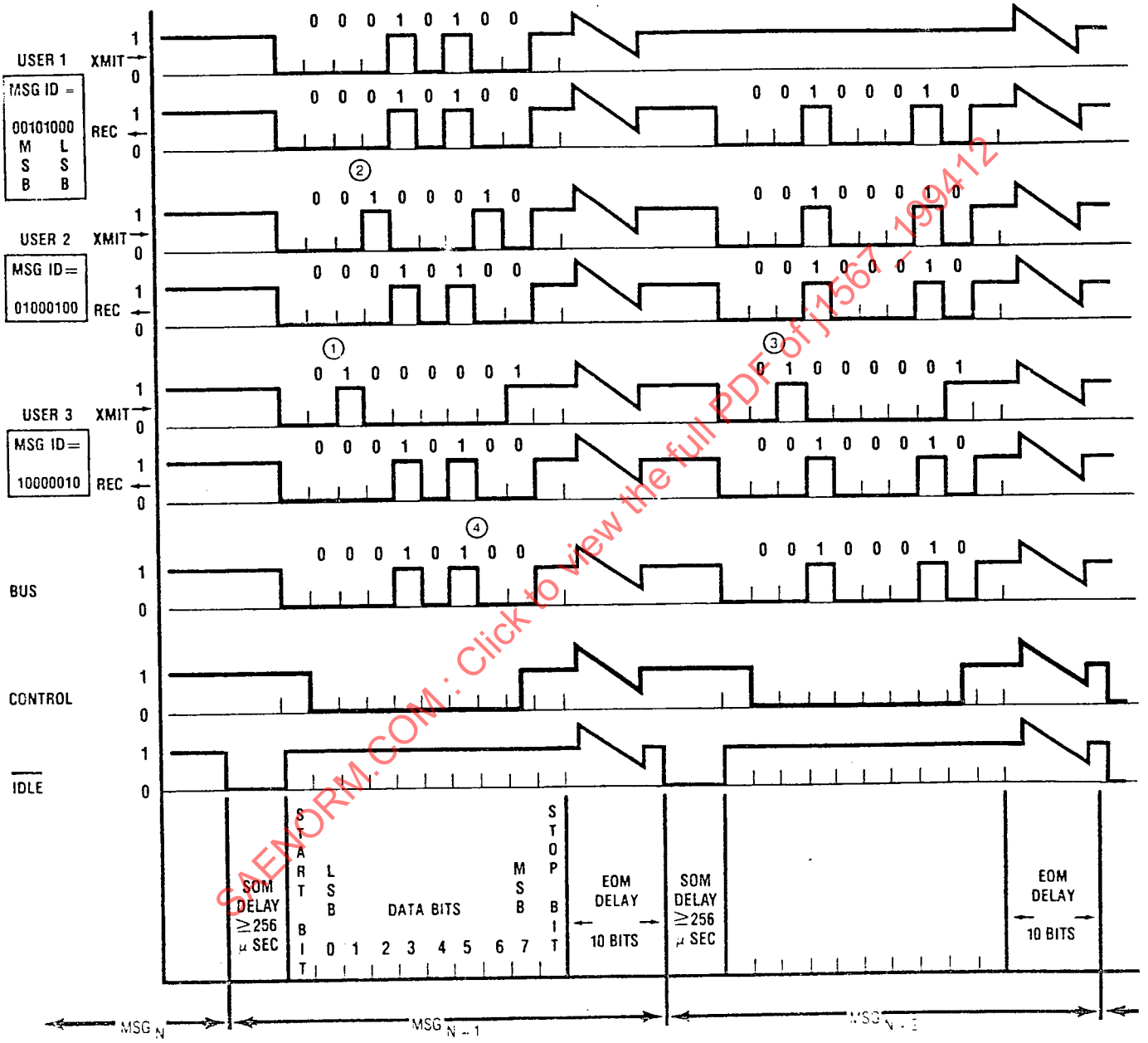
The bus shall depend on external resistor and other components for bias and termination. Clamping diodes may be added to provide a high level of transient protection.

