SAE International™

SURFACE VEHICLE INFORMATION REPORT

SAE J2106

CAN-CELLED JUL2002

Issued 1991-04
Cancelled 2002-07

Superseding J2106 OCT1996

Token Slot Network for Automotive Control

Foreword—Data Links in Vehicle Control

Introducing the Token Slot Network Protocol (TSN)—This document describes a network protocol called the Token Slot Network that has been developed by General Motors for use in real-time vehicle control applications. This protocol offers a number of technical advantages such as controllable maximum latency, efficiency, and reliability. In addition, it is cost competitive with other protocols which lack these advantages.

Token Slot Protocol Background—The need exists for data links to be used in real-time control of vehicle systems. These systems require message delivery capabilities beyond those of the SAE Class B protocol that is now known as the SAE J1850 protocol. After examining the communications requirements of the real-time control domain, no existing protocol was found that would completely meet those needs. Therefore, a new protocol was developed with the basic features of token passing schemes, but which included a time slot scheme for performing the token pass.

The following sections will first discuss the application needs that drove the protocol development, then describe the protocol itself, and finally present methods for time analyses of systems which use the protocol.

Token Slot Network Application Characteristics—The Token Slot Network is intended for use in distributed, real-time control applications in land-based vehicles. The network nodes are typically intelligent (i.e., microprocessor based) components that cooperate in order to perform some control task. They may be functionally independent nodes, such as an engine control module and a transmission control module, that each have relatively autonomous functions that need to be coordinated for better total vehicle behavior. Alternatively, there may be a hierarchical control structure where some nodes implement a local low-level control algorithm, but are subservient to a higher level control running in another node. In either case, rapid and predictable communication is needed.

Real-time control communication systems must provide:
— Rapid control loop feedback via data link
— Regular data updates for periodic or event driven data
— Minimal, predictably bounded message delays (latencies)
— System specified and limited transmit privilege hold time

TO PLACE A DOCUMENT ORDER:
Tel: 877-606-7323 (inside USA and Canada)
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: custsvc@sae.org
SAE WEB ADDRESS: http://www.sae.org

Data transfer efficiency requirements:
—Low message overhead
—No physical limits to higher data rates

Fault tolerance requirements:

—Rapid fault detection and data error rejection
—Quick initialization and recovery from bus errors
—Masterless—unaffected by node drop-off

Token Slot Network Design Philosophy—The need for bounded latency characteristics in applications with both bursty and periodic traffic require us to reject all protocols that use a contention mechanism for media access control (such as SAE J1850 and CAN). While such protocols can be fashioned to provide statistical “guarantees” that message latency will not exceed required limits, there can always be cases where traffic loading will cause latencies to grow beyond any predefined limit. This is unacceptable for applications controlling real-time physical processes.

For this reason, it was decided to pursue protocols with more deterministic media access methods. A token passing scheme was chosen because it offers determinable and bounded latencies and it can be implemented efficiently with low data overhead.

In token passing protocols, the single node in possession of the token is a temporary bus master and can broadcast messages or initiate message transmission to other nodes as it chooses. When ths node either finishes sending its messages or approaches its token hold (transmit privilege) time limit, it passes the token to another node which then becomes the master and the process repeats. The network can be configured so as to guarantee orderly access to the data link by all nodes.

One problem area in such schemes has to do with the token passing mechanism itself. Typically, the current token owner has a “next” node in a logical sequence order. When the token is passed, it is to be explicitly passed to the “next” node. If that node has no network traffic to send, it must still accept the token, and then pass it on to its successor. This procedure can lead to substantial inefficiencies especially if the “next” node has dropped off the bus. In this case, the current node must follow an exception procedure for determining to which node it is to pass the token. This usually involves polling to determine which nodes are still active, building an “active node” list, and broadcasting the list to all nodes. This total procedure is often complicated and is always time-consuming.

Similarly, following system power-up initialization or if noise causes the corruption of the token message, the token may become lost. This also requires a time-consuming reinitialization process for reorganization of the virtual node sequence ring formed by each node’s idea of the logical “next” node in the sequence.

To solve this problem, it was decided to use a different token passing method based on rotating time slots. This allows transparent skipping of nodes which have no pending traffic or which are inactive. This protocol also provides for quick system initialization and for error recovery. The details of this method will be described in the following sections.

TABLE OF CONTENTS

1. ***Scope*—**The Token Slot Data Link is intended to provide periodic, broadcast communications (communication that must occur on a regular, predetermined basis) within a vehicle system.

The Token Slot protocol achieves this by implementing a masterless, deterministic, non-contention Token Slot sequence which is designed to offer a transmit token to all devices (or nodes) without requiring that they respond. After acquiring the token, messages may be sent and verified using a variety of built-in techniques. The token passing slot sequence is then reinitiated by the current token holder.

1.1 **Purpose—**This SAE Information Report describes the Token Slot Data Link Communication Protocol. It is intended to cover the attributes of the network structure, network management, bus access procedures, message framing, bit encoding, and message delimiting required to communicate on the Token Slot Data Link. These aspects relate to those items that must be standardized excluding the physical implementation. The physical details of the network such as media, line driver/receivers, and semiconductor implementation are discussed generally but are not specified by this document.

1.2 Background—In recent years it has become apparent that future electronic applications within vehicles will need to communicate critical control information among distributed controllers. This control information to be transmitted is typically periodic in nature and must arrive on a regular, predetermined basis. In many cases, it is changing at rapid rates and only sampled information is provided at appropriate intervals (rather than updating the information only when it changes). This must be done at speeds that are sufficiently fast to meet the requirements of the interactive control applications. The Token Slot Network Data Link is designed to provide this capability.

1.3 Design Objectives—The Token Slot Network was defined with the following goals in mind:

a. It must meet the needs for on-vehicle, rapid, periodic communication between computer assemblies.
b. Target applications for this link will be higher speed control applications (typically above 1 Mbit/s).
c. Message maximum latency time delays must be minimal and bounded to assure stability in closed loop control applications. See Appendix A for methods of calculating and predicting message latencies.
d. There must be an open data rate growth path to accommodate future media technologies and demands for increased data traffic.
e. The architecture must be an open system to allow for the addition and deletion of nodes (devices) both in new designs and during dynamic operation. Note that open architecture does not mean that devices can be added to the data link without regard to their affect on message latency times.

It is intended that this link will be used primarily in vehicle applications requiring feedback and control data to be shared among various distributed devices. For example, the link could be used between distributed engine, transmission, anti-lock braking, and power steering controllers. Flexibility has been built into the protocol to also include event driven and block memory transfer data types. This allows vehicles to be equipped with a single data link to cover all communications to avoid gateways and reduce costs, when system and reliability conditions permit.

2. References

2.1 Applicable Publications—The following publications form a part of this specification to the extent specified herein. The latest issue of SAE publications shall apply.

2.1.1 SAE PUBLICATIONS—Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

SAE J1850—Road Vehicles—Serial Data Communication
SAE J2056/1—Class C Application Requirements
SAE J2056/2—Survey of Known Protocols
SAE J2056/3—Selection of Transmission Media

2.1.2 OTHER PUBLICATIONS

82527 Serial Communications Controller Architectural Overview—INTEL Corporation
"Data Transmission Over the Telephone Network: Series V Recommendations" Section V41, The Orange Book VIII.1, International Telecommunications Union, Geneva (1977)

3. Technical Requirements

3.1 Bus Access Procedure—This section describes the token slot bus access method of providing transmit privileges to devices communicating on the Token Slot Data Link. It is the primary difference between Token Slot and contention protocols (e.g., SAE J1850, CAN, etc.).

The intent of this bus access protocol is to guarantee periodic opportunities for message transmission by each node on the bus. It is to ensure that the bus remains operational when devices are dynamically added or deleted and it must provide for quick recovery from error conditions.

The token passing bus protocol is open, peer oriented, and multimaster. It is non-contention and uses a time slot token passing technique. The bus access procedure is performed on a distributed, masterless basis, with all devices behaving identically to maintain an orderly rotation of transmit privileges.

