The Engineering Society For Advancing Mobility Land Sea Air and Space®
INTERNATIONAL
400 Commonwealth Drive, Warrendale, PA 15096-0001

SURFACE VEHICLE INFORMATION REPORT

Submitted for recognition as an American National Standard

SAE J1583

Issued 1990-03-16

CONTROLLER AREA NETWORK (CAN), AN IN-VEHICLE SERIAL COMMUNICATION PROTOCOL

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

FOREWORD

Controller Area Network (CAN) is an advanced serial communication protocol which efficiently supports distributed real-time control with a very high level of security.

Its domain of applications range from high speed networks (greater than 125K bits/s) to low speed multiplex wiring (10K bits/s or less).

1. SCOPE:

The scope of this specification is to define the transfer layer and the consequences of the Controller Area Network (CAN) protocol on the surrounding layers.

2. GENERAL FEATURES:

CAN has the following general features:

MULTIMASTER ARCHITECTURE: Assurance that any node can have access to the bus for transmission of messages.

PRIORITIZATION OF MESSAGES: Bus access is determined by the selectable predetermined priority of the 2032 different messages allowable.

GUARANTEE OF LATENCY TIME: Assurance that the maximum latency time for the highest priority message is less than 150 µs (at 1M bits/s).

BROADCAST MESSAGE TRANSFER: Any number of nodes can receive and simultaneously act upon the same message.

MESSAGE ARBITRATION: If two or more senders start transmitting messages at the same time, the bus access conflict is resolved by nondestructive prioritized bitwise arbitration via the message identifier.

DATA CONSISTENCY: Assures that a message is simultaneously accepted either by all nodes or by no node.

PROGRAMMABLE TRANSFER RATE: The bit-rate is programmable up to 1m bit/s; once set it is uniform and fixed in the system.

POWERFUL ERROR HANDLING: Includes automatic retransmission of corrupted messages and distinction between temporary errors and permanent failures of nodes.

CONFIGURATION FLEXIBILITY: Nodes may be added without requiring any change in the software or hardware of any node or application layer.

FAULT DETECTION: Distinction between temporary errors and permanent failures of node and autonomous switching off of defect nodes.

3. CAN ARCHITECTURE:

To achieve design transparency and implementation flexibility CAN has been developed structurally in layers. Different silicon implementations will integrate these layers to different depths in hardware. This information report will address the protocol, or bus communication rules, that all implementations must support or have the capability of supporting. One implementation example already in production is the Intel 82526. An overview of its functionality and use is provided in Appendix A. Another such implementation is BasicCAN. It is not within the scope of this document to cover BasicCAN in detail. Future updates may provide additional information.

The layers are:

a. Object Layer: The scope of the object layer includes:

1) Deciding which messages received by the transfer layer are actually to be used (acceptance filtering)
2) Determining which messages are to be transmitted (prioritized message handling)
3) Providing an interface to the host CPU if there is a host in the system.

b. Transfer Layer: The scope of the transfer layer includes:

1) Error checking
2) Error signalling and confinement
3) Message validation and acknowledgement
4) Performing arbitration
5) Controlling message framing
6) Fault confinement
7) General features of bit timing

c. Physical Layer: The scope of the physical layer is the actual transfer of the message bits between the different nodes in the network with respect to all electrical properties. This includes: Signal level, bit representation, and transmission medium. The protocol does not specify the signal level and transmission medium. This is done intentionally to allow the user or system designer to adapt CAN to the specific application and application environment. Within one network the physical layer, of course, has to be the same for all nodes. There may be, however, much freedom in selecting a physical layer.

4. BIT REPRESENTATIONS, CODING/DECODING, AND STUFFING:

4.1 Bit Representations:

There are two logical bit representations defined: "dominant" and "recessive". A recessive bit on the bus line appears only if all connected nodes at that time send a recessive bit. If one or more nodes send a dominant bit, the bus line reflects a dominant bit. In other words, a dominant bit always overwrites a recessive bit if emitted on the bus at the same time by a different node. If the electrical bus is implemented as an npn-open-collector bus, a dominant bit corresponds to a low level signal and a recessive bit corresponds to a high level signal.

4.2 Bit Coding/Decoding and Stuffing:

Bit encoding is performed with an NRZ bit waveform representation using bit stuffing. Bit stuffing is used in the following frame segments:

a. Start of frame
b. Arbitration field
c. Control field
d. Data field
e. CRC sequence

Whenever the transmit logic of CAN device detects five consecutive bits of identical levels to be transmitted, it inserts a complement bit in the transmitted bit stream. Whenever a CAN device receives five identical consecutive bit levels in the bit stream, its receive logic will automatically delete the next bit from the data stream (destuffing). See Figure 1 for further detail.

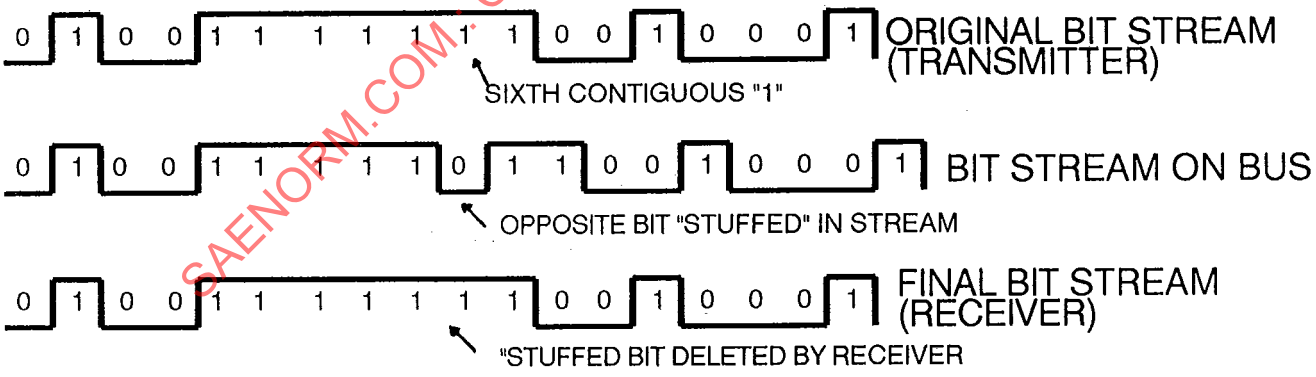

FIGURE 1 - Bit Stuffing

5. ARBITRATION:

The CAN protocol defines that each single message used in the communication network has a unique identifier. Using this method, the identifier assigns a name to the data frame and automatically implies the priority of the message. As a result, the identifier during bus access represents not only the message name but, more important, the priority of each specific message. Since the most significant bit (MSB) of an identifier is transmitted first, the identifier with the smallest digital value has the highest priority for bus access.

When bus idle is detected, any node may start to transmit a message. In a case where two or more CAN devices start a message transfer concurrently, the bus access conflict is solved by prioritized bitwise arbitration performed during the transmission of the arbitration field.

The transmit logic of a given node compares the bit level transmitted to the level monitored on the serial bus. The transmit logic immediately will stop a current message transfer if a recessive bit was sent but a dominant bit was monitored. This method guarantees the data transfer of the message with the highest priority even if there is a collision during the arbitration field of one or more messages.

