CAN BASED CLUSTER CONTROLLER

PROJECT REPORT

SUBMITTED BY

S JAGANATHAN

MAYA SUBRAMANIAN

RONY JOHN GEORGE

During the academic year **2003 - 2004**

DONE IN

PRICOL, COIMBATORE

GUIDED BY

Mr. RAM PRAKASH M.E., (KCT)

Mr. NANDHA KUMAR B.E., (PRICOL)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING IN ELECTRONICS & COMMUNICATION ENGINEERING OF THE BHARATHIAR UNIVERSITY, COIMBATORE.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore - 641 006

Department of Electronics of Communication Engineering

CERTIFICATE

This is to certify that the project entitled

CAN BASED CLUSTER CONTROLLER

Has been submitted by

S JAGANATHAN (0027D0195)
MAYA SUBRAMANIAN (0027D0205)
RONY JOHN GEORGE (0027D0222)

In partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering in **Electronics & Communication Engineering** branch of **BHARATHIAR UNIVERSITY, Coimbatore**

During the academic year 2003-2004

(Internal Guide) 11/3/04

(Head of the Department)

Certified that the candidate was examined by us in the project work

Viva – Voce Examination held on _____

(Internal Examiner)

(External Examiner)

ACKNOWLEDGEMENT

We are greatly indebted to our beloved Principal **Dr.K.K. Padmanabhan B. Sc. (Engg.), M.Tech., Ph. D.**, who has been the backbone of all our deeds.

We earnestly express our gratitude and sincere thanks to our guide **Mr. Ram Prakash M.E., Dept. of Electronics and Communication Engineering** for his consummate technical guidance and constant encouragement with suggestions in carrying out this project successfully.

We would also like to thank **Prof. M. Muthuraman Ramaswamy M.E., FIE., FIETE,.MIEEE(USA).,MISTE.,MBMESI.,C.ENG.(I).,Head of the Department of Electronics and Communication Engineering** for lending a helping hand in this project.

We also express our sincere thanks to **M/S PRICOL, Coimbatore** for giving us an opportunity to do this project in their esteemed firm. We also express our deepest gratitude to **Mr. Nandha Kumar B.E.**, our company guide, for his valuable guidance in the successful completion of this project.

We also express our thanks to **all the faculty members of the Department of Electronics and Communication Engineering** for their support.

We owe much to our **parents and friends** for their moral support and valuable help rented to us.

SYNOPSIS

A CAN based cluster controller has been developed to reduce the wiring complexity in an automobile's Electronic Control Unit (ECU). It also helps in improving the response time of a particular operation.

We use a Motorola controller-MC68HC908GZ16 as the controlling unit. This controller has an MSCAN module in it which constitutes a CAN transceiver that helps in the transmission and reception of data through a single twisted pair of CAN bus.

The software code for the transmission and reception of the data through CAN is written in 'C' language.

CONTENTS

	Pg.No.
1. INTRODUCTION	1
2. WHY CAN ?	2
2.1 What is CAN ?	3
2.2 The layered ISO 11898 standard architecture	4
2.3 Need for CAN & Advantages	4
2.4 Types of CAN	5
2.5 CAN Protocol	6
2.6 CAN Message format	7
2.7 Bus arbitration	8
2.8 CAN Error checking	9
2.9 CAN Applications	10
3. BLOCK DIAGRAM	11
3.1 Description of the modules in the block	12
4. CIRCUIT LAYOUT	13
4.1 Circuit diagram	13
4.2 Realization of the design	14
5. MICROCONTROLLER	15
5.1 Advantages of microcontroller	16
5.2 General features of MC68HC908G16	19
5.3 MSCAN Module	22
5.4 Monitor ROM	27

6. HARDWARE DESCRIPTION 28
6.1 Power supply 28
6.2 Speed & Tacho sensors 30
6.3 Differential Air core meter driver 31
6.4 SN75LBC031 – High speed CAN transceiver 33
6.5 Crystal oscillator 35
6.6 MAX 232 36
6.6.1 RS 232 38

7. SOFTWARE 41
7.1 Programming aspects 41
7.2 Flow chart 42
7.3 Algorithm 44
7.4 Program coding 47

8. CONCLUSION 50

APPENDIX

BIBLIOGRAPHY

INTRODUCTION

A Controller Area Network is used to reduce the wiring complexity that is involved in connecting the different gadgets in an automobile cluster.