In addition to the transmission of data, the differential transceiver receives data at its bus "+" and bus "-" terminals. The received data is translated back into the standard digital logic levels by a differential amplifier. The microcomputer always receives the actual transmitted data and in this manner can test for loss of arbitration.

3.2 Collision Detector: Data collision detection occurs because the transceiver output is reflected back into its input. The data collision detector samples the transmitted signal and the received signal. The timing of this sampling is determined by the timing circuit. When the collision detector determines that a logic one bit is being transmitted, but a logic zero bit is being received, the collision detector blocks the transmitted signal (See waveform shown in Fig. 2). In this way the data collision detector will permit only the interface with the highest priority to continue transmitting. The collision detector action of blocking transmission is also reset by detection of bus idle (10 consecutive idle bits).

3.3 Arbitration Detector:

The arbitration detector works in conjunction with the timing and synchronizing circuit to arbitrate between the start of data to be transmitted with the start of a received message from the bus. The arbitration detector blocks a transmission that could corrupt a message that is already in progress. Also, it allows the device that starts transmission first, after a bus idle, to pass its data through the interface and out onto the bus. In all other nontransmitting devices the arbitration detector blocks transmission of data until the detection of a bus idle condition. When more than one device wants to transmit at about the same time, (greater than 1/4 bit time) the arbitration detector will allow transmission on a first-come, first-serve basis. If data transmission from more than one device on the bus is attempted in near synchronism, (less than or equal 1/4 bit time) the arbitration detector will allow the transmission. However, the data collision detector will permit only the one with the highest priority to continue transmission. The arbitration detector is also reset by the detection of a bus idle.




NOTE: ①, ③ User 3 Loses Arbitration ② User 2 Loses Arbitration

④ A 1 Bit from User 1 is not overridden by 0 Bits from Users 2 or 3 because Users 2 and 3 have both been blocked from bus access due to earlier collisions

FIG. 2 - ARBITRATION OF THREE SINGLE BYTE MESSAGES

3.4 Digital Filter: A low pass digital filter is placed between the transceiver and the received data output. This circuit functions to filter out any received EMI from the desired digital data signal before being processed by the other circuits of the interface.

3.5 Bus Idle Detector: The function of the bus idle detector circuit is to detect when the bus is idle (not active) or busy (active) and then feed this information back to the microcomputer. It accomplishes idle detection by sensing a received stop bit followed by ten (10) bits of continuous idle or logic ones. Normally an active or busy period follows an idle period. The sensing of an active or busy bus is accomplished by detecting a start bit. During unusual conditions such as node start up, any transition from a logic one to a logic zero that is maintained for a period longer 1/4 bit time sets the active or busy flag.

3.6 Timing and Synchronizing Circuit: The timing and synchronizer circuit uses an external clock and establishes the synchronizing and baud rate timing signal. This generator circuit first synchronizes on the negative edge of a start bit and then generates a timing signal at the center (1/2 bit time) of each bit. This timing signal is used by the arbitration and collision detector circuit for sampling received data. This timing signal is also used by the idle detection circuit to determine an idle bus.

4. UNIVERSAL INTERFACE IC: A block diagram of the integrated circuit developed that supports all three modes: (1) SCI, (2) SPI, and (3) Buffered SPI is shown in Fig. 3. Notice the additions to the diagram shown in Fig. 1. An Over Range Latch has been added between the differential receiver and the Digital Filter to hold the received data at the logic level it was before the out of dynamic range signal occurred. Also the timing and Synchronizing circuit is shown in greater detail.

4.1 SCI Mode of Operation: When the Start Bit Detector senses a valid start bit, it causes the Word Counter to synchronize itself to the timing of the received data word (8 bit byte). A valid start bit is determined by the arbitration detector to be negative signal that remains negative for 1/4 bit time after an idle period or a valid stop bit. The word counter is used to generate the 1/4 bit time (32 μ s for 7812.5 baud) pulse for the arbitration detector and 1/2 bit time (64 μ s for 7812.5 baud) pulse for the data collision detector. The word counter triggers the framing error detector at the stop bit time. If the stop bit is not detected, the Idle counter is extended by the framing error circuit until 10 consecutive idle bit periods are received.