An identifier can never be used for more than one specific message (total available number of identifiers = 2032). This guarantees that two or more nodes never simultaneously start a transmission of a data frame with the same priority of data. One exception would be a frame transfer simultaneously initiated by a transmitter and receiver. If one CAN device generates a request for actual data of a certain type by transmitting a remote frame and the CAN device normally transmitting that data transmits simultaneously, the CAN device responsible for this type of data will win the arbitration and continue the transmission. Hence, arbitration can not be solved by the identifier itself. For this reason, the remote transmission request (RTR-BIT) is included in the arbitration field. The RTR-BIT of the transmitter is always set dominant and, therefore, has a higher priority than the requesting RTR-BIT recessive. In this case, the remote transmission request by the receiver gets an immediate response by the transmitter.

6. CAN MESSAGE FRAME TYPES:

The CAN device supports four different frame types:

a. Data frame
b. Remote frame
c. Error frame
d. Overload frame

6.1 Data Frame:

A Data Frame is composed of seven different fields:

a. Start of frame
b. Arbitration field
c. Control field (Identifier)
d. Data field (Data segment)
e. CRC field
f. Acknowledgement field
g. End of frame

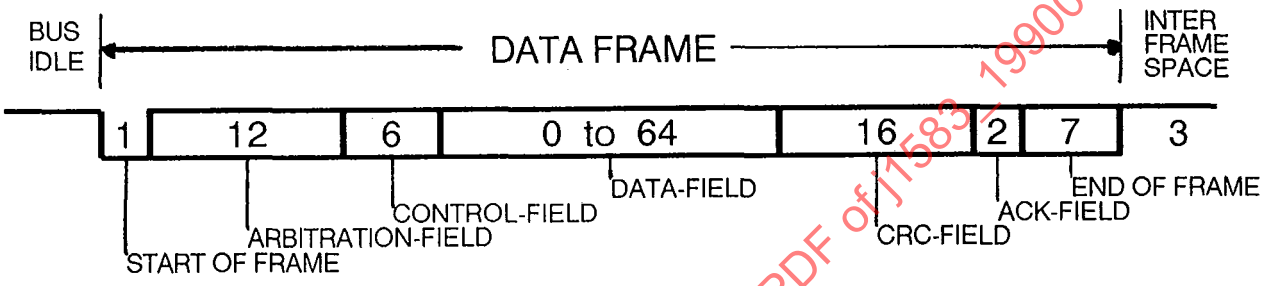

FIGURE 2 - Data Frame

6.1.1 Start of Frame: Signals the start of a data or remote frame. It consists of a single dominant bit used for a synchronization of receiving nodes. The CAN device will start a transmission only if a bus-idle state is detected. All nodes must synchronize to the leading edge of the node starting transmission first.

6.1.2 Arbitration Field: Consists of the message identifier and one additional control bit RTR-BIT. The simultaneous message transmission start of two or more nodes is solved by bitwise arbitration during the transmission of the arbitration field (for more details see Section 5).

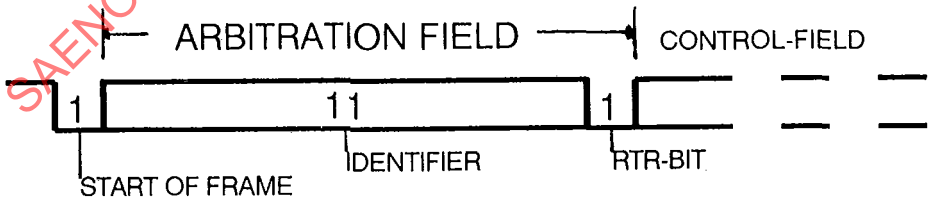

FIGURE 3 - Arbitration Field

6.1.2.1 Identifier: This 11-bit field provides information about each individual message as well priority for the message. The identifier defines the corresponding specific message (e.g., engine speed, temperature, pressure, etc.) of a communication object. It's digital value represents the message priority for bus access.

NOTE: An identifier may be a physical or functional address or a combination of the two to determine a receiver of a frame on a multidrop line. The receiving CAN device decides, based on the acceptance filter process of a received identifier, whether data being received within a correct frame is to be accepted or not.

6.1.2.2 RTR-BIT: (Remote Transmission Request) A station, active as a receiver may initiate the transmission of the data by transmission of a remote frame to the network, addressing the data source via the identifier. Within a data frame, the RTR-BIT is transmitted as a dominant bit level (for more information see 6.2).

6.1.3 Control Field: This field consists of 6 bits. It includes the data length code and two reserved bits. The reserved bits are automatically transmitted with a dominant bit level. The length of the data field (data segment) is coded in bytes.

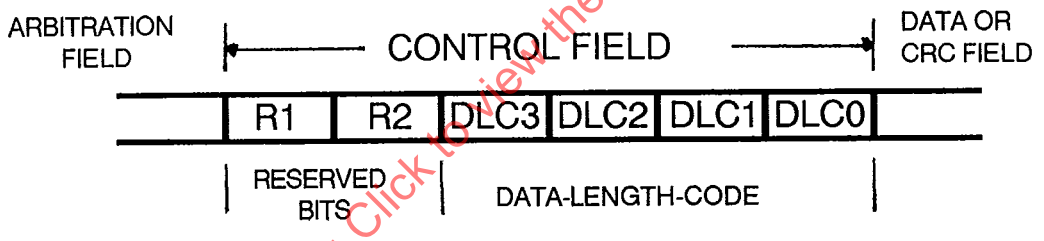

FIGURE 4 - Control Field

6.1.4 Data Field: Data stored within the corresponding data segment in the Communication Buffer is transmitted within the data field. The length of the data field varies in the range from 0 to 8 bytes based on the value of the data length code. The byte at the lowest address of the data segment will be transmitted MSB first.

# DATA BYTES	DATA-LENGTH-CODE			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

"d" = dominant, "r" = recessive

FIGURE 5 - Data Length Code

6.1.5 CRC Field: Contains the CRC checksum (15 bits) followed by the CRC delimiter (1 bit). The cyclic redundancy code includes the start of frame bit, arbitration field, control field, data field and CRC field. The most significant bit (MSB) is transmitted first. The frame check sequence implemented in the CAN device is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits (BCH Code). With start of frame included in the code word, any rotation of the code word can be detected by the absence of the CRC delimiter ("recessive" bit). (For further information about the CRC, see 7.6.)

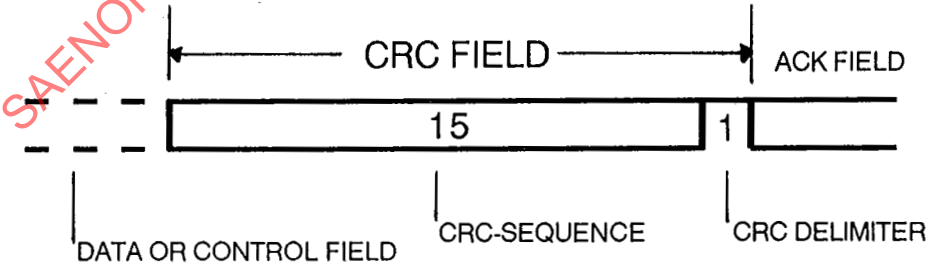

FIGURE 6 - CRC Field

6.1.6 Acknowledgement Field: The ACK-field consists of two bits, the ACK-slot and ACK-delimiter which are sent with a "recessive" level by the transmitter. Any receiving CAN device acknowledges to the transmitting CAN device a match of the complete/correct CRC check sum within the ACK-slot by superscribing this recessive bit with a dominant bit. A transmitter monitoring the bus level recognizes that at least one receiver within the system has received a complete and correct message (no error was found). The ACK-SLOT is surrounded by two recessive bits: The CRC delimiter and the ACK delimiter.