The token slot method described here has several features which help achieve the goals of a deterministic data link. The principal advantage of this method of bus access is that the worst-case delay of information exchange due to the data link can be determined and bounded. Requirements for line driver/receiver circuitry are less stringent than would be required with a contention based system since only one transmitter is normally on at a time and since the protocol is not data rate limited by signal propagation delay. The protocol has provisions for rapid initialization of the link and for handling cases in which the number of devices on the bus changes during operation (due to power moding sequences, failure or power loss of individual devices, or intermittent software operation of devices on the bus). Fast error recovery is a primary feature of this approach.

3.1.1 TOKEN SLOT OPERATION—The token slot method offers transmit privileges to all devices on the bus in an orderly manner. After a node has completed sending its message traffic, a sequenced scan of short, equal time intervals (slots) offer bus transmit privileges to the node slot owners as follows: A token pass message (see Figure 1) instructs all nodes to begin the token slot timing mode. Each slot interval following a token pass message has a sequential Slot Number (SLN). Control is offered to the slot owning node devices by time progressing through successively higher Slot Numbers. Each node is assigned one or more specific time slots (Slot Numbers) and will activate its transmitter to send a message during its slot only if it is operational and has message traffic to send. Otherwise the token slot interval is allowed to pass. When the transmitter is activated, all other nodes recognize that the token has been taken and they enter the receive mode. See Appendix B for a summary of token pass messages. See Appendices C and D for a comparison with contention based protocols.

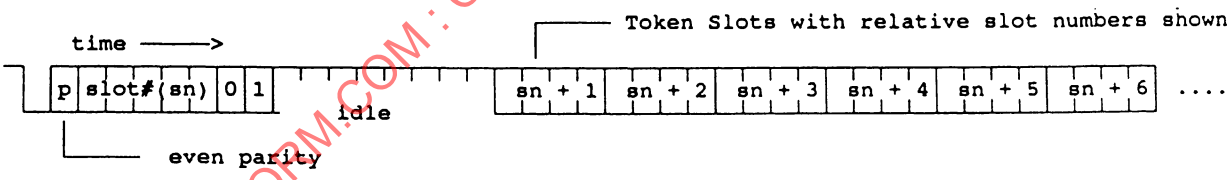

FIGURE 1—TOKEN PASS MESSAGE FORMAT

The token slot method differs from time-division multiplexing, which provides a “time slot” wide enough for an entire message, to each device on the bus. If the device is not present on the bus or has no message to transmit, the bus must remain inactive during that entire time slot.

The Token Slot protocol does not send a directly addressed token pass message to hand off control of the bus to a specific device which may or may not be active. This avoids the requirement that all devices must maintain a prioritized list of all currently active nodes to which the token must be passed.

Figure 2 depicts the bus waveform of a token message sent by the device which owns SLN 5, followed by the owner of time slot 1 (SLN1) taking the token, sending two messages, and then initiating its own token pass message.

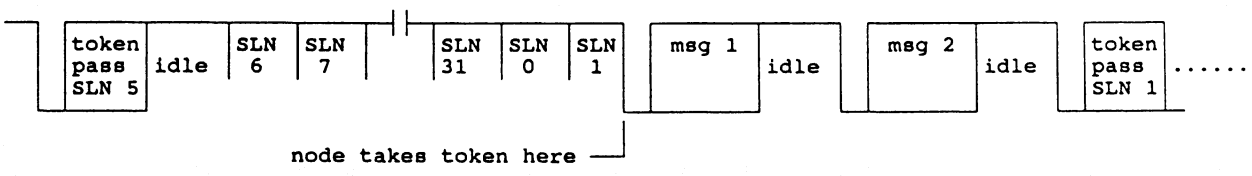

FIGURE 2—TYPICAL TOKEN PASS SEQUENCE

In this example, the devices holding slot numbers 6, 7, ... through 31 and 0 were either not present or did not have a data message ready for transmission. Thus, the bus remained in the idle state during those time slots.

As shown, when the maximum slot number is reached, the sequence wraps around to slot 0 and continues until the slot is picked up or until the original token passer sees its slot, at which time a new token pass message is generated and the token slot cycle begins again.

Once a device takes the token, it proceeds to send its message traffic. Token hold times are individually assigned to each node by the system designer and are strictly limited to assure a system maximum message latency limit. Individual message transmit priorities are determined by each node's application and are not restricted by the communications data link.

3.1.2 TOTAL NUMBER OF SLOTS AVAILABLE—All applications of the Token Slot bus have thirty-two time slots available. Thus, there can be a maximum of thirty-two transmitting type devices present on the bus in any particular application.

3.1.3 NUMBER OF SLOTS PER NODE—A particular device may have zero (for a "receive only" device), one, two, or more Slot Numbers, depending on how often that device needs to transmit messages compared to the other devices on the bus. A device which owns multiple Slot Numbers must take care to include the Slot Number for the time slot it actually took during that token possession in any token messages it sends. The number of time slots that a particular device may occupy is to be defined by the system application designer who is responsible for message latency assurance.

3.1.4 INSTRUMENTATION SLOT NUMBER—SLN 0 is ideally reserved for use by instrumentation in all applications to provide an external access by standard test equipment. It may be prudent to require that the vehicle be placed in a special instrumentation interactive mode before SLN 0 be allowed to take the token.

3.1.5 TIME SLOT SEQUENCE DELAY CALCULATION—All devices use the Slot Number contained in a received token message to determine how long they must wait until they should take control of the bus if no other node intervenes. When a token message is received, each device, including the one that sent the token, calculates its delay using Equation 1:

$$\text{Transmit Delay Time (TDT)} = \{[(A-B)-1]\} \text{ modulo32} * T \qquad \text{(Eq. 1)}$$

where:

A = Slot Number of calculating device
B = Slot Number in last received token message
T = Time Slot Width, expressed as an integral number of bit times (bt)
Modulo32 means that:

If [(A–B)–1] < 0
then TDT { = [A–B)–1] + 32} * T

Else TDT = [(A–B)–1] * T
Therefore: 0 < TDT < 32*T

Any device holding multiple Slot Numbers will need to calculate a transmit delay time for each of its Slot Numbers, then take the token on the earliest TDT.

If another device begins a transmission in an earlier slot before this transmit delay time has elapsed, the bus access calculation must be restarted by waiting for a new token message to be received. However, if the bus remains inactive for this calculated transmit delay time following a token message, it means that this device's token slot interval has arrived. If the node has messages to send, it may begin a transmission at this point and thus take control of the token and the bus.

The transmit delay time begins immediately following the 8 bit time (bt) idle line delimiting the token message. This means that the transmit delay time counter should begin counting at the beginning of the 9th bit time following a token message. Figure 3 illustrates this situation.

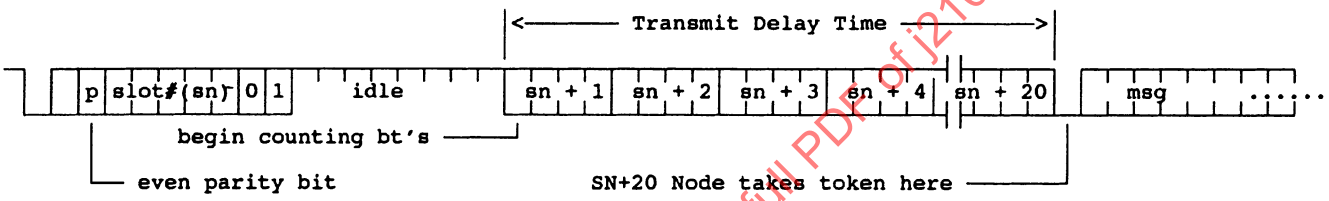

FIGURE 3—TOKEN PASS MESSAGE FORMAT

3.1.6 TIME SLOT WIDTH—The time slot width is determined by a timing analysis of the system bus propagation delay and is shown as an integral number of bit times (bt). The time slot must be long enough to allow a device to start a transmission and for all other devices to recognize that event. Its width must also take into account the accumulated time base drifts that occur as each device is counting and waiting for its time slot to occur. A typical slot width might be 1 to 5 bit times when operating at 1 Mb/s for propagation delay characteristics on the order of 5 ns/m for either a properly terminated electrical or a plastic fiber optic medium.