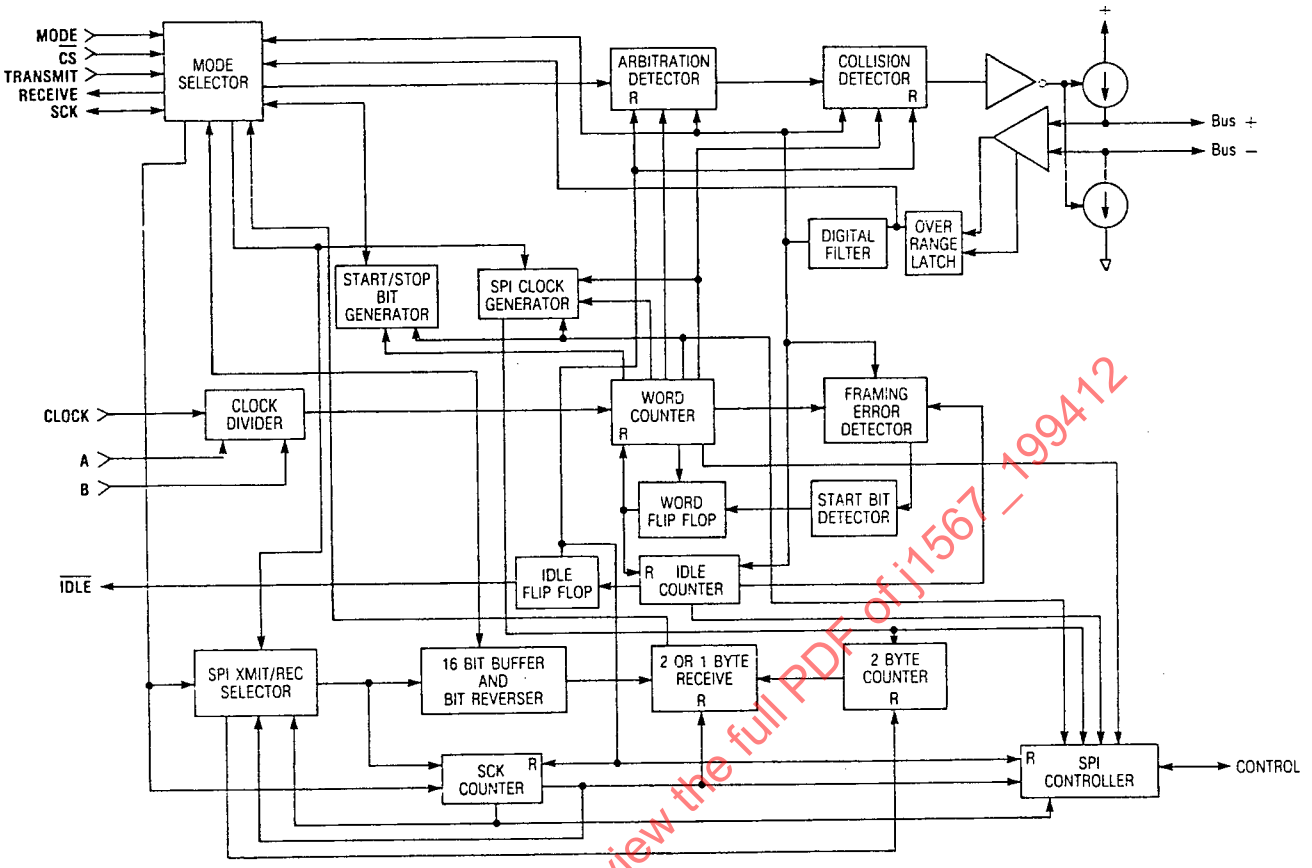



FIG. 3 - BLOCK DIAGRAM OF UNIVERSAL BUS INTERFACE IC



4.1 (Continued):