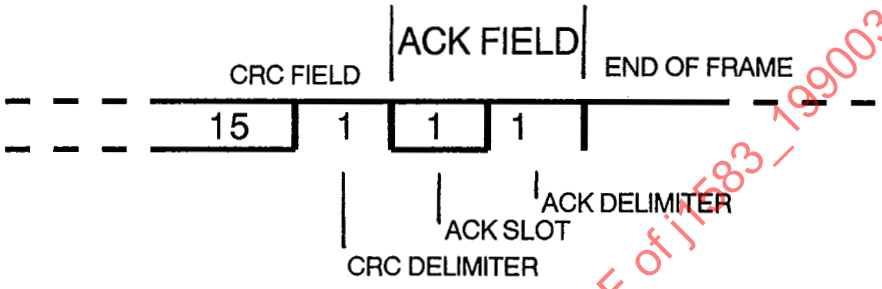

FIGURE 7 - Acknowledgement Field

6.1.7 End of Frame: Each data frame or remote frame is delimited by the end of frame bit sequence which consists of seven "recessive" bits (exceeding the bit stuff width by 2 bits). A receiver detects the end of a data frame independent of a previous transmission error because the receiver expects all bits up to the end of CRC-field to be coded by the bit stuffing method and bit stuffing is not used in end of frame.

6.2 Remote Frame:

A remote frame is composed of six different fields:

a. Start of frame
b. Arbitration field
c. Control field
d. CRC field
e. Ack field
f. End of frame

Contrary to the data frame, the RTR-BIT of the remote frame is "recessive" and no data segment is transmitted (regardless of the data length code set in the descriptor of the corresponding communication object).

6.2 (Continued):

The RTR-BIT allows remote transmission requests from any node to the system. This provides the capability to request information in addition to the standard broadcast characteristics. It also supports powerful diagnostic capability by being able to determine if the primary supplier (data source) of a specific parameter(s) is nonfunctional.

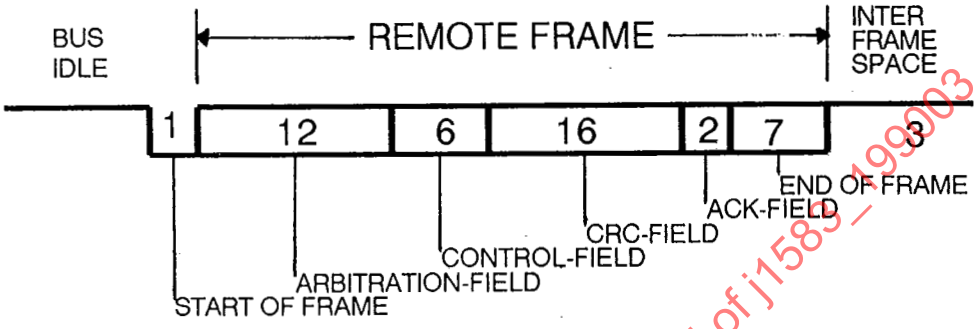

FIGURE 8 - Remote Frame

6.3 Error Frame:

An error flag is transmitted if a CAN device operates as an error active node and has detected an error condition during or after a message transfer. A receiving or transmitting node may generate an error flag during or after a transmission. The error flag consists of six consecutive dominant bits. Since this "violates" bit stuffing guidelines, it is used as an error indicator to the system (see 4.2). If an error flag is generated by a transmitter, or a receiver, all other nodes interpret the error flag as a bit stuffing rule violation. As a consequence, the transmission of an error flag occurs.

The error-delimiter consists of seven recessive bits generated by the CAN device after the end of an error flag on the serial bus line. This is monitored by detection of a transition from the dominant to recessive bit level.

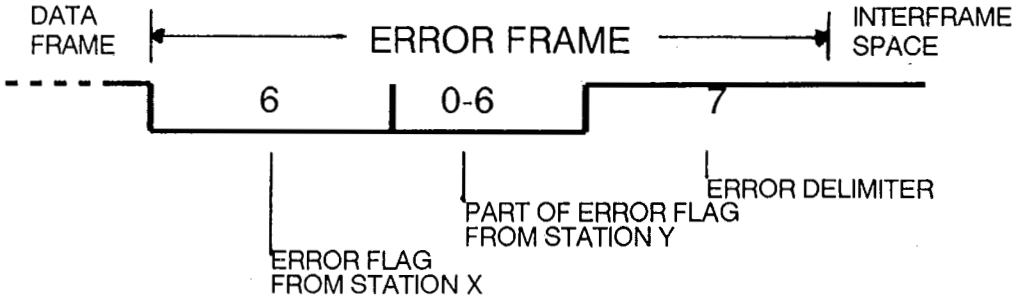

FIGURE 9 - Error Frame

6.3 (Continued):

The bit sequence of dominant bits result from a superposition of different error flags transmitted by individual nodes. The total length of the error flag sequence varies between 6 bits minimum to 12 bits maximum. An error condition is signaled by the transmission of six recessive bits while in the error passive operation mode. Hence, an error passive node with a temporary local receiver problem will not destroy messages received correctly by other nodes. The recessive bits may be overwritten by an error flag generated by one or more error active system nodes, but the error passive CAN device waits for at least six bits of equal polarity before entering into the next internal receive or transmit mode. See 7.7 for additional error active/passive mode data.

Detected errors during the transmission of a data or remote frame can be signaled within the transmission time of the respective frame. This procedure associates an error flag to the frame and initiates an automatic retransmission of the frame. If the CAN device detects any deviation of its error frame it will start retransmitting an error frame. If this occurs several times in a sequence the CAN device will become error passive.

6.4 Overload Frame:

The overload frame consists of 2 bit fields: The Overload Flag and the Overload Delimiter.

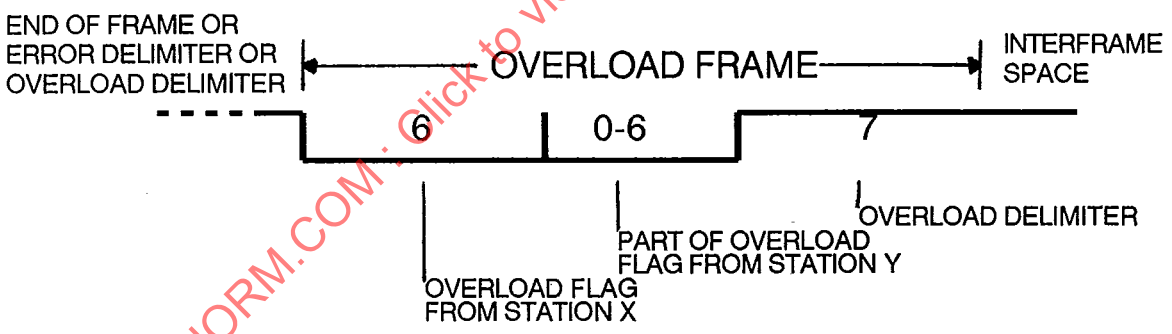

FIGURE 10 - Overload Frame

There are two cases of overload conditions which result in the transmission of an overload flag:

a. Internal conditions of the receiver circuitry of the CAN device which require a delay time before receiving the next frame (receiver not ready).

b. Detection of a "dominant" bit during intermission.