3.1.7 TOKEN POSSESSION GUIDELINES

3.1.7.1 *Decision to Transmit*—When the bus remains inactive for a device's transmit delay time following a token message, all devices except the one that sent the last token message have two choices. These devices have the option of taking possession of the token and beginning transmission of a message or of remaining silent. A device may send one or more messages once it takes token possession, and it then relinquishes control of the bus by sending a token pass as its last message.

3.1.7.2 *Maintaining Synchronization*—The device that sent the last token has the duty to maintain a minimum level of activity on the bus. In the event that no other device initiates a transmission during the thirty-one slots following a token, the device that sent the last token must take the token and send either a data message if one is ready or only a token message if no data message is ready. The purpose of this is to maintain synchronization of the time slots during periods of low bus utilization. Accumulated time base differences among all the devices causes blurring of the time slot boundaries as the time since the last token message increases. This explains the need for periodic synchronization. Figure 4 illustrates the case where no device takes the token and the original token passer must re-take it.

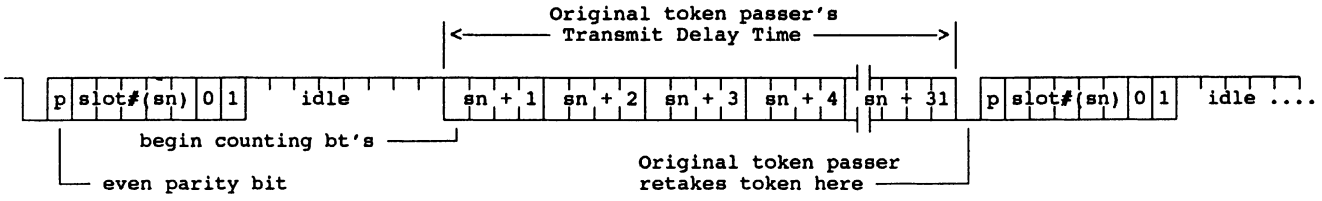

FIGURE 4—TOKEN PASS MESSAGE FORMAT

3.1.7.3 *Maximum Token Possession Time*—Each device on the bus is assigned a maximum time that it may hold the token on any individual possession. This maximum token possession time helps to bound the token rotation time for the system and is used to avoid continuous transmission caused by a device failure. A device must not begin transmission of a particular data message unless completion of the message can be accomplished without exceeding its maximum token possession time. If that message is too long to be sent during the present token possession, the device must send a token message, then cease transmission. The time that the token has been held and time required to send the next message may be calculated by the message lengths + bit insertions + message acknowledges + idle lines between messages. There are various methods of calculating and/or estimating the maximum token possession time. See the section on Maximum Token Possession Time Rationale for further discussion. See Appendix A for Methods of calculating and predicting message latencies.

The system designer may assign different maximum token possession times to different devices in a particular application.

3.1.8 TOKEN SLOT BUS EXCEPTION CONDITIONS—This section describes exception conditions that may occur on the bus and how these conditions should be handled.

3.1.8.1 *Initialization*—When a device requires initialization following a power up or internal reset condition, it monitors the bus for a token message or for a Bus Time Out to occur. The Bus Time Out (BTO) occurs when the bus is idle for thirty-two time slots (see 3.1.8.5). If a token message is received, the device calculates its transmit delay time and begins normal operation.

3.1.8.2 *Bus Jam Signal*—The Bus Jam is defined as the receipt of six or more consecutive logic zeros by a receiving device. A transmitter communicates error situations to all devices on the bus with the Bus Jam. It is intended to cause all devices on the bus to cancel any current message and begin recovery procedures.

The bus jam must be a dominant signal level and must last at least 6 bit times which is longer than any normal data signal condition. This will prevent a legal bit stuffing signal from causing a bus jam. The bus jam signal must be started before a post-message 8 bit time idle line is allowed to occur on the bus. This ensures that the jam signal causes all devices to discard the corrupted message. If a complete idle line were allowed to occur, then devices receiving the subsequent bus jam would not necessarily reject the corrupted message.

3.1.8.3 *Response to Bus Jam*—When a device recognizes a bus jam condition, it rejects the message in progress (if any) as invalid and calculated a transmit delay time as if a token message containing a Slot Number of thirty-one was just received. This results in slots for devices beginning with SLN 0, 1, 2, ... etc., to take the token following a bus jam. Once the jam condition ends and an 8 bit time (bt) idle line is detected, all devices begin waiting for their time slot to occur. If a device finds that the bus remains idle for its transmit delay time following any jam, it is to transmit either a data message or a token message if it has no data

messages to send. This ensures that normal bus activity resumes properly under light bus load conditions. Figure 5 shows the bus activity following a bus jam.

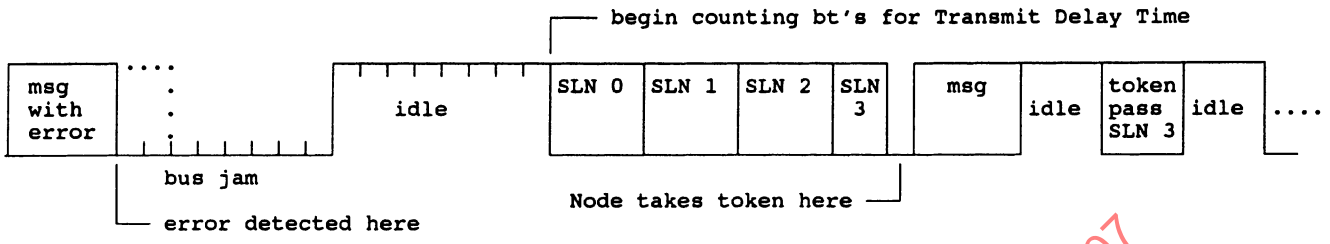

FIGURE 5—BUS ACTIVITY FOLLOWING BUS JAM

In this example, no device on the bus holds SLNs 0, 1, or 2. The device holding SLN 3 transmits either a data message or a token message if it has no data messages ready for transmission.

3.1.8.4 *Collisions*—A collision is the result of two or more devices attempting to transmit at the same time. Collisions may occur during operation of the bus due to corrupted token messages, noise during the time slots following a token message, or due to other noise-induced conditions. Each transmitting device must monitor the data it receives during message transmission and compare it to the data it transmitted. This comparison may be done on a bit or byte level. If a discrepancy is detected between transmitted and received data, the transmit device is to jam the bus for 8 bit times (bt) and relinquish control of the bus (see 3.1.8.3).

3.1.8.5 *Bus Time Out*—The Bus Time Out (BTO) occurs when the bus is idle for thirty-two time slots. This condition may occur due to failure of a transmitting device to send a token message or loss of the token message due to noise. The reaction to a BTO is similar to the response to a message error. When a BTO occurs, all devices assume that a token message containing a Slot Number of thirty-one was just received and calculate the appropriate transmit delay time. If any device finds that the bus remains idle for this transmit delay time, it transmits either a data message or a token message if it has no data messages to transmit. Figure 6 depicts this situation.

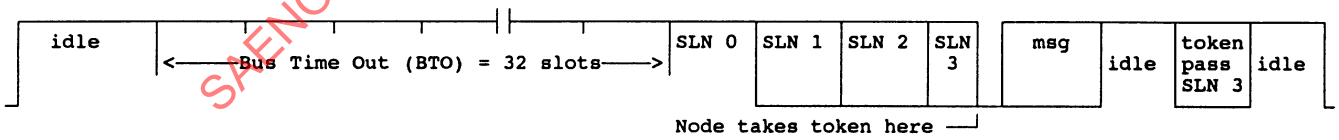

FIGURE 6—BUS ACTIVITY FOLLOWING A BUS TIME OUT

In this example, no device on the bus holds SLNs 0, 1, or 2. The device holding SLN 3 begins either a data message if it has one to send, or a token message if it does not, in the appropriate time slot.

3.1.8.6 *BTO During Token Possession*—It may be possible for a Bus Time Out to occur while a particular device is holding the token. For example, a device may transmit one message properly and then experience a long delay in preparing the next message for transmission during that token possession. Once the device senses a BTO, it must immediately relinquish possession of the bus, and begin waiting for its time slot to occur just as if it had not had token possession prior to the BTO.