When a microcomputer connected to the bus is ready for transmission, it should use the following procedure. First it reads the $\overline{\text{IDLE}}$ line and waits until it goes to a logic "0", which indicates the bus is idle. Then the microcomputer tries to transmit the 8 bit ID word associated with the data to be transmitted. If it started transmitting first or has the highest priority ID, the collision detector circuit will permit transmission. The microcomputer must confirm transmission by reading the received ID word and comparing it with the ID it wanted to transmit. If there is confirmation that the same ID was transmitted, then data can be transmitted. If not, then the microcomputer needs to compare the received ID with the list of required ID's in order to determine if it needs to process the received message. Please take note, the microcomputer need not cut off the data transmission because the collision detector will accomplish that.

It is important to realize that data collision may result due to outside interference. The microcomputer that is transmitting data can compare the transmitted data with the received data for this type of data collision. Appropriate action should then be taken by the microcomputer.

4.1 (Continued):

The function of the idle counter and idle flip flop circuit is to detect when the bus is in the idle (not active) condition. It does this by sensing a received stop bit and then delays a short idle period of ten (10) bit times. The idle detector output is then set to a logic "0". The idle detector output is set to a logic "1" by receiving a start bit.

4.2 SPI Mode of Operation: In the SPI mode the word counter generates the timing signals to drive the SPI clock generator and the start/stop bit generator. The microcomputer operates as an SPI slave. The interface IC operates as an SPI master. When the microcomputer needs to transmit a message, it loads the ID of that message into its SPI data register. The microcomputer should first monitor the IDLE pin to determine when the bus goes idle. The microcomputer then pulses the CONTROL pin to a logic "0". This sets the SPI controller circuit to schedule a transmission at the appropriate delay time after idle. The interface IC will: (1) then generate a start bit, (2) supply the microcomputer with 8 SCK shift pulses in synchronization with the start pulse and, (3) then generate a stop bit. If the transmitted ID collides with a higher priority ID transmitted by another module, the interface IC will stop transmitting immediately and transfer to the microcomputer the data it receives from the bus. If no collision is detected, the module has won arbitration and the microcomputer is free to complete data transmission. The microcomputer must monitor the CONTROL pin of the bus IC. The SPI controller circuit will set its output to a logic "1" when it completes the transmission of each ID or data word.

The SPI data buffer can then be loaded with the next word and the CONTROL pin must be momentarily pulled low to transmit that data. The SPI controller circuit works in conjunction with the start/stop bit generator and produces synchronized shift clock pulses for both receiving and transmitting of data. However, the start/stop bit generator does not output shift pulses for the start or stop bits.

The data on the bus must be transmitted least significant bit (LSB) first and most significant bit (MSB) last (ref. Fig. 2). Generally, SPI transfers are done MSB first and LSB last. The data transmitted and received may have to be reversed within the microcomputer for the SPI mode of operation.

4.3 Buffered SPI Mode of Operation: In the buffered SPI mode, a 16 bit buffer and bit reverser circuit is provided for both receiving and transmitting data. This buffer is accessed by the microcomputer through the mode selector circuit. The order of the data bits is reversed as the data is transferred between the microcomputer and the buffer. The microcomputer SPI generally transfers data MSB first and LSB last while the data on the bus is transferred LSB first and MSB last. This insures that the order of the data on the bus while operating in the SCI, SPI, and buffered SPI modes is the same (LSB first). The 2 or 1 byte receive circuit works in conjunction with the buffer system and the 2 or 1 bytes counter to allow the reception of odd numbers of bytes in a message. The control flip flop in the SPI xmit/rec selector circuit is used to determine whether the buffer is connected to the microcomputer input or to the bus transmit circuitry. When powered up, the buffer is connected to the microcomputer.