At most, two overload frames may be generated to lengthen one data frame or remote frame.

6.4 (Continued):

The overload flag consists of six dominant bits that correspond to the error flag and destroy the fixed form of the intermission field. As a result, all other stations also detect an overload condition and start transmission of an overload-flag. If a "dominant" bit is detected during the 3rd bit of intermission by some but not all nodes, the other nodes will interpret the first of these six 'dominant' bits as start-of-frame. The sixth "dominant" bit will then violate the bit stuffing rule and cause an error condition. The overload delimiter consists of seven recessive bits generated by the CAN device.

NOTE: An overload frame can be transmitted earliest at the first bit time of the interframe space field. This is contrary to the error frame and allows the CAN device to differentiate between the two frame types.

6.5 Interframe Space:

Data frames and remote frames are separated from preceding frames by an interframe space consisting of the intermission bit field and a possible bus idle time. An error frame is not preceded by an interframe space.

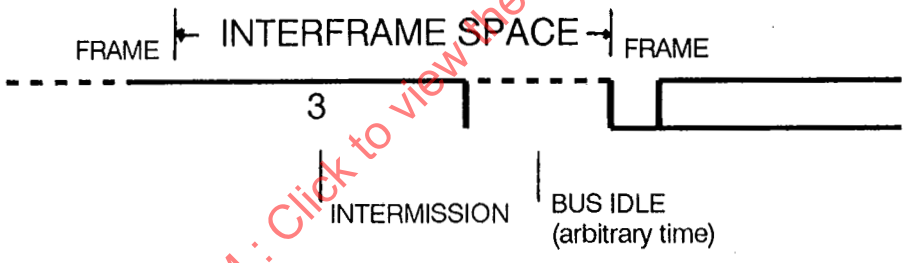

FIGURE 11 - Interframe Space

Intermission consists of three recessive bits. During intermission time the CAN device will not start transmission of a frame. Intermission requires a fixed time period for the CAN device to execute internal processes prior to the next receive or transmit task.

Data received within a data frame will be stored in the communication buffer and the control bits are updated if no error condition has occurred through last bit of the end of frame field.

The bus idle time may be of arbitrary length. After the interframe space period, CAN devices look for bus idle before initiating the next transmissions. The detection of a dominant bit after intermission or bus idle is interpreted by the CAN device as start of frame.

7. ERROR DETECTION:

7.1 Bit Error:

During a transmit operation, the CAN device monitors the bus on a bit-by-bit basis. If the bit level monitored is different from the transmitted one, a bit error is signaled.

The exceptions are during arbitration and ACK-SLOT. During arbitration, a recessive bit can be overwritten by a dominant bit. In this case, the CAN device interprets a bit error as an arbitration loss. During the ACK-SLOT, a transmitter may detect a falsified bit (recessive to dominant). This situation will only occur if all receivers have detected a CRC error and, therefore, this bit error will not be detected by a CAN device. This error is not critical because an error frame by the receivers will be generated after the ACK-SLOT.

Except during transmission of the arbitration field and during the time window of the ACK-SLOT, all global and local errors at the transmitter are detected.

7.2 Bit Stuffing Error:

There are two possibilities where bit stuffing errors may occur:

a. A disturbance generates more consecutive bits of equal level than allowed by the rule of bit stuffing. These errors are detected by all nodes.

b. A disturbance falsifies one or more of the 5 bits preceding the stuff bit. This error is not recognized by a receiver, but if an error appears at the transmitter as well, it will be detected as a bit error.

Otherwise, the error is detected by a receiver either by the bit stuffing mechanism (the stuff bit of transmitter is not dropped but taken as an information bit) or by the CRC check.

7.3 CRC Error:

To ensure the validity of a transmitted message, all receivers perform a CRC check. In addition to the information digits, any code word includes control digits used for error detection.

7.3.1 Description of the CRC Code: The code used for the CAN device is a (shortened) BCH Code, extended by a parity check and has the following attributes:

a. 127 bits as maximum length of the code word
b. 113 bits as maximum number of information digits
c. Length of the CRC SEQUENCE 15 bits
d. Hamming distance d = 6
   d = min A (x EXOR y) / x,y different code words
   A(x) = number of "recessive" bits in the code word x

7.3.1 (Continued):

As a result, d-1 random errors are detectable (some exceptions exist).

The CRC sequence is determined by the request that the code word, if interpreted as polynomial with coefficients 0 or 1 is devisable by the polynomial.

$$f(x) = (xE14 + xE9 + xE8 + xE6 + xE5 + xE4 + xE2 + x + 1) \quad \text{(Eq.1)}$$
$$= 1100010110011001 = 0C599 \text{ HEX}$$

Burst errors are detected up to a length of 15 (degrees of f(x)). Multiple errors (number of disturbed bits at least d = 6) are not detected with a residual error probability of 3*10E-5.

7.4 Form Error:

Form Errors result from the violation of the fixed form of the following bit fields:

a. End of frame
b. Interframe space
c. Ack delimiter
d. CRC delimiter

During the transmission of these bit fields, an error condition is recognized if a "dominant" bit level is detected.

7.5 Acknowledgement Error:

An Acknowledgement Error has to be detected by a transmitter whenever it does not monitor a "dominant" bit during ACK-SLOT.

7.6 Error Detection Capabilities:

Global errors, which occur at all fully functional nodes, are 100% detectable.

For local errors, e.g., errors which may appear at some nodes only, the shortened BCH Code extended by the parity check has the following error detection capabilities:

a. Up to 5 single bit errors are detected 100% even if those errors are being disturbed randomly within the code word.

b. All single bit errors are detected if their total number within the code word is odd.

c. The residual error probability of the CRC check is 2E-15 or 3*10E-5. As an error may be detected by the CRC check, and/or by additional implemented error detection mechanisms, the residual error probability is significantly less than the above.

7.7 Error Confinement:

7.7.1 Error Active: Normal operation for each node is an error active node. If an error is detected during transmission of a frame, a node enters into the send error flag state (see 6.3).

7.7.2 Error Passive: An error passive node, like an error active node, may function as a receiver and/or transmitter, but actions based on transmit and/or receive error conditions are different. As an example, after detection of an error, an error passive node will send six recessive bits. If the error passive node is the transmitter of a disturbed frame, all other nodes detect an error with the six recessive bits since it violates the bit stuffing rule. An error passive receiver does not signal the detection of an error to the system. An error passive node will, however, acknowledge the reception of a valid frame during the ACK-SLOT.

7.7.3 Off Bus: A node may enter the off-bus mode and will not receive or transmit any message until a reset is sent by the host CPU and predefined "wait time" (128 x 11 recessive bits) has passed.

7.7.4 ACK: A correctly received message is signaled to the transmitting node by setting a dominant bit level on the bus in the ACK-SLOT of the respective frame.

7.8 Error Counting:

For error confinement two counts are implemented in every bus unit:

a. Transmit-Error-Count
b. Receive-Error-Count

These counts are modified according to the following rules: (note that more than one rule may apply during a given message transfer)

a. When a receiver detects an error the receive-error-count will be increased by one, except when the detected error was a bit error during the sending of an active-error flag.

b. When a receiver detects a 'dominant' bit as the first bit after sending an error flag the receive-error-count will be increased by eight.

c. When a transmitter sends an error flag the transmit-error-count is increased by eight.