3.1.8.7 *BTO at Initialization*—When a device requires initialization following a power up or internal reset condition, it monitors the bus for a token message or for a BTO to occur. If a BTO is detected, it calculates its transmission delay time as if a token message with a Slot Number of thirty-one had been received and continues as if the BTO had occurred during normal operation.

3.1.8.8 *Transmit Error Counter*—Each device maintains a transmit error counter to monitor the frequency of errors that occur while that device is transmitting. The transmit error counter should be initialized to zero and incremented by some system assigned algorithm consistent with maintaining bus message latency boundaries in the face of a number of bus failures.

3.2 Bit Encoding and Message Delimiting—This section describes the mechanism for transmitting information on the Token Slot Data Link. It includes the bit encoding and the rules governing synchronization and delineation of message frames.

3.2.1 NRZ5 BIT ENCODING RULES—All messages on the Token Slot data link are encoded as Non-Return-to-Zero with opposite level bit insertion after the fifth consecutive bit remains at a constant one or zero logic level. This technique will be called NRZ5 encoding. NRZ5 is applied regardless of transmission media used.

NRZ5 is similar to standard NRZ in that the logic level on the bus is a logic one for the entire bit period to transmit a binary one and is a logic zero for the entire bit period to transmit a binary zero. In addition, synchronization information is added to the bit stream at the transmitter by inserting a bit of the opposite state into the bit stream whenever five consecutive bits of the same state are to be sent. For example, when the transmitter detects that a string of five zeros is to be transmitted it inserts a one into the bit stream, then continues the data transmission, again watching for a string of five consecutive zeros or ones including any inserted bits. Upon reception of this string of five zeros, the receiver checks that the next bit to arrive is a one (the one inserted by the transmitter), and deletes this one from the bit stream. If a sixth zero is received, the receiver treats these six zeros as a bus jam signal and takes the proper recovery steps. A corresponding situation occurs for five ones in a row, except that six received logic ones in a row indicate the start of an idle line, not a bus jam. An example of this procedure is shown in Figure 7.

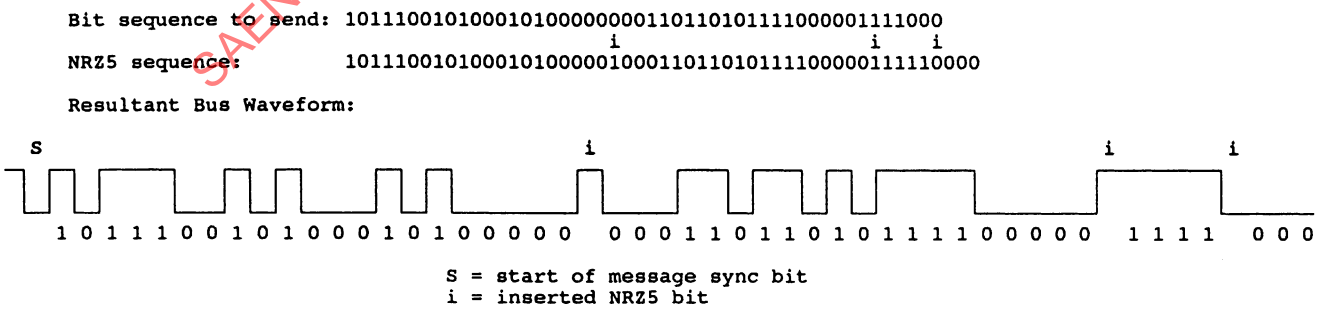

FIGURE 7—NRZ5 EXAMPLE

If a message should happen to end in five zeros or five ones, the transmitter must still perform the one or zero insertion. The receiver will check for the presence of the inserted bit and delete it as required.

3.2.2 ASYNCHRONOUS OPERATION—Serial data transfers are done asynchronously, meaning that a single data stream with imbedded pre-message synchronization is the only signal among the devices on the bus. This avoids sending a separate clock and data signal which would increase the cost of the transmission medium and bus transceivers. Receivers decode the incoming data by synchronizing on the logic one-to-zero transitions in the data stream. During normal data transmission these transitions will never be more than 10 bit times apart due to zero and one insertion in NRZ5 encoding. When used with closely matched clock sources in all devices, this synchronization information is sufficient for proper data decoding.

3.3 Message Framing—All information transmitted on the link must be consistently framed for correct interpretation. For the Token Slot Data Link, four message types are identified, distinguished by two control bits in the first byte of each message. The four message types are broadcast data messages, requiring an acknowledge, single byte acknowledge messages, and single byte token messages

Data bytes are transmitted least significant bit first to conform to traditional series data link standards. See Figure 8.

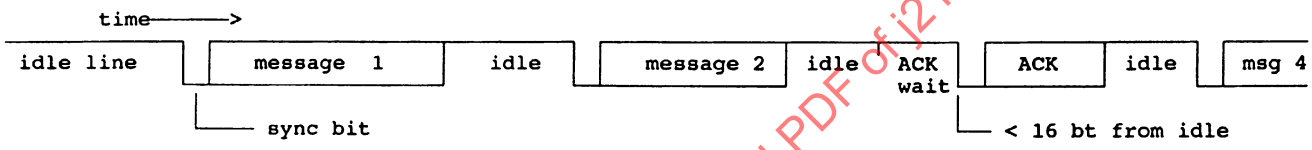

FIGURE 8—GENERAL MESSAGE FRAMING

The two most significant bits in the first byte of every message identify the type of message being transmitted. These bits are labelled Control Bits 0 and 1 (CB0 and CB1) and are defined in Table 1:

TABLE 1—DEFINITIONS—CONTROL BITS 0 AND 1 (CB0 AND CB1)

CB0	CB1	Message Type
0	0	Broadcast data message
1	0	Data message with acknowledge request
0	1	Token message
1	1	Acknowledge message

Note that CB1 distinguishes data message frames from the two single byte message frames, while CB0 indicates that this message is either requesting or supplying an acknowledge. See Figure 9.

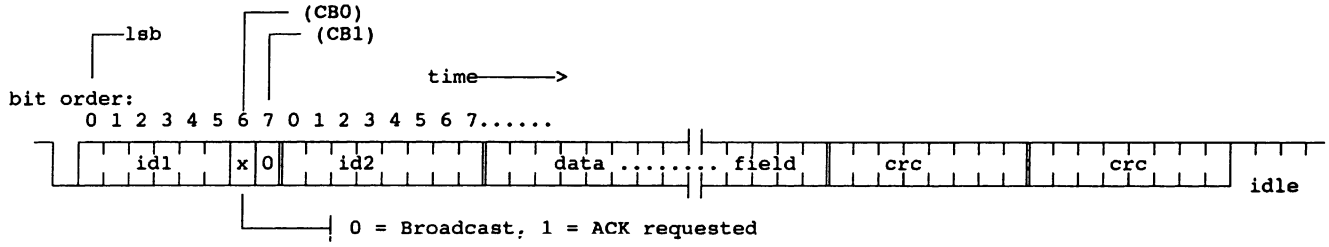

FIGURE 9—DATA MESSAGE FORMAT

3.3.1 DATA MESSAGE FRAMES—The data message is used for the actual transfer of information on the data link. There are two types of data messages, one which requires no acknowledgment, and another which requires that an acknowledge message be returned by a single receiving device.

The data message frame consists of a two byte message control (2 bits) and identification (14 bits) field, an optional data field, and a two byte CRC frame check sequence (FCS) field. These fields are described in the following sections. The shortest possible data message frame is four bytes long, consisting of only the ID and FCS fields. Message frames shorter than this minimum length are to be rejected by the receiving device.

3.3.1.1 Data Message Identification Field—The identification Field identifies the message using a 14 bit message ID number which is contained in the first two bytes of a data message. The format of the ID field for a data message is in Figure 10.