4.3 (Continued):

The microcomputer is the SPI master and the IC is an SPI slave. In a system where there are multiple SPI slave units, the bus IC can be selected by using the $\overline{CS}$ pin. When the microcomputer wants to transmit, it selects the bus interface IC by outputting a "0" to the $\overline{CS}$ pin and then monitors the CONTROL pin. Before the microcomputer can transmit, it must wait for the completion of all bus activity. The SPI controller circuit monitors the idle flip flop and the SCK counter circuit to generate the control signal at the proper time. When the CONTROL pin goes to a logic "1" signifying that the buffer register is full of received data and can be read by the microcomputer, the microcomputer then supplies 16 shift pulses (two 8 bit SPI transfer cycles) and reads the received data at the same time it loads the SPI buffer with the ID and data it wants to transmit. The microcomputer then momentarily pulls the CONTROL pin to a logic "0" and the data will be transmitted at the proper time. If the microcomputer just needs to access the received data it just does the two 8 bit SPI transfers and does not pulse the CONTROL pin low. The interface IC contains the circuitry to hold the received data in the buffer register and ignore receiving new data until after the old data has been read. The maintenance of the received data in the buffer register until it is accessed by the microcomputer insures that the transmitted data can be tested even in the case of a slow data transfer. It is important to test all attempts at bus access to verify when correct transmission has been completed and when it will be necessary to retransmit the message.

5. AN ITEM FOR FUTURE DEVELOPMENT: The normal method used to detect message data error is a firmware checksum scheme. Based on the following statements, a good alternative to this practice may be the detection of Out of Common Mode Range.

- All electrical interference that would corrupt data in a vehicle network is single ended, for example, it effects both Bus+ and Bus- equally.

- Maintaining a sufficiently high ratio between single ended interference to differential interference using a properly terminated twisted pair transmission media may be feasible.

- Therefore, if the above paragraphs are attainable then all electrical interference that would corrupt data would first generate an out-of-range condition. Also, out-of-range integrity checking of the data is a much simpler than the firmware generated method and does not have the bandwidth reduction.

6. CONCLUSIONS: The quality of a serial communications bus protocol should be judged by the following factors:

6.1 Low Cost: Cost is contained by achieving a very simple hardware interface IC design and the handling of the bus protocol in microcomputer ROM firmware.

6.2 Universally Microcomputer Friendly: This goal is achieved by being capable of directly connecting to the microcomputer's SCI or SPI serial communications I/O ports.

6.3 Low Generation of EMI while Maintaining High Bus Message Throughout: Byte sync NRZ data modulation was chosen because it has the advantage of being the most efficient modulation method, that is, has fewer transitions for data byte. Empirical testing shows that this translates to low generation of EMI.

6.4 Low Data Error Rate: This goal was achieved by using differential current mode NRZ data modulation. NRZ data demodulation has been shown to achieve superior data recovery performance in a noisy environment and can be accomplished using simple data sampling techniques.

6.5 Reasonably Fault Tolerant: The Interface IC is protected to survive bus shorts to battery and ground. Microcomputer watchdog circuits can be implemented to protect against bit streaming failures. These measures are not sufficient to guarantee data communications. The known methods of overcoming or partially overcoming many of these failure modes are either too complex or generates too much EMI to be practical. The best protection is for the system designer to realize this when partitioning the vehicle system. The few failure conditions that could cause a safety problem can be handled using other partitioning options or with limp in modes. Another option, of course, is to separately hard wire these few safety related situations.

## APPENDIX A - TECHNICAL SUMMARY
General Bus Interface IC Functionality

### Bus Message Format

<msg id 1> <msg id 2> ... <data byte 1> <data byte 2> ... <data byte n>

The msg id and data bytes are individual bytes that are asychronously transmitted with 1 start bit, 8 data bits and 1 stop bit using a typical NRZ format with a '1' level signal for a bus idle and stop bit value and a '0' level signal for a start bit value.