Although different parameters like ABS, HVAC, Power window, Stereo system, Fuel gauge, Speedometer & Tachometer can be connected to the CAN bus, we have chosen speed and revolution's of the engine as the two testing parameters.

This project aims at standardizing the communication protocol of different parameters from their respective sensors to their displays.

2 WHY CAN

CAN – 'Controller Area Network' is a serial bus system especially suited for networking intelligent devices as well as sensors and actuators within a system or sub-system.

Our project aims at standardizing the automobile transmission cluster and subsequent components by using CAN.

CAN is an ISO-OSI defined bus originally developed for the automobile industry to replace the complex wiring harness with a simple two wire bus.

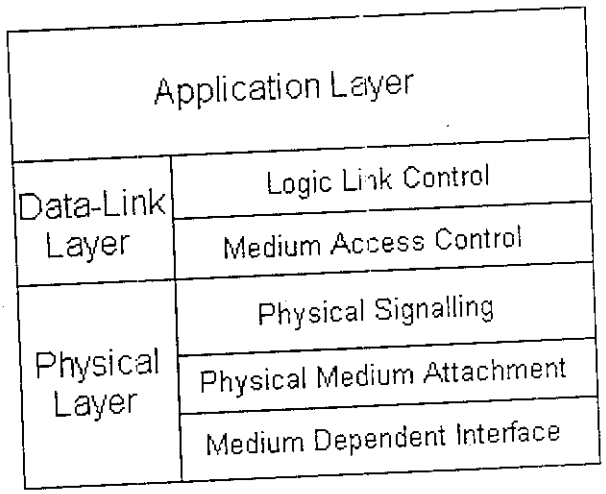
2.1 WHAT IS CAN

The CAN bus was developed by BOSCH as a multi-master message broadcast system that specifies a maximum signaling rate of 1 Mbit/sec, unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point to point from node A to node B under the supervision of the central bus master.

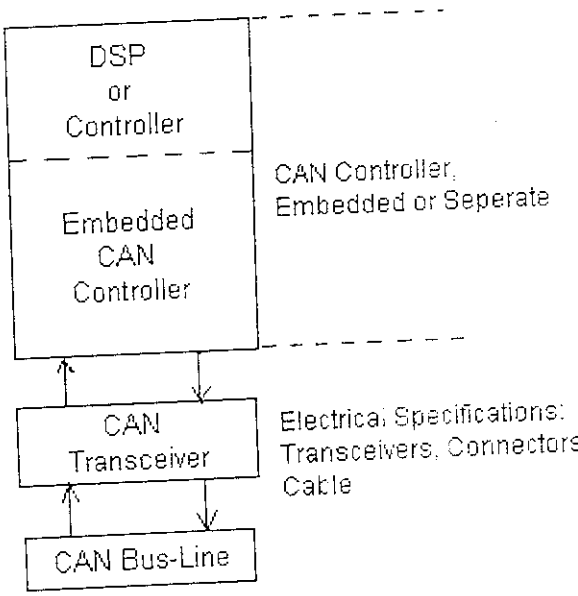
In a CAN network, many short messages like temperature or RPM are broadcast to the entire network.

It conforms to the Open System Interconnect (OSI) model which is defined in terms of layers. CAN architecture defines the lowest two layers of the model, wiz the data link layer and the physical layer.

2.2 THE LAYERED ISO 11898 STANDARD ARCHITECTURE:



ISO 11898 Specification



Implementation

2.3 NEED FOR CAN AND SOME ADVANTAGES

- CAN consists of a twisted pair wire as the bus with appropriate termination.
- It reduces the wiring complexity involved in the cluster.
- It has multi-master capabilities, i.e. all CAN nodes will be able to transmit and receive data
- Several CAN nodes can request the bus simultaneously
- It has a high bit rate and high immunity to electrical interference
- It has an ability to detect any errors produced and also correct them
- Data consistency is preserved in every node of the system
- The serial bus system has real time capabilities and hence is prevalent in the automotive industry
- Internal prioritization is done by the means of 'Local Priority Concept'

- Simplicity in CAN protocol means that very little cost and effort needs to be expended on the personal training
- CAN chip interfaces make applications programming relatively simple
- More than 50 CAN protocol controller chips from more than 15 manufacturers are available

2.4 TYPES OF CAN

CAN exists in two forms –

a) Basic CAN
b) Full CAN

The Basic CAN has a tight coupling between the CPU and the CAN controller, where all messages broadcast on the network have to be individually checked by the microcontroller. This results in limiting the baud rate to 250 kbaud.