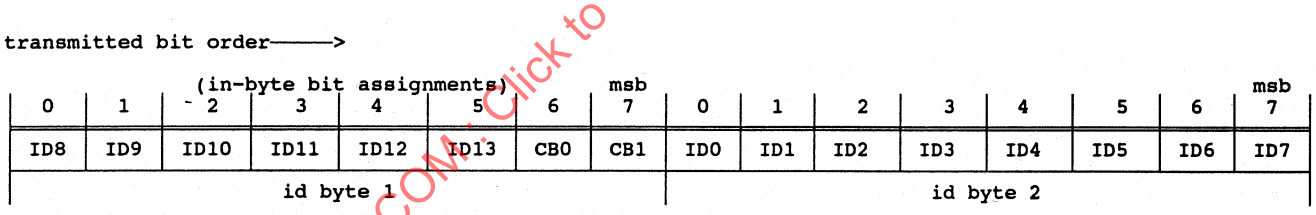

FIGURE 10—DATA MESSAGE ID FIELD

In the data message, CB1 is set to zero to indicate a data message. CB0 is set to zero for a broadcast data message and set to one for a data message which requires an acknowledgment. ID13 through ID0 form the rest of the message ID.

3.3.1.2 Data Field—The optional data field contains any information beyond the message ID that needs to be transferred by the message. The information transmitted in the data field is sent in byte format, meaning that the data field is always an integral number of bytes. The data field may be from 0 to 256 bytes in length. The contents, scaling, order of appearance, and processing of any data in the data field is user defined and is a function of the message ID number.

3.3.1.3 Frame Check Sequence (CRC) Field—The 16 bit CRC conforms to the CCITT standard and detects all single bit errors, all parity errors, and all burst errors less than 17 bits long. For burst errors longer than 16 bits, the CRC misses 0.0015% of all errors.

The Frame Check Sequence (FCS) field consists of two bytes of CRC code. The FCS field is used by receiving devices to detect errors in incoming data messages. Generation and decoding of this field is as follows:

The CRC is calculated by performing a binary division of the entire contents of a message (before NRZ5 bit insertion) by a generating polynomial G(X) (see Equation 2). The following CCITT-CRC (see Section 2, "Data Transmission Over the Telephone Network: Series V Recommendations") generating polynomial is used where X = 2 (for modulo 2 operations):

$$G(X) = X^{16} + X^{12} + X^{5} + 1 \quad \text{(Eq. 2)}$$

All bits in the ID and data fields are included in the CRC calculation. The remainder obtained by the division of the data by the generating polynomial is appended to the end of the message as the FCS, with the following additions:

a. First, the dividend initially contains all ones, and the CRC is calculated on each data message bit as it is transmitted. This allows the CRC to detect any missed or inserted ones at the beginning of a message.
b. Second, the remainder from the binary division is inverted (one's complement) by the transmitter before transmission of the FCS. This detects errors which make the data field appear to be rotated by a number of bits.
c. Third, the FCS contains the CRC remainder, transmitted most significant bit first. Note that NRZ5 bit insertion is performed on the FCS after the calculation and prior to transmission.

Receivers perform a similar computation on incoming data after the NRZ5 bit has been deleted, but include the Frame Check Sequence (CRC) field within the binary division. This should result in a constant remainder of $FOB8 after the ID, data, and FCS fields have been processed for a valid message. If the receiver does not find this remainder, then the FCS check has failed and the received message should be discarded.

3.3.2 ACKNOWLEDGE FRAME—The acknowledge frame has the fixed format as in Figure 11.

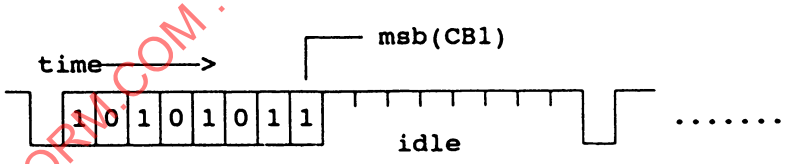

FIGURE 11—ACK MESSAGE FORMAT (FIXED)

The control bits CB1 and CB0 are both set to ones to indicate an acknowledge message. Thus, the valid acknowledge frame is always $D5.

The system designer must assign only one node the responsibility for returning the acknowledge response to a given message.

3.3.3 TOKEN PASS MESSAGE FRAME—The token frame is used to pass control of the link from one device to the next. This frame indicates the Slot Number of the device that currently has transmitting privileges on the bus. This frame is transmitted by the device currently in control of the link after it has completed sending its current messages. The token message is always a single byte in length, and the format of this field is as in Figure 12.

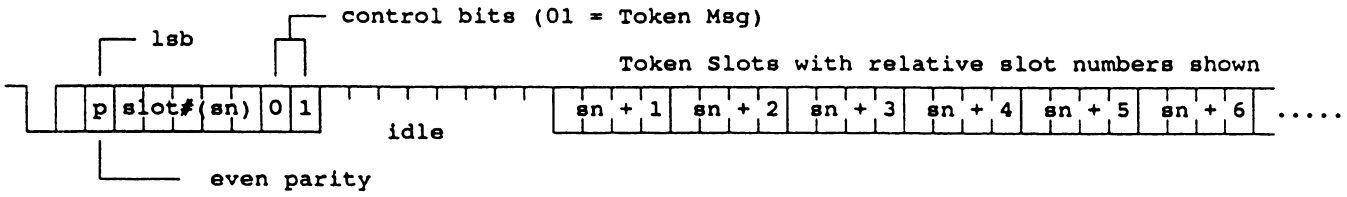

FIGURE 12—TOKEN PASS MESSAGE FORMAT

3.3.3.1 *Token Pass Message Control Bits*—The CB0 and CB1 bits are set to zero and one, respectively, to indicate that this frame is a token message.

3.3.3.2 *Token Pass Slot Number Bits*—The five slot number bits contain the slot number that the device used to gain access to the bus during this token possession. All devices use the slot number to determine their transmit delay time following the idle line at the end of the token message.

3.3.3.3 *Token Pass Parity Bit*—The parity bit is set to the proper logic state to provide even parity in the token message. A token frame which has odd parity will be ignored if received.

3.4 Message Delineation—Message delineation is defined as the proper recognition of message frames within the serial data stream. The NRZ5 bit insertion and deletion is used to provide the basis for message frame recognition. Bit insertion prevents the bus from remaining in one logic state for more than 5 bit times. Thus, the occurrence of six or more consecutive logic ones or zeros is the shortest method of bounding a message. Therefore, the minimum message delimitation time is specified as 8 bits of logic one (or idle line) to provide a reasonable signal propagation delay margin. Longer periods of bus inactivity may occur in several situations: the end-of-message (EOM) idle line message delimiter, the start-of-message (SOM) synchronization bit, the acknowledge time out (ATO), the bus time out (BTO), and the bus jam.

3.4.1 IDLE LINE (NORMAL MESSAGE DELIMITATION)—The idle line is the primary method of message delimiting. Only messages that are delimited by idle lines are to be accepted as valid messages. An idle line is defined as the receipt of eight consecutive logic ones by a receiver following the end of a received data byte. Even if a message data field ends in one or more logic ones (after bit insertion), these are not counted as part of the 8 bit idle line.

An idle line is to be recognized only after the bus has remained at a logic one state for eight entire bit times past the boundary of the last data byte in a message. This is important when passing the token. If a device were to declare an idle line when it actually samples an eighth one, it could start transmitting immediately, possibly before all devices had actually sampled that 8th bit. Delaying idle line recognition beyond six until eight full bit times have elapsed eliminates that possibility, since all devices would have had time to properly sample the 8th bit.

3.4.1.1 *Idle Line (Inter-Message Gap)*—Messages from the same device during a token possession time should be transmitted with a minimum of idle time between them. The shortest idle time is the 8 bits necessary to properly delimit the messages, and this should be achieved whenever practical. However, exception cases may exist which cause this inter-message gap to exceed the 8 bit time minimum. As a goal, consecutive messages should not have more than 16 bit times of idle between them to preserve the data throughput on the bus. This figure is subject to change if the constraints it places on transmitting devices is too severe. The maximum message gap must always be less than a Bus Time Out in any case. The occurrence of a BTO will force a device to relinquish token possession and respond to the BTO error.