**Interbyte Delay:**
The maximum delay between bytes is <10 bit times. The minimum delay between bytes is 0 bit times.

**IDLE:**
A continuous period 10 bit times long of a '1' level signal after the detection of a proper stop bit is interpreted as an IDLE condition. IDLE is used to synchronize all stations to the condition that: 1. the last message on the bus is complete, 2. the next byte on the bus will be the first msg id byte of a message, 3. any stations that have messages to transmit can begin the transmission 2 bit times later.

**Use of Msg Id byte(s):**
The msg id byte(s) are used to determine the winner in message collision situations. The msg id can be 1 or more bytes.

**Use of Data Bytes:**
The data bytes can be anything (that is, CRC byte, Checksum, etc.)

**Minimum Message Size:**
The minimum message size is 1 byte (that is, 1 msg id byte).

**Maximum Message Size:**
The maximum message size is unlimited (Defined by host microcomputer ROM firmware)

### CONTROL PROCEDURE

The control procedure determines when a station may attempt to transmit on the bus and who wins the use of the bus if multiple stations attempt to transmit on the bus at the same time.

A collision occurs on the bus if more than 1 station attempts to transmit a msg id byte within 1/4 bit time of each other. Stations that attempt to transmit a msg id byte later than 1/4 bit time after the first station began to transmit have already lost bus arbitration.

If multiple stations transmit the same msg id byte at the same time, then these stations will continue to arbitrate with each other for the use of the bus using the next msg id byte just like the first msg id byte (that is, the winner is the station that transmits the next msg id byte first or whose next msg id byte has the highest priority value).

The priority of msg id bytes is determined by the rule that a '0' bit will win out over a '1' bit. As soon as a given station loses during bit by bit comparison, it is blocked from transmitting any more bits onto the bus. Bit by bit comparison includes all bits of a byte (that is, start bit, data bits and stop bit) but normally is determined by the value of the data bits. Data bit transmission begins with the LSB and continues on to the MSB.

All stations receive the data that is actually transmitted on the bus. Thus, a station that is attempting to begin transmitting a message will receive the msg id of whatever message won the use of the bus.

**Abort:**
Stations that receive a msg id different from the one that they were attempting to transmit have collided with the transmission of a higher priority message from another station and have lost. Stations that detect this should stop attempting the transmission of their message and should consider switching over to receiving the winning message if it is something that they want to receive.

**Defer:**
A station that loses at message arbitration should immediately stop further transmission of message bytes but do not have to. The bus interface IC will automatically cut off its transmitter from effecting the bus as soon as that station has lost to another station and will not allow the losing station to transmit onto the bus until IDLE has occurred again. EMI that causes message corruption has the same effect and the IC will automatically cut off transmission.

**Retransmission:**
A station that loses at bus arbitration can attempt to retransmit its message as soon as IDLE occurs again.

**Number of Stations:**
Any number of stations may be connected to that same network.

**Transmission of Start Time:**
The transmission of the first msg id byte shall be such that at least a nominal 2 bit times delay exists between IDLE and the beginning of the transmission of the start bit.

**Bus Driver Interface:**
The bus driver drives the bus through a 'wired OR' type of connection where any single station transmitting a '0' level signal forces the bus to a '0' level even if multiple stations attempt to simultaneously transmit a '1' level signal.

**Hardware Collision Detection and Bus Arbitration:**
The hardware collision detection function is reset at IDLE. Bus arbitration begins when one or more stations begin to transmit a message after IDLE. Collision begins when more than one station attempts to transmit a message on the bus at the same time. Bus arbitration and any collisions end: 1. at 1/4 bit time after the beginning of a start bit (that is, the first '0' level signal that lasts for at least 1/4 bit time and occurs after a full stop bit) if the '0' level signal lasts for longer than 1/4 bit time and if only one station is attempting to transmit or 2. when the last remaining station(s) lose a bit by bit priority comparison with the resulting winning station due to the 'wired OR' nature of the hardware bus drivers, if multiple stations began transmitting a start bit within the 1/4 bit time of each other.