Exception 1: If the transmitter is 'error-passive' and detects an acknowledgement error because of not detecting a 'dominant' ACK and does not detect a 'dominant' bit while sending its passive-error flag.

Exception 2: If the transmitter sends an error flag because a stuff error occurred during the arbitration whereby the stuff bit should have been 'recessive' and has been sent as 'recessive' but monitored as 'dominant'.

In exception 1 and 2 the transmit-error-count is not changed.

7.8 (Continued):

d. If an 'error-active' transmitter detects a bit-error while sending an error flag.

e. If an 'error-active' receiver detects a bit error while sending an error flag, the receive-error-count is increased by eight.

f. After the successful transmission of a message (getting ACK and no error until end of frame is finished) the transmit-error-count is decreased by one unless it was already zero.

g. After the successful reception of a message (reception without error up to the ACK-SLOT and the successful sending of the ACK bit), the receive-error-count is decreased by one if it was between one and 127. If the receive-error-count was zero, it stays zero, and if it was greater than 127 then it will be set to a value between 119 and 127.

Based on the above rules, a node can enter the following states:

a. A node is 'error passive' when the transmit-error-count equals or exceeds 128, or when the receive-error-count equals or exceeds 128.

b. A node is "off-bus" when the transmit-error-count is greater than or equal to 256.

c. An "error-passive" node becomes "error-active" again when both the transmit-error-count and the receive-error-count are less than or equal to 127.

d. A node which is "off-bus" is permitted to become "error-active" (no longer "off-bus") after reset with its error counters both set to 0 after 128 occurrences of 11 consecutive "recessive" bits have been monitored on the bus.

An error count value greater than about 96 indicates a disturbed bus. It may be of advantage to provide means to test for this condition.

If during system start-up only one node is on-line, and if this node transmits some message, it will get no acknowledgement, detect an error and repeat the message. It can become "error-passive" but not "off-bus" due to this reason. If a wake-up message is used, a node which sends this message can become "error-passive" for the same reason.

8. BIT TIMING:

A bit time is subdivided in a number of BTL cycles. This number results from an addition of the segments SJW 1, SJW 2, TSEG 1 and TSEG 2, plus the general segment INSYNC.

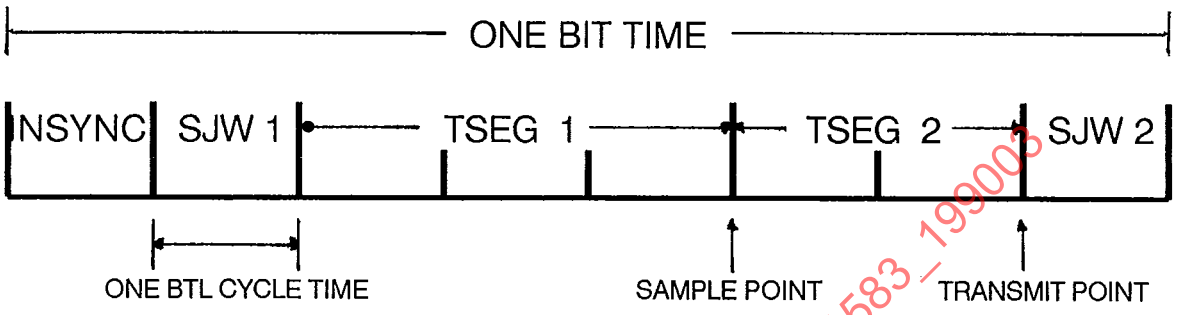

FIGURE 12 - Bit Segments

8.1 INSYNC:

The incoming edge of a bit is expected during this state; this segment corresponds to one BTL cycle.

8.2 SJW 1 & 2:

Both segments determine the maximum synchronization jump width and are programmable from 1 to 4 BTL cycles. The width of SJW 1 is increased to a maximum of two times the bit time during resynchronization. The width of SJW 2 is reduced or canceled to shorten the bit time during resynchronization.

8.3 TSEG 1:

Determines the sampling point based on the number of BTL cycles programmed by TSEG 1 (4 bits). The sampling point is located at the end of TSEG 1 (if sampling only once per bit). TSEG 1 is used to compensate delay times on the bus and to have some reserved time to tolerate one or more nonsynchronization pulses caused by spikes on the bus line. TSEG 1 is programmable from 1 to 16 BTL cycles.

8.4 TSEG 2:

Defines the time between the sampling point and the end of the bit time; programmable from 1 to 8 BTL cycles. This segment is necessary to tolerate one or more nonsynchronization spikes on the busline. Also necessary to guarantee sufficient time to generate a transmit signal dependent on the sampled bus level. The transmit point is determined internally in such a way that with zero delay the generated transmit signal will appear within the INSYNC state. For example, no transmit signal would be generated if an arbitration was lost. This guarantees that the transmit logic immediately stops and immediately enters into the receive mode.

8.5 Bit Time Calculation:

The number of clock cycles at every bit time determines (together with the oscillator frequency and the baud-rate-prescaler) the period of each bit time, and as a consequence, the baud rate.

bit time = (INSYNC + SJW1 + TSEG1 + TSEG2 + SJW2) BTL cycles (Eq.2)

BTL cycle = 2 * Toscillator * (baud-rate-prescaler + 1)

BAUD rate = 1 / bit time

9. SYNCHRONIZATION:

Synchronization is performed by comparing the incoming edges with the intended bit timing, and adapts the bit timing by hard synchronization or resynchronization.

Hard synchronization occurs only at the beginning of a frame. The CAN device synchronizes to the first incoming recessive to dominant edge of a frame.

Resynchronization occurs during the message bit stream to compensate differences in the oscillator frequencies of individual CAN devices as well as changes introduced by switching from one to another transmitter; e.g., after arbitration of two or more nodes. SJW 1 and SJW 2 define the maximum number of clock cycles a bit time may be shortened or lengthened by resynchronization.

There are two modes of sampling.

a. Resynchronization may take place on both edges. In this case, resynchronization is always done on the edge of a bus level which is different from the one read at the last sample point.

b. Resynchronization may take place on the edge of a dominant level only. In this case, resynchronization is done if the bus level monitored at the last sample point was a recessive level.

These modes are necessary due to physical bus characteristics at the maximum cable length and high baud rates.

Synchronization is done once during a bit time and is released after the sample point dependent on the level of the actual bit. Thereafter, the CAN device synchronizes to the next relevant edge and synchronization is suppressed until the following sample point.

By resynchronization, the bit time can be lengthened or shortened based on the following conditions:

a. If an edge appears in the segment SJW 1 or TSEG 1, then lengthen.
b. If an edge appears in the segment SJW 2 or TSEG 2, then shorten.

10. SLEEP-MODE/WAKE-UP:

In order to wake up other nodes of the system, which are in sleep-mode, a special wake-up message with the dedicated lowest possible identifier (rrr rrrd rrr where r = 'recessive'and d = 'dominant') may be used.

APPENDIX A

A1. 82526 FUNCTIONAL OVERVIEW:

The CAN major functional blocks are:

a. Interface management processor (IMP)
b. Quasi dual port RAM
c. Transceiver control logic (TCL)
d. Bit stream processor (BSP)
e. Error management logic (EML)
f. Bus timing logic (BTL)
g. Processor interface unit (PIU)
h. Clock generator (CG)

A block diagram is shown Figure A1.