3.4.2 SYNCHRONIZATION BIT—The synchronization bit is a single binary zero which is transmitted for one bit time immediately preceding any message on the bus. This bit is similar to a start bit used in standard UARTs. Before the start of any message, the bus will be at the logic one (idle) state for at least an idle line time (8 bt) and the logic zero sync bit both informs all receivers that a message is starting and provides the first synchronizing edge for NRZ5 bit decoding purposes.

Figures 13 and 14 depict the idle line, sync bit, and start of message.

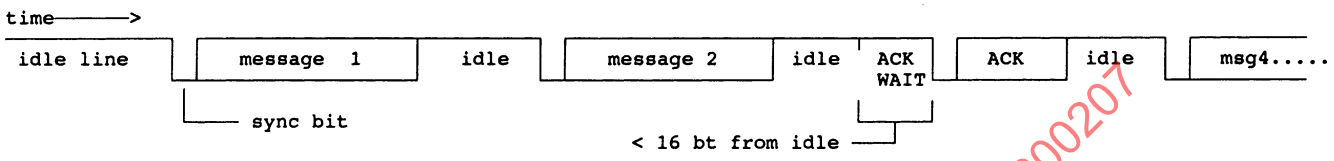

FIGURE 13—SYNCHRONIZATION AND GENERAL MESSAGE FRAMING

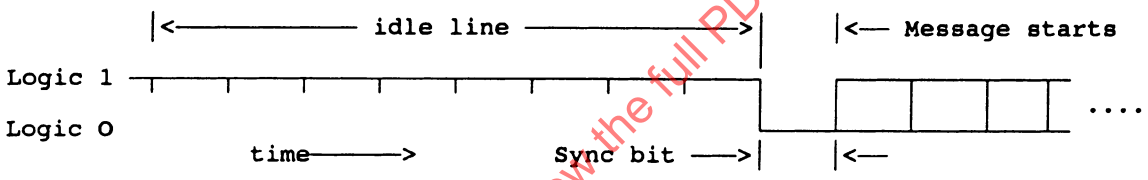

FIGURE 14—SYNCHRONIZATION BIT STARTING A MESSAGE

3.4.3 ACKNOWLEDGE TIME OUT—Messages which have requested an acknowledge must allow an Acknowledge Time Out (ATO) period of 20 bit times beyond the request message's idle line delimiter before declaring a no-response and beginning to transmit the next message.

The Acknowledge Time Out (ATO) is the maximum number of bit times that a device is to wait for an expected acknowledge message to begin. The ATO is defined as the time from the end of the idle line delimiting the message which requires an acknowledge to the start of the acknowledge frame being returned by the destination device. The required ATO time must be less than a Bus Time Out (see 3.4.4) and should be as short as practical for minimal impact on bus throughput. The ATO time is therefore defined to be 20 bit times in duration. This allows adequate time for the device transmitting the acknowledge to decide if the incoming message requires an acknowledge and time to begin its transmission. To provide a timing margin for the ATO, a device should not begin the acknowledge if it is unable to do so within 16 bit times from the idle line at the end of the previous message. See Figure 15.

Another important, though less critical, time associated with the acknowledge function is the time for the device holding the token to resume transmission. This could be either a retransmission of that message if an ATO occurred or another data message or a token message if a valid acknowledge was received. The transmission start time should be as short as practical, but in general should not exceed 20 bit times following either an ATO or a valid acknowledge message. This requirement is important to keep the bus throughput burden of the acknowledge function to a definable minimum.

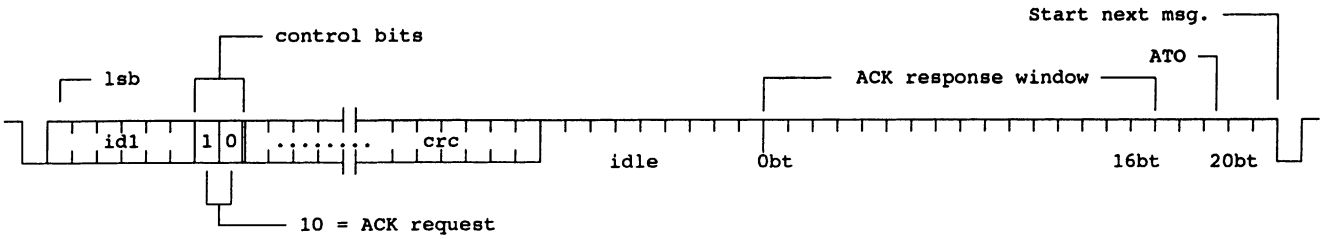

FIGURE 15—ACKNOWLEDGE MESSAGE TIMING

3.4.4 BUS TIME OUT—A Bus Time Out (BTO) is declared when the bus has remained at the logic one state for 32 time slots. The BTO does not include the idle line at the end of a message. During normal operation, the requirement that one device continue to send tokens in the absence of other bus traffic prevents idle periods of more than 31 time slots. See Figure 16.

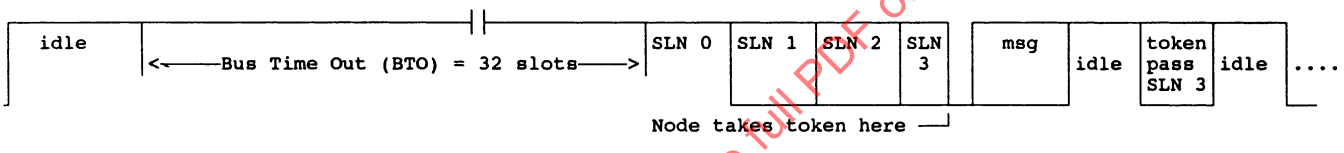

FIGURE 16—BUS TIME OUT

In general, a device should start counting time slots whenever it finds that an idle line has occurred. Following any bus activity except for a valid token message, the device should begin waiting for the 32 time slots which define a BTO. When a valid token message is received, the device should instead calculate and wait for its transmit delay time and decide whether to begin transmission as described in the “Token Slot Operation” section.

3.5 Network Structure—This section describes the physical structure of the Token Slot Data Link. This includes the connection of devices to the bus, transmission media, and bit rate. Each application may have different objectives which affect the final configuration of the system. For instance, a requirement for easy addition and subtraction of devices on the bus will affect the method of coupling to the transmission medium. The guidelines given here represent the basic characteristics which all applications of the Token Slot bus are likely to meet.

3.5.1 BUS TOPOLOGY—The Token Slot Data Link is functionally configured as a bus, such that all devices transmit and receive over a single common path, and all devices receive information transmitted on the link simultaneously (within the bounds imposed by finite propagation delays in the wiring, receive circuitry, etc.).

A discussion of key considerations associated with the physical bus configuration is given in the following paragraph. Consideration of these factors during the physical bus design will produce a proper determination of the time width of the token slot. This is necessary to assure that all nodes can detect that the token has been taken. The Token Slot Protocol is specially adept at accommodating the bus propagation delay in that the token slot periods can be adjusted to varying time widths to accommodate a given vehicle platform design without affecting either the ability of the protocol to operate at the highest practical media data rate or the ability to determine the maximum message latency time.

Figure 17 describes the propagation delay considerations for the Token Slot Network. It is most important that the slot width time is long enough to include two worst-case maximum propagation delays (2*Tpd max = a round trip), an adequate receive node sample window time, and a transmitter turn-on delay. The two nodes in question are presumed to be the furthest time distance apart (Tpd max). Note that many of these considerations also apply to the contention based protocols–and on each bit in the contention field.