A Full CAN is a higher form with an acceptance filter which makes out the irrelevant messages using identifiers. It allows for two lengths of identifiers, one with 11 bits and another with 29 bits.

2.5 CAN PROTOCOL

The CAN communication protocol is a carrier sense multiple access protocol with collision detection and arbitration on message priority (CSMA/CD and AMP)

CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message

CD and AMP means that collisions are resolve through a non destructive bitwise arbitration based upon a pre-programmed priority of each message in the identifier field

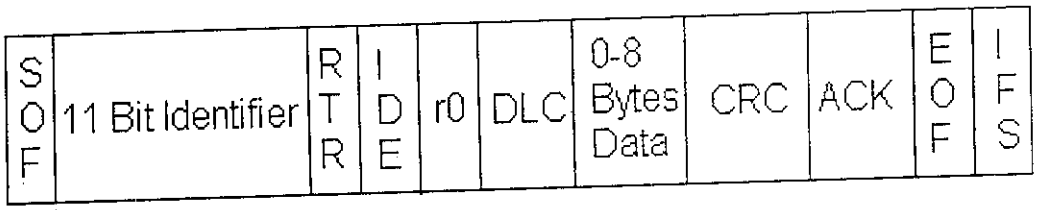
Higher priority wins the access of the bus, an identifier consisting entirely of zeroes gets the highest priority.

There are two versions of CAN –

a) low speed CAN is for applications upto 125 kbps with a standard 11 bit identifier called as standard CAN V2.0A
b) the latest one with signaling rates from 125 kbps to 1 Mbps with extended 29 bit identifier called Extended CAN V2.0A and CAN CAN2.0B

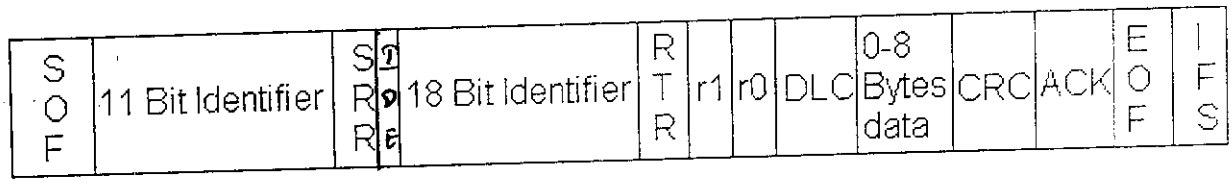
2.6 CAN MESSAGE FORMAT:

STANDARD CAN:



- SOF – Start Of Frame bit marks start of a message and is used for synchronization
- IDENTIFIER – it has 11 bits to establish priority of the message
- RTR – Remote Transmit Request is DOMINANT when information is required from another node
- IDE – Identifier Extension Bit when DOMINANT implies that a standard CAN with no extension is transmitted
- R0 – reserved bit
- DLC – four bit Data Length Code has number of bytes of data being transmitted
- Data – up to 64 bits can be transmitted
- CRC – Cyclic Redundancy Check contains 16 bits that checks sum of data preceding error detection
- ACK – when a node receives a data, it overwrites this RECESSIVE bit by a DOMINANT bit indicating an error free message has been send. It has two bits.
- EOF – End Of Frame is a 7 bit field that marks the end of a CAN message frame and disables bit-stuffing
- IFS – it is a 7 bit Inter Frame Space and contains the amount of time needed by microcontroller to move the message frame into a message buffer