A1.1 IMP (Interface Management Processor):

The IMP executes commands from the CPU and controls data transmissions on the serial bus. This processor computes addresses for the many communication buffer accesses required during normal operation. Global status and control register bits, as well as the control bits of the descriptor are used primarily by the interface management processor. The IMP consists of a data path section with various local registers including an ALU with an accumulator and a control section for receive and transmit processes.

A1.2 Quasi Dual Port RAM:

The host CPU communicates with the CAN device through a quasi dual port RAM memory which includes global status registers and control registers. This memory serves as the communication buffer interface between the host CPU and the IMP. The host CPU initializes the global status and control registers and creates data structures called communication objects within the communication buffer for reception and transmission of defined messages. The memory size is 64 bytes.

A1.3 TCL (Transceiver Control Logic):

The transceiver control logic consists of bit stuffing logic, programmable output driver logic, CRC logic and shift registers.

A1.4 BSP (Bit Stream Processor):

The Bit Stream Processor controls the data stream between the IMP (parallel data) and the bus line (serial data). Message reception and transmission, bus arbitration, error signaling and control of the TCL are duties of the BSP.

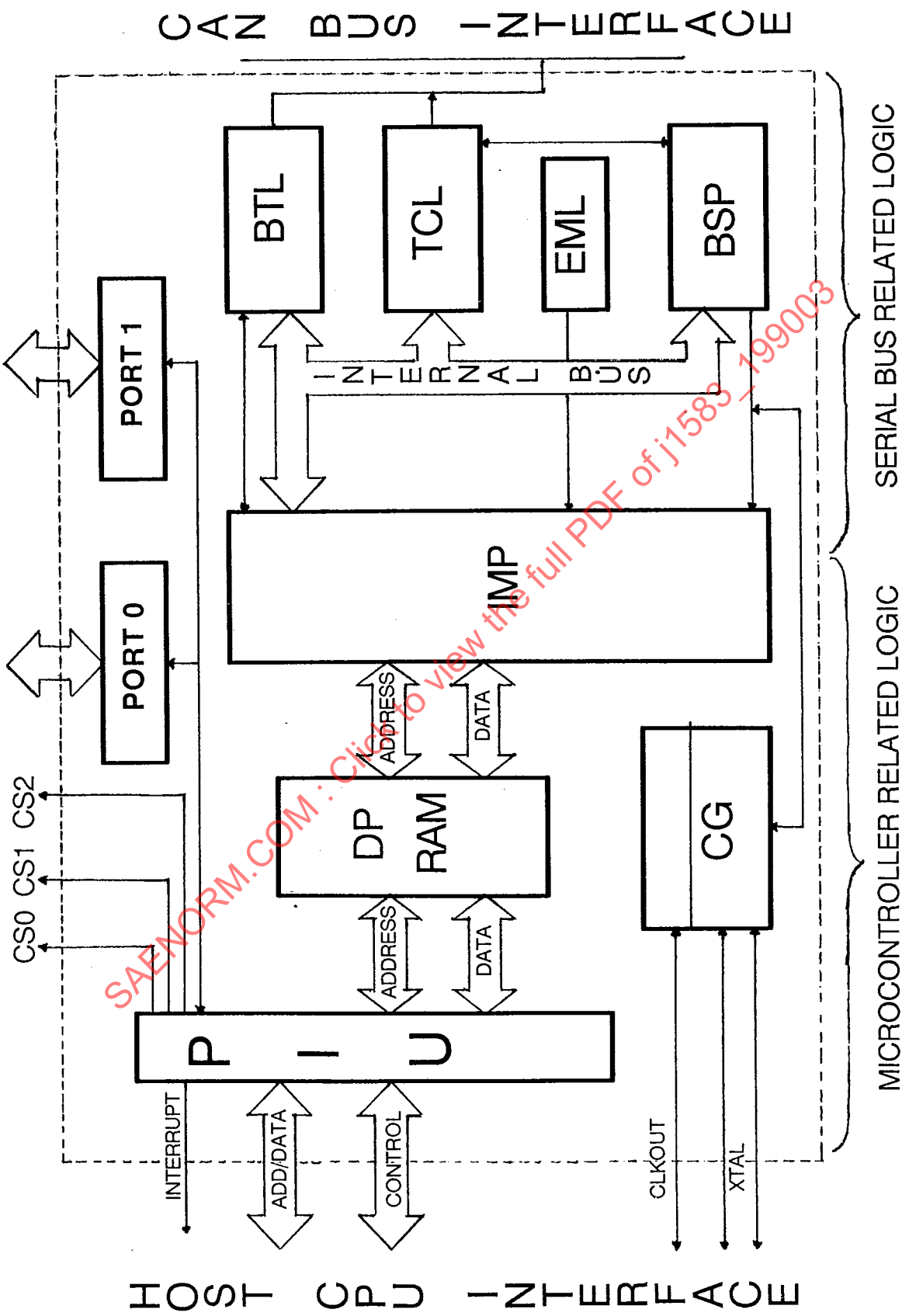

FIGURE A1 - Functional Block Diagram

A1.5 EML (Error Management Logic):

Errors are reported to the EML by the bit stream processor (BSP). The EML informs the BSP, TCL and IMP of error statistics.

A1.6 BTL (Bus Timing Logic):

This logical block monitors the busline through an input comparator and performs the busline related bit timings. The BTL synchronizes on a busline transition at the start of a frame and resynchronizes on further transitions during the reception of a frame. The BTL also provides programmable time segments to compensate for propagation delay times and phase shifts. This feature determines the sampling point and the number of samples within a bit-time slot.

A1.7 PIU (Processor Interface Unit):

The Processor Interface Unit provides for the link to the host CPU. The PIU consists of the 8-bit multiplexed address and data bus, read/write control, address latch enable, chip select, interrupt output, external interrupt input, reset, ready output signal, two 8-bit I/O ports (Port 0 and Port 1), and three chip select output lines for additional system peripheral devices.

A1.8 CG (Clock Generator):

The on-chip clock generator consists of an oscillator, clock divider register and driver circuit. The oscillator is a high-gain parallel resonance circuit. The oscillator is driven by an external crystal or, in case of low baud rates, with a ceramic resonator. A host CPU can be driven from the clock output which uses the programmable divider to select the output.

A2. QUASI DUAL PORT RAM MEMORY LAYOUT AND CONTROL REGISTERS:

The on-chip quasi dual port RAM (communication buffer) is seen as part of the host CPU's memory map and may be defined as a "decoupling device" between the CPU and the "Bus-Interface" of the CAN device. The buffer memory area used for communication is configured by the user during an initialization download after power-up. It consists of dedicated control registers, communication objects, and an end mark signifying the end of RAM area used for communication. Each individual communication object is composed of the identifier, control bits, and data. The CPU programmer operates only on this communication buffer to perform message transfers. The buffer memory layout is shown in Figure A2.