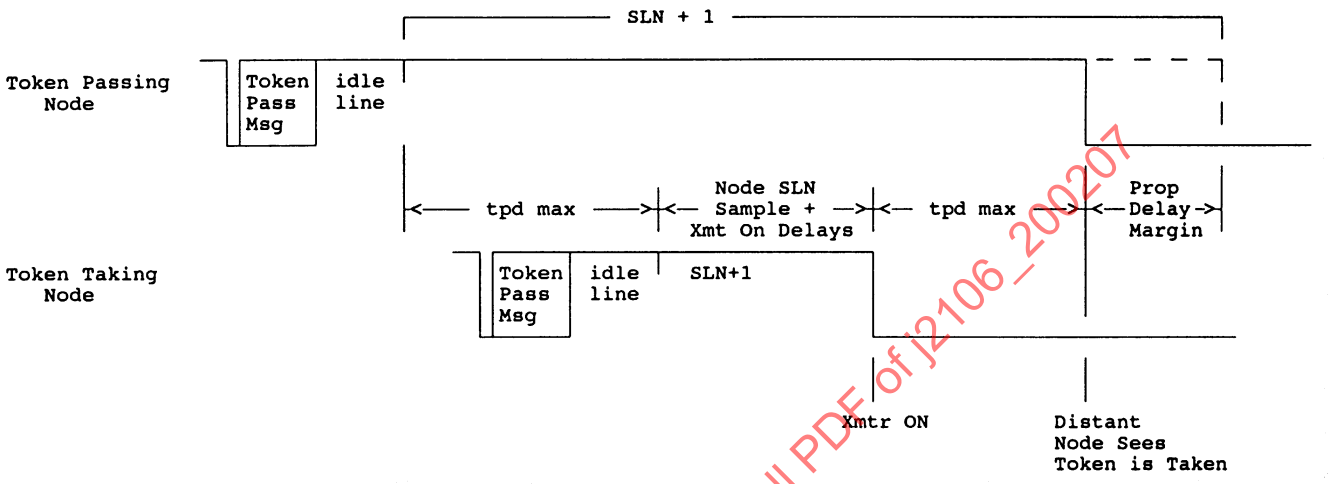

FIGURE 17—PROPAGATION DELAY CONSIDERATION

The physical layer configuration is not specified here, since transmission medium, physical location of devices on the bus, and failure mode concerns dictate these requirements at the system level. However, the method of coupling devices to the bus must ensure that individual device loss of power, transmitter turn-off, or other device failures do not render the bus inoperative.

3.5.1.1 *Bus Length*—The maximum length of the bus will be application determined by the characteristics of the bus media components. Consideration of such factors as transmitter output energy, receiver sensitivity and dynamic range, bus losses, terminations, propagation delays, environmental effects on components, etc., determines the greatest acceptable length of wire or optical fiber between the bus transceivers of any two devices on the bus. The connection to any off-vehicle instrumentation must be included in this length budget.

3.5.2 PHYSICAL MEDIUM—The Token Slot Data Link offers implementation flexibility to accommodate various transmission media at a variety of bit rates. Each medium supports a maximum bit rate which can be achieved with acceptable electromagnetic interference (EMI) and compatibility (EMC) performance in on-vehicle applications. Thus the maximum bit rate is limited by the range allowed by the transmission media.

3.5.2.1 *Transmission Media*—The following media are candidates for use in systems using the Token Slot bus: single random lay wire, twisted pairs, twisted and shielded pairs, and fiber optics. The actual choice depends on the required bit rate and economic trade-off studies for each application.

3.5.2.2 *Data Bit Rates*—Data bit rate is only limited by the available technology of bus media, receivers, transmitters, etc. Future improvements in components and media will allow a virtually unlimited future data rate growth path beyond the 1 or 2 Mbits/s capability.

3.5.2.3 *Bit Rate Tolerance*—Each device must maintain the selected bit rate within a tolerance maximum consistent with reliable bit sampling and token acceptance detection. The Bus Time Out condition is the most vulnerable to this parameter with 32 token slot times to run without a clock synchronization signal. Factors which influence the bit rate tolerance include variations due to part tolerances, operating temperature range, and component aging.

3.5.2.4 *Logic Zero Dominance*—The transmission medium and the bus transceivers must be designed such that a logic zero being transmitted by one device will override logic ones being transmitted by one or more other devices. Use of a dominant logic state allows both better collision detection and the ability to force a jam signal onto the bus.

3.6 Node Device Implementation Considerations—This section describes the duties of devices which transmit and receive information on the Token Slot Data Link. The actual partitioning of these duties between the microcomputer software, protocol support hardware, and the line driver/receiver hardware is not specified here, although the indicated timing constraints will affect that partitioning.

3.6.1 MESSAGE TRANSMISSION—The required capabilities and duties of a device transmitting a message include acquisition of transmitting privileges, message selection, actual transmission of the message, message validation, and proper recovery from error conditions. These requirements are detailed in the following sections.

3.6.1.1 *Required Resources for Transmission*—A device must have both transmit and receive capabilities and a valid slot number to transmit messages on the Token Slot bus. Receive capabilities are needed so that it can acquire the token and monitor its own transmissions. The slot number is needed so that the proper time slot can be used to take the token before beginning transmission. Devices without these resources are not to be allowed to acquire token possession.

3.6.1.2 *Transfer of Transmitting Privileges*—A device obtains transmitting privileges by determining the proper transmit delay time following activity on the bus and then beginning a transmission if the bus has remained at the logic one state for that time. The transmit delay time is calculated using the Slot Number in a received token message or using a Slot Number of 31 after a bus jam or bus time out has occurred. The device must begin its transmission (that is, drive the bus to the dominant state) within a short time after its transmit delay time elapses. This time includes the delays in the serial hardware and the bus transceiver and is compensated by adjusting the system token slot width. In normal operation, the last message sent by a device will be a token message containing that device's Slot Number.

3.6.1.3 *Message Selection*—When a device obtains transmitting privileges, it may transmit one or more messages on the bus. Each device is responsible for deciding which message(s) to transmit.

3.6.1.4 *Message Update Rate*—In general, it is undesirable for a device to send every possible message during every token possession. Such an approach would use excessive bus bandwidth and tend to limit the growth potential of the network. Each message should be transmitted only at the update rate required by other devices in the system or when an event causes a change in status. Background status summary messages may be scheduled and sent periodically to update receiving nodes which may have missed the initial event driven message. Decisions to send or withhold messages based on current bus "business" and maximum hold time limits may be implemented by a node if authorized by the system designer.

3.6.1.5 *Maximum Token Possession Time*—Each device on the bus should be assigned a maximum time that it may hold the token on any individual possession. This maximum token possession time helps to bound the token rotation time for the system. A device must look ahead and may not begin transmission of a data message which would cause the maximum token possession time to be exceeded. The token message must be sent as the final message even if this maximum time has been reached or exceeded. While this function is defined as a time period, the actual implementation may count the number of bit times or number of bytes that the device has token possession.

The actual time required to send a message is not known before beginning the message. This is due to bit insertions which may occur in the message and the limited but unknown length of time which may be required for an acknowledge to the message. These factors force the token possession time to be somewhat hard to exactly control. The minimum implementation of the maximum token possession time monitor is for the device to add the number of bytes already transmitted to the length of the next message, and compare this sum to a fixed maximum. If the sum is less than the maximum, the message is transmitted. This method does not account for bit insertions, inter-message gaps, and acknowledges which may have occurred during the current token possession. A better method is for the device to use a timer to accurately assess the current token possession time and then estimate whether the next message will fit into the maximum token possession time by its message length. In this way, the token possession time will be affected only by bit insertions in the next message and time required for an acknowledge.

See Appendix A for methods of calculating and predicting message latencies.

3.6.1.6 *Transmission Monitoring*—Each device must monitor its transmitted data stream for evidence of corrupted data on the bus. This monitoring by the transmitting device is done to detect bus collisions, noise corrupted transmissions, and transmitter underruns. Since a transmitter knows what data is being transmitted, it can compare the data it receives to the data it transmitted with a high probability of detecting these errors. This comparison can be done at either a bit level or a byte level.

3.6.1.7 *Transmit Message Validation*—A message is not to be considered to have been successfully transmitted until the following conditions are met:

a. All transmitted bytes have been received correctly by the device's receiver. This gives a high degree of confidence that no interruptions in the data stream occurred, that a collision situation did not exist during transmission, and to a lesser degree that noise did not corrupt the transmission.
b. An idle line condition was sensed by the receiver both preceding and following the message. This ensures that the message was properly delimited.
c. If the message was a data message (CT1 bit = zero in ID field) and contained a request for acknowledgment (CB0 bit set in the ID field), a valid acknowledge frame must have been received following the idle line at the end of that message. This acknowledge frame must have been started before an Acknowledge Time Out (ATO) condition is reached.