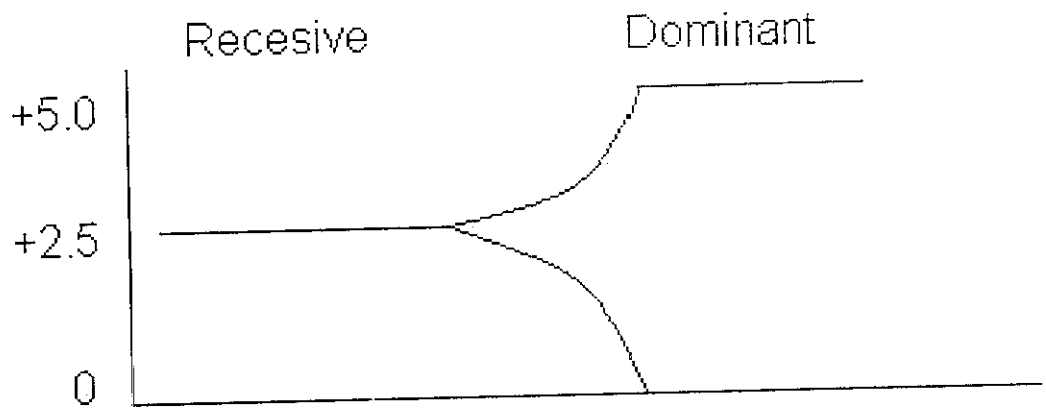
In an **Extended CAN frame**, the changes are:



- SRR – Substitute Remote Request bit replaces RTR
- IDE – a RECESSIVE bit indicates more identifier bits are to follow
- R1 – following RTR & R0, it is an additional RECESSIVE bit

2.7 BUS ARBITRATION:

- ✓ The signaling in CAN is carried out using differential voltages and it is from this that CAN derives most of its noise immunity and fault tolerance
- ✓ The two signal lines, CAN-H & CAN-L in the quiescent state sit at 2.5 V
- ✓ A logic '0' is denoted by CAN-H and is called a DOMINANT bit
- ✓ A logic '1' is denoted by CAN-L and is called a RECESSIVE bit
- ✓ When a logic '1' is written, the two wires sit at 2.5V and is termed as a RECESSIVE bit
- ✓ When a logic '0' is written, one wire is pulled to +5V (CAN-H) and the other is pulled to GND (CAN-L) and is termed as a DOMINANT bit
- ✓ A DOMINANT bit overrules the RECESSIVE bit. This technique is called "Non Destructive Bit-wise Arbitration"



2.8 CAN ERROR CHECKING:

CAN protocol has five methods of error checking – three at message level and two at the bit level. If a message fails anyone of these error detection methods at a given node, it will not be accepted and an error frame will be generated and hence the message will be resent.

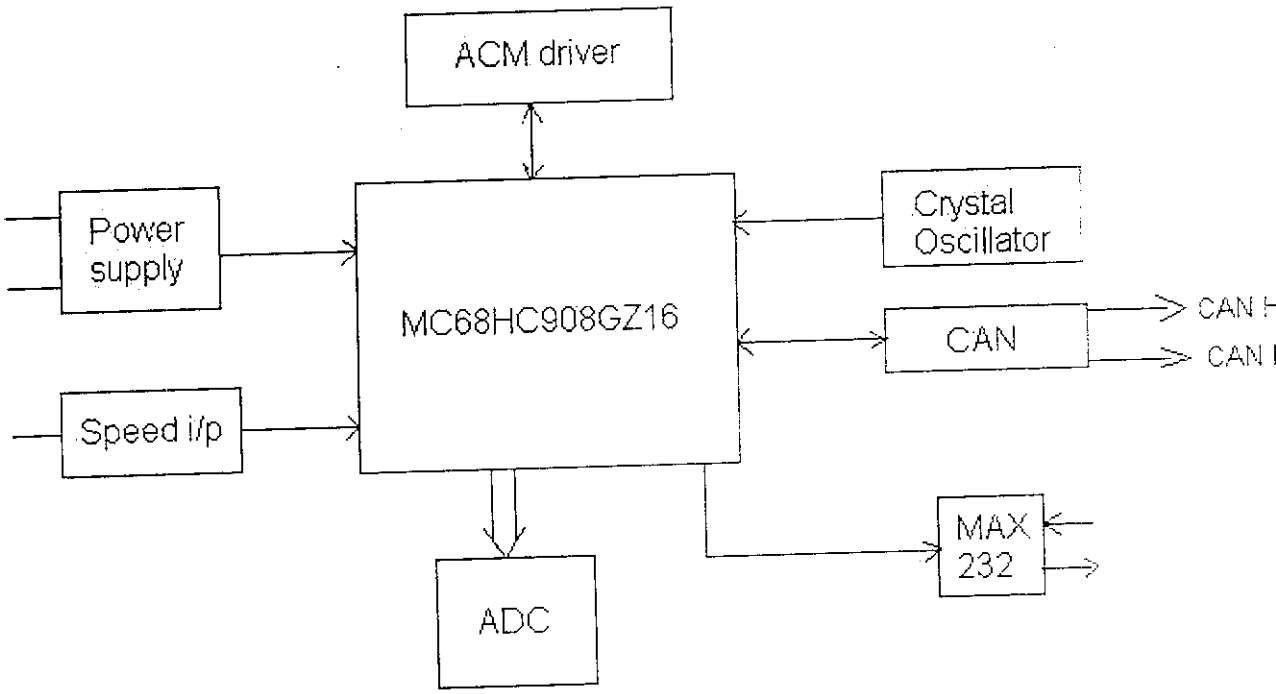
At the message level, there is a CRC check, an Acknowledge slot and a frame check, where CRC is a 15 bit field, ACK is a 2 bit field and a frame check must always have RECESSIVE bits.

At the bit level, each bit is monitored. The compliment of a bit should not be read and the exceptions to this are the identifier field and the acknowledge slot. Finally, bit stuffing rule is when a message is de-stuffed, if after five consecutive bits of the same logic level, the next bit is not a complement, an error is generated.

2.9 CAN APPLICATIONS

- CAN can be used in an embedded communication system for microcontrollers
- It can be used as an open communication system for intelligent devices
- It was originally developed for use in automobiles and is now used in industrial field bus system
- In the field of medical engineering, CAN is opted because the user has to meet with stringent safety requirements
- Also used in manufacture of other equipments such as Robots, lifts and other transportation systems
- Ideally suited in applications which require high reliability and transmission of short messages
- Suited when data is needed by more than one location and system – wide data consistency is needed
- Solutions have been found within CAN in manufacturing automobiles, trucks, trains, buses, airplanes, agriculture, construction, mining and marine vehicles
- It also finds its application in factory and building automation

3 BLOCK DIAGRAM



For our project, we decided to test the CAN based cluster controller by using 'speed' and 'tacho' as the two inputs, inputted through the same circuitry. The controller then controls the ACM driver, providing the necessary deflection. Thus, this is the way we test the transmission and reception by using a CAN controller.

4.2 REALIZATION OF THE DESIGN:

With the completion of the circuit diagram, the process of realization of the design started with the selection of the suitable software for the design of the PCB. This lead to the selection of a software called "CADSTAR".

The first step in this to draw the PCB schematic. This schematic obtained is checked for errors. Errors, if present, are rectified, and we proceed on to the second stage, i.e., PCB design itself. The size of the board is selected and the placement of different chips in the required position is done. The final stage is 'Routing'. With the routing completed, the design obtained is given for fabrication.

Once the PCB is obtained, the board is checked for short circuits and track discontinuity. After this phase, the soldering procedure starts. The components that are to be soldered are checked for proper operation and its characteristics are verified. Then the components are soldered on the board. Once soldering is over, the next step involved is that of loading the software into the EPROM. This procedure consists of erasing the previous contents of the EPROM by exposing the chip to UV rays.

Once the erasure procedure is over, this is then loaded into a universal programmer which loads the necessary program into the EPROM. The PCB is set into the cluster and necessary power supply connections are given. Thus an explanation of the hardware is given above.

5 MICROCONTROLLER

A microcontroller is a device that integrates a number of the components of a microprocessor system onto a single microchip. It consists of powerful CPU tightly coupled with memory (RAM, ROM or EPROM), various I/O features such as serial ports, parallel ports, timers/counters, interrupt controller, data acquisition interfaces, analog to digital converters (ADC), digital to analog converters (DAC) everything integrated into a single silicon chip.

Depending upon the need and area of application for which the chip is designed, the on-chip features present in it may or may not include all the individual sections mentioned above.

A designer will use a microcontroller to:

- Gather input form various sensors
- Process this input into a set of actions
- Use the output mechanisms on the microcontroller to do something useful

Any microcontroller system requires memory to store a sequence of instructions making up a program, parallel port or a serial port for communicating with an external system, timer/counter for generating time delays apart from the controlling unit called the Controller's Central Processing Unit (CPU).

A general purpose microcontroller is a very powerful tool that allows a designer to create a special purpose design. The design becomes partially hardware and partially software. There is a great flexibility in the software end as the designer can create practically unlimited variations on the design.

5.1 ADVANTAGES OF MICROCONTROLLER:

If a system is developed with a microprocessor, the designer will have to go in for external memories such as RAM, ROM/EPROM and peripherals and the size of the PCB will be large enough to hold all the required peripherals. But the microcontroller has all these peripheral facilities on a single chip. So, development of a similar system with a microcontroller reduces the PCB size, cost of the design and number of components.

One of the major differences between the microcontroller and microprocessor is that a controller deals with bits, not with bytes as in the real world applications. For example, switch contacts can only be opened or closed and motors can either be turned ON or OFF.

Microcontroller ports can be used to operate LEDs and relays, as well as input the state of switches and logic circuit inputs. Not all microcontroller ports are able to drive an LED directly; some need to be interfaced via a buffer such as 7406 open collector inverting buffer.

In this project, the microcontroller employed is MC68HC908GZ16.

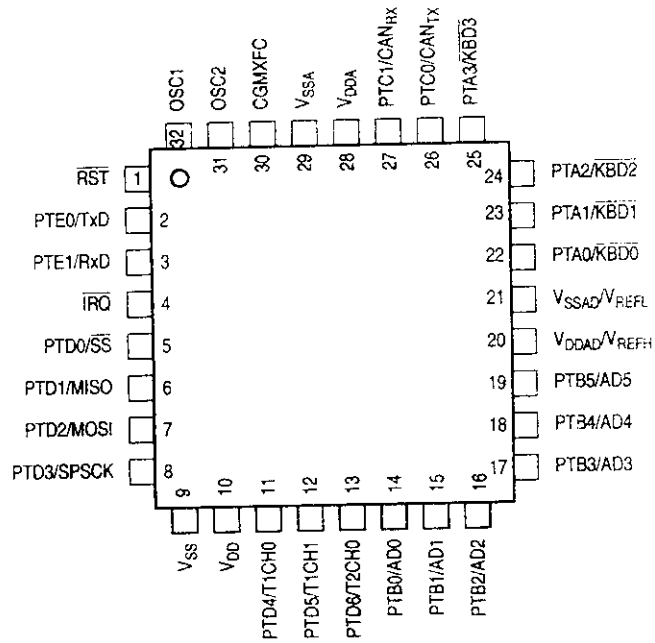


Figure 1-2. 32-Pin LQFP Pin Assignments

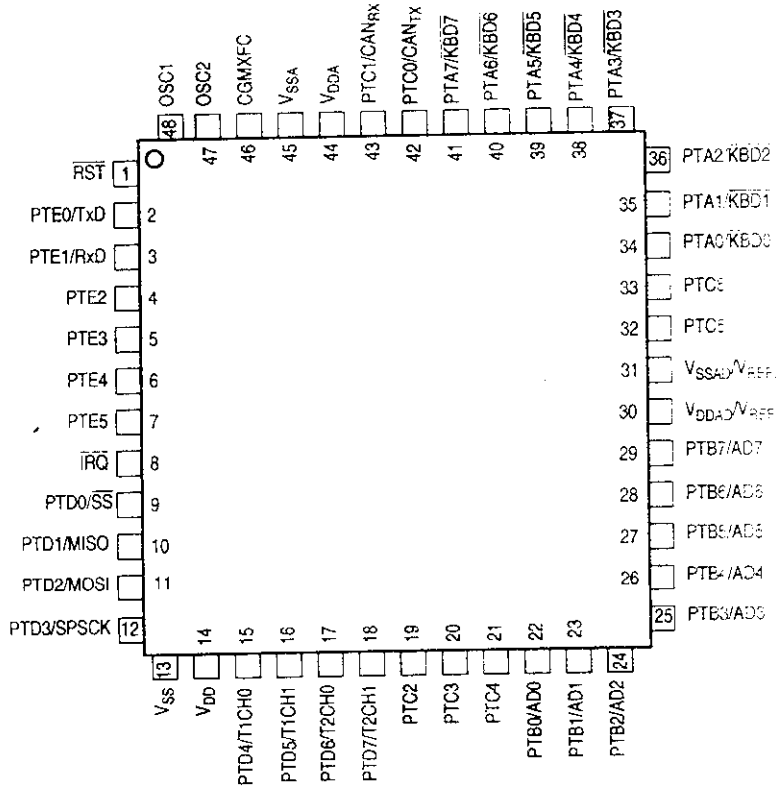


Figure 1-3. 48-Pin LQFP Pin Assignments

5.2 GENERAL FEATURES OF MC68HC908GZ16

The MC68HC908GZ16 is a member of the low cost, high performance M68HC08 family of 8 – bit microcontroller units. This family is based on the Customer – Specified Integrated Circuit (CSIC) design strategy.

All MCUs in the family use the enhanced M68HC08Central Processing Unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

FEATURES

1. High performance M68HC08 architecture
2. Fully upward compatible object code
3. Maximum internal bus frequency -
   8 Mhz at 5V operating voltage
   4 Mhz at 3.3Voperating voltage
4. 32 Khz crystal oscillator clock input with 32 Mhz internal Phase Locked Loop (PLL)
5. 7936 bytes of user flash memory
6. 406 bytes of flash programming routines ROM
7. 44 bytes of user flash vector space
8. 1024 bytes of on chip RAM
9. 350 bytes of monitor ROM
10. 64 bytes of control and status register
11. Two 16 bit, 2 channel Timer Interface Module (TIM1 & TIM2) with selectable input capture, output compare and PWM capability or each channel
12. A Clock Generator Module with 1-8 Mhz oscillator and phase locked loop

13. An enhanced serial communication interface module
14. A serial peripheral interface module
15. IRQ` external interrupt pin with integrated pull up
16. A System Integration Module (SIM) with an external Reset
17. A 10 bit successive approximation Analog to Digital Converter (ADC)
18. 8 bit keyboard wake up port with programmable pull up
19. An MSCAN08 Module which is a unique feature of the controller
20. Power-on Reset
21. A monitor module
22. A 32-pin and 48-pin Low Profile Quad Flat pack
23. Ports – A,B,C,D,E are the five input output ports
    a. PTA0/KBD0` - PTA7/KBD7`
    b. PTB0/AD0 - PTB7/AD7
    c. PTC0/CANtx – PTC1/CANrx
    d. PTC2 – PTC6
    e. PTD0/SS` - PTD7/T2CH1
    f. PTE0/TxD, PTE1/RxD, PTE2 – PTE5
24. System Protection Feature:
    a. Low voltage detection with optional Reset
    b. Computer Operating Properly (COP) reset module
    c. Illegal op code detection with reset
    d. Illegal address detection with reset

CPU

The MC68HC08 CPU is an enhanced and fully object code – compatible version of the M68HC05 CPU

Some of its **important features** are:

It has an object code fully upward compatible with M68HC05 family, and a 16 bit stack pointer with stack manipulation instruction. It has a 16 bit index register with X – register manipulation instructions with 8 Mhz CPU internal bus frequency, 64 Kbyte program/data memory space. There are 16 addressing modes. Memory to Memory data moves without using accumulator is possible. It is built in with a fast 8 bit by 8 bit multiply and 16 bit by 8 bit divide instruction. Enhanced BCD data handling is capable. It also has low power STOP and WAIT modes.

The CPU has an accumulator, stack pointer, index register, program counter, condition code register and ALU.

5.3 MSCAN – MODULE

MSCAN08 CONTROLLER

It is the Motorola Scalable Controller Area Network concept targeted for the