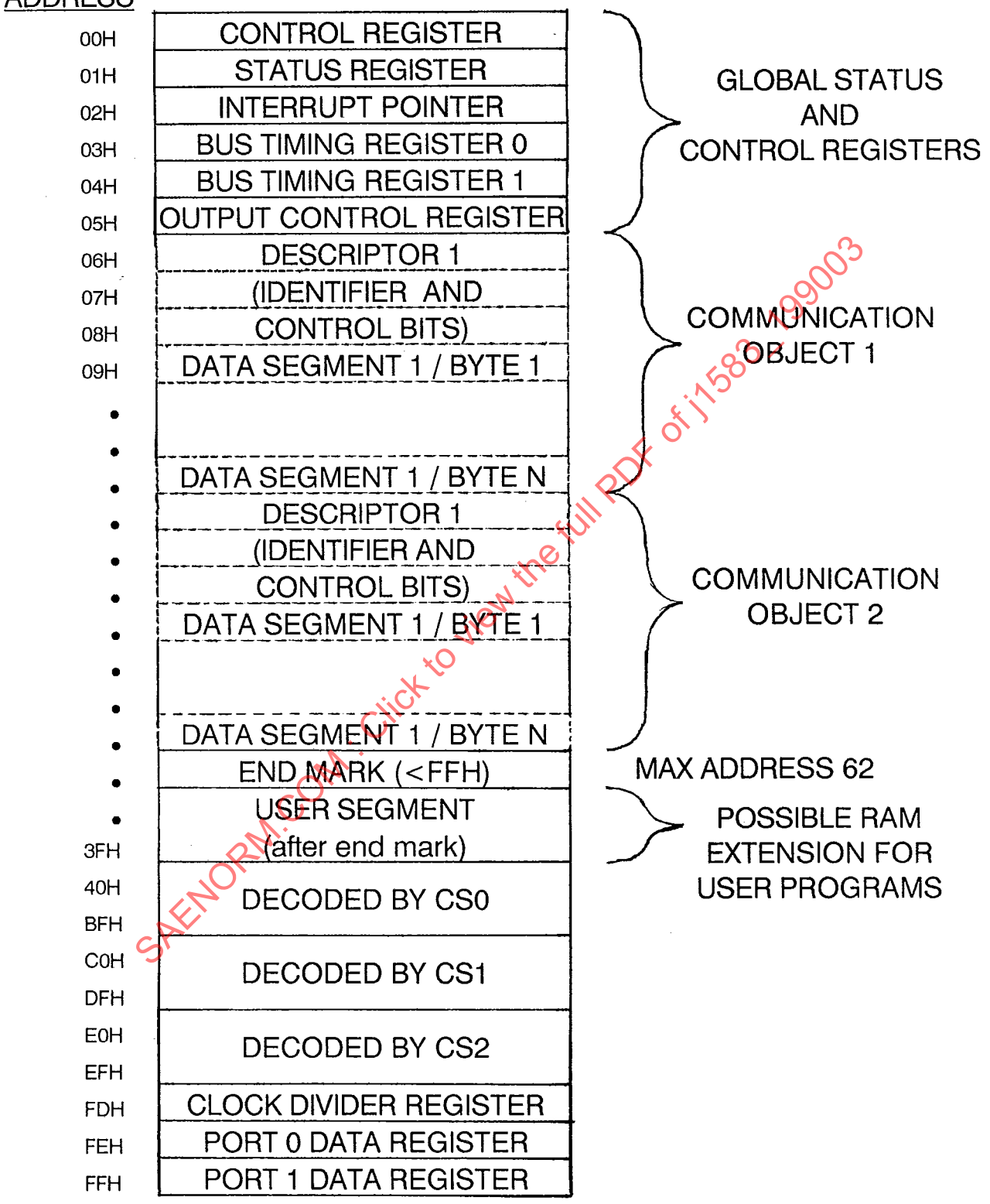

FIGURE A2 - Buffer Memory Layout

A2.1 Control Register (Address 00H):

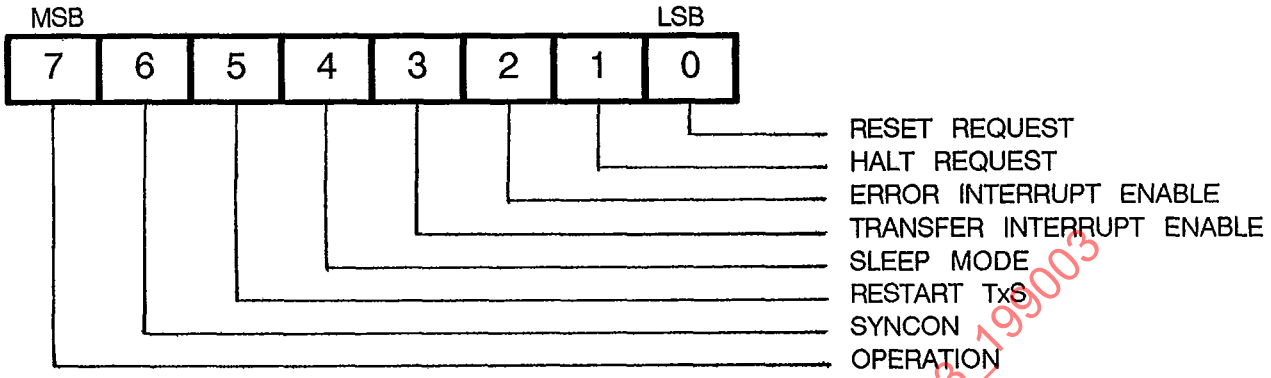

FIGURE A3 - Control Register (00h)

A2.1.1 RESET REQUEST is set by the host CPU and read by the IMP. If set to "high" and detected by the IMP, it causes an internal hardware reset.

A2.1.2 HALT REQUEST is set by the host CPU and read by the IMP. When set to "high" the CAN device continues processing the current transmission or reception and then stops any further activity until the halt request bit is set to "low" which restarts normal operation.

A2.1.3 ERROR INTERRUPT ENABLE is set by CPU and read by the IMP. If set to "high", the interrupt output signal of the CAN device to the host CPU is enabled, and disabled if set to "low".

A2.1.4 TRANSFER INTERRUPT ENABLE is set by the host CPU and read by the IMP. If set to "high", the CAN device will generate an interrupt to the host CPU after a message is successfully received or transmitted.

A2.1.5 SLEEP MODE is set by the host CPU and read by the IMP. If the bit is set to "low" (normal operation) the CAN device will not enter the sleep mode. If set to "high", the device may enter into the sleep mode based on bus traffic (both transmission and reception). A return from the sleep mode to the normal operation mode may be activated by the following events:

a. Start bit detection
b. Any CPU access to the on-chip quasi dual port RAM, Port 0 or Port 1, or to the clock divider register

A2.1.6 RESTART TxS (Restart Transmit Search) Any write access by the CPU to set this bit to "high" forces the IMP to start a new transmit search loop at the first communication object.

A2.1.7 SYNCON is set by the host CPU and read by the IMP. This bit controls resynchronization of the bit timing logic during the reception of a frame.

A2.1.8 OPERATION bit must be set to "low" by the host CPU during initialization. It will force the CAN device to start operation as soon as the reset status bit is reset by the IMP.

A2.2 Status Register (Address 01H):

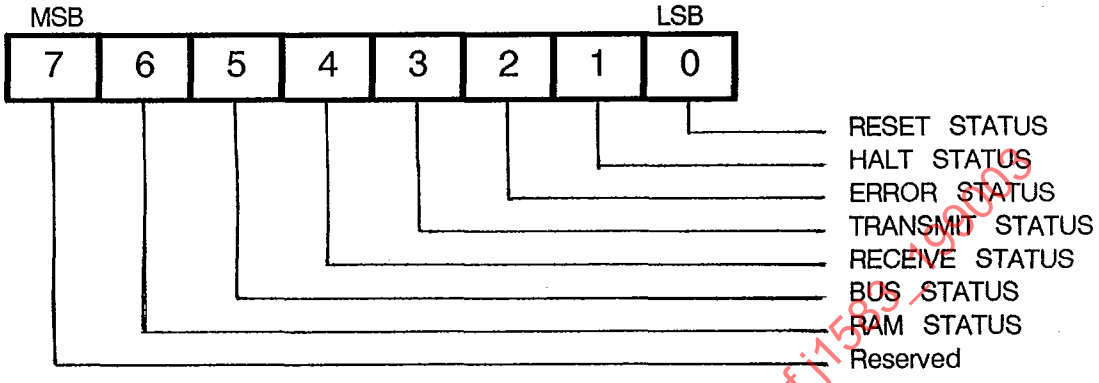

FIGURE A4 - Status Register

A2.2.1 RESET STATUS bit set to "high" by the IMP acknowledges to the host CPU that the reset request from the CPU was detected and an internal hardware reset is being performed by the CAN device.

A2.2.2 HALT STATUS bit is altered corresponding to the halt request from the host CPU.

A2.2.3 ERROR STATUS bit set to "high" by the IMP indicates to the host CPU that the error management logic (EML) has detected a major problem either with transmission or reception of messages.

A2.2.4 TRANSMIT STATUS bit set "high" indicates to the host CPU that the CAN device currently is in the transmit mode.

A2.2.5 RECEIVE STATUS bit set "high" indicates to the host CPU that the CAN device is in the receive mode.

A2.2.6 BUS STATUS bit is modified by the IMP and read by the host CPU. If set to "low", the CAN device is bus active (bus-on). If set to "high" the CAN device remains in the bus-off mode until a reset request was set by the host CPU and executed by the IMP.

A2.2.7 RAM STATUS bit normally is set to "low" by the IMP after RAM configuration set up is completed by the host CPU and there was no buffer memory configuration inconsistency detected by the IMP.

A2.3 Interrupt Pointer (Address 02H):

The interrupt pointer is an 8-bit register that contains either the address of the status register (error related interrupt), or the RAM address of a communication object with highest priority for which the conditions transfer interrupt enable is enabled and transfer status is completed.

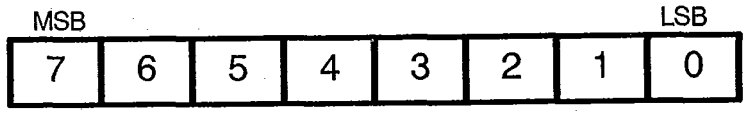

FIGURE A5 - Interrupt Pointer (02H)

A2.4 Bus Timing Register 0 (Address 03H):

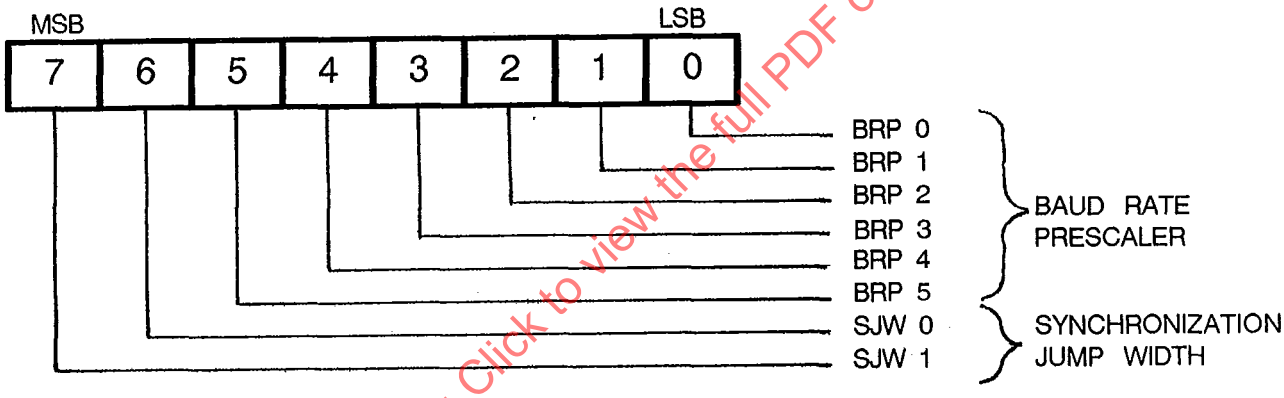

FIGURE A6 - Bus Timing Register 0 (03H)

A2.4.1 Baud Rate Prescaler: By programming the six bits of the baud rate prescaler, the BTL cycle time is determined. The BTL cycle time is derived from the system cycle time. The desired baud rate is determined by the BTL cycle time and the programmable bit timing segments.

BAUD RATE PRESCALER BRP0-BRP5	BIT CYCLE TIME
000000	1 * System Cycle Time
000001	2 * System Cycle Time
000010	3 * System Cycle Time
:	:
:	:
111111	64 * System Cycle Time

FIGURE A7 - Baud Rate Prescaler

A2.4.2 Synchronization Jump Width: These two bits are introduced to compensate for phase shifts between clock oscillators of different bus nodes. Any node may resynchronize to a relevant transition of the bus bitstream of a transmitter. The synchronization jump width defines the maximum number of BTL cycles. A bit may be shortened or lengthened by one resynchronization during transmission of a data frame or remote frame.

A2.5 Bus Timing Register 1 (Address 04H):

The number of samples per bit, the delay time and the phase shift buffer are programmed by bus timing register 1.

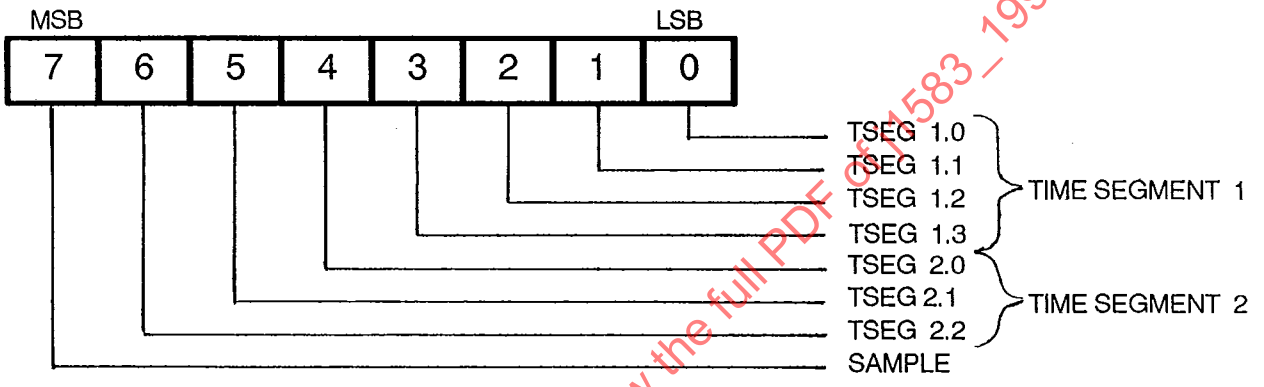

FIGURE A8 - Bus Timing Register 1 (04H)

A2.5.1 Bus Sample: This bit determines whether a bit is sampled directly by the BTL or if each bit first is filtered by the majority logic. If sample is set to "low", a bit is sampled once by the BTL, and if set to "high", three samples per bit are taken.

A2.5.2 Time Segment 1/Time Segment 2: Time segments within a bit time are introduced to determine the number of BTL cycles per bit time and the location of the sample point within a bit time.