3.6.1.8 *Error Handling and Recovery*—Proper response to error conditions is essential for consistent, reliable operation of the Token Slot bus. The following error recovery procedures must be observed by all devices.

3.6.1.9 *Transmitted and Received Data Differences*—When a discrepancy between transmitted data and received data is detected, the current message must be terminated and a jam signal sent on the bus. The jam signal is intended to communicate the error condition to all devices by forcing bit encoding errors to occur in their received data. This jam signal must be started before an idle line occurs which would delimit this message to the receiving devices. Otherwise, the jam would not cause other devices to reject the corrupted message. For NRZ5 encoding, the transmitted jam signal should be 8 bits in duration to ensure that all receivers sense at least six consecutive zeros.

The jam signal is also sent if a transmitter detects that the idle line following the current message is corrupted. Once the message has been delimited by eight consecutive logic ones, a continuation of logic ones may extend the idle line. The transmitting device is to continue to monitor the bus for an idle condition (logic ones received) during this inter-message gap and issue a bus jam if a logic zero is received during this time.

3.6.1.10 Transmitter Underrun—A transmit underrun typically occurs when the transmitter hardware runs out of data to transmit before a message is completed, causing a gap in the bit stream. If a transmitter should detect a transmit underrun condition, it must immediately issue the bus jam signal and cease transmission. As discussed in the previous section, this jam must begin before a valid idle line is completed following the message data.

3.6.1.11 Acknowledge Frame Errors—When a device sends a message requesting an acknowledge, an acknowledge frame must be started before an Acknowledge Time Out (ATO) is declared. Any activity on the bus before the idle condition is achieved will cause a transmission error in the transmitting device, since an early start of the acknowledge frame will cause the acknowledge to be ignored. If, following the idle line, some bus activity occurs before an ATO elapses, but a valid acknowledge frame is not received (acknowledge byte plus idle line delimiter), the transmitting device again assumes that the message was not received by the proper device. Finally, if an ATO elapses following the idle line at the end of the message, the transmitting device re-assumes transmitting privileges and continues transmission with either a repeat of that message, a new data message, or a token message. This subsequent transmission must begin before a BTO can be detected by any receiver on the bus. The only condition that will cause a transmitting device to validate a message containing an acknowledge request is that a valid acknowledge frame is started within one ATO of the idle line delimiter.

If the transmitting device waiting for the acknowledge detects bus activity before an ATO occurs but does not receive a valid acknowledge message, it should wait for an idle line to occur then attempt one retry of the message or send a token message if a retry has already been made.

3.6.1.12 Transmitter Error Counter—Each device should maintain an error counter for transmitted messages to assist in fault diagnosis and isolation.

3.6.2 RECEIVE MESSAGE VALIDATION—This section defines the required actions of devices receiving messages on the Token Slot bus. The duties of a receiving device include message reception, message selection, message validation, acknowledgment (when applicable), and error handling and recovery.

3.6.2.1 Resources for Message Reception—To receive messages from the Token Slot bus, a device needs only receive capability. Transmit capability is not required, but if available, it may be used to send an acknowledge message when appropriate. A valid slot number is not required to either receive messages or send an acknowledge message.

3.6.2.2 Receive Message Selection—Because of the bus configuration of the Token Slot bus, all messages are actually receivable by all devices since any device can listen to any transmission on the bus. However, certain restrictions on message reception exist and are described in this section.

3.6.2.3 Token Message Reception—All devices that require transmitting privileges on the bus must accept and interpret all token messages before assuming the transmit mode.

3.6.2.4 Broadcast Data Messages—Messages identified as broadcast data messages can be received and used by any device on the bus that knows the format and processing rules of that message. This implies that changes to this message must be communicated to all users of the bus.

3.6.2.5 Data Messages with Acknowledge Request—These messages are used when only one device on the bus has been designated to receive and acknowledge the message. The receiving device is required to transmit an acknowledge frame in response to that message if it meets the requirements in 3.6.2.11.

All devices except the one requesting an acknowledge to a message should disregard acknowledge messages which may occur on the bus. This restriction does not apply to diagnostic equipment.

3.6.2.6 *Acknowledge Frame*—If a received data message containing a request for acknowledge (CB0 bit in ID = 1) passes the preceding validation tests, an acknowledge frame must be transmitted. This acknowledge frame is to begin within 16 bit times from the end of the idle line delimiting the message to allow the original data message sender time to detect the response before it declares an ATO. This 16 bit time limit may need to be adjusted to accommodate physical bus propagation properties. If the message did not pass all the validation tests, or if the receiving device does not have transmit capability, no transmission shall take place in response to the acknowledge request. A device that has transmit capability must return the acknowledge message when required if it is functional and capable of acting on the message.

Any errors encountered during transmission of the acknowledge frame are to be handled as if the device were transmitting a data message or token message. It is appropriate for the receiver to issue a bus jam when these errors are detected.

3.6.2.7 *Message Delimiting*—All messages are to be delimited with an 8 bit time idle condition that will be seen in all receivers as an idle line. Receiving devices use idle lines to define the boundaries of individual messages on the bus.

3.6.2.8 *Bit Encoding Checks*—During reception of a message, each receiving device is to monitor the data stream for bit encoding errors. An NRZ5 bit encoding error is the occurrence of six consecutive bit times of the same logic state. Six or more consecutive logic zeros is interpreted as a bus jam, while six or more consecutive ones may be seen as either an encoding error or the beginning of an idle line.

3.6.2.9 *Error Check Field*—All data messages contain a 16 bit CCITT CRC at the end of the message. All receiving devices should use this CRC to check the validity of the received data.

3.6.2.10 *Minimum Message Length*—The minimum length for data messages is 4 bytes not including the idle line delimiter (2 ID bytes + 2 CRC bytes, if no data bytes). A receiver is to reject any data message that is not at least 4 bytes long. The length of token messages and acknowledge messages is always 1 byte. Note that Control bit CB1 = 0 in the ID field distinguishes data messages from tokens and acknowledges.

3.6.2.11 *Receive Message Validation Summary*—A message will not be considered to have been successfully received until the following conditions are met:

a. The message was properly delimited by idle lines (this is a basic requirement for defining an individual message).
b. No bit encoding errors were detected between the idle lines.
c. The CRC or parity check revealed no data errors.
d. The received message contained an integral number of bytes (counted after the bit deletion performed during decoding), and if it was a data message, was at least the minimum 4 bytes in length.
e. No receiver overflow conditions occurred during reception of the message.

Failure of a message to meet these criteria will cause the receiving node to reject (ignore) the message and, if assigned Acknowledge responsibility, it will not return the ACK message.

APPENDIX A

DETERMINATION OF MESSAGE LATENCIES

A.1 Token Slot Message Data Overhead Summary (in bit times = bt):

	Synchronization bit	1	
	ID	16	
	Data (system determined limit, e.g., 0 to 256 bytes)	var	
	CRC	16	
	Intermessage delimiter gap	8	
	Acknowledge response (if requested) + gap	16	
	Per message overhead—with ACK:	57	bt
or	Per message overhead—without ACK:	41	bt

A.2 Token Passing Slot Overhead:

Slot width (assume xmtr on at mid slot = 1 bt)	1	
Token Pass Message	8	
Delimiter gap	8	
Token pass overhead per node:	17	bt

A.3 Summary of Loop Time Calculations:

For P nodes using R slots sending an N message loop with M total message data bytes:		
Total message overhead (no ACK)	=	41N
Total token overhead	=	17R
Total message data time (m x 8 bits)	=	8M
Unused slots = slot width x (max #slots–#used)	=	1(32–r)
Total Loop Time Formula		41N + 17R + 8M +1(32– R)

A.4 Example Token Slot Timing Calculation:

For a 32 node system using 8 slots to send a 16 message loop with 32 total message data bytes (2 msgs x 2 bytes) per slot x 8 slots:	
Total message overhead (no ACK)	41x16 = 656
Total token overhead	17x8 = 136
	792
Total message data time (16 msgs x 2 bytes x 8 bits)	+256
Unused slot time (1 bt per slot x 24 unused slots)	+ 24
Total Loop Time	1072 bt

Data efficiency = Total Data/Total Loop Time = 256/1072 = 24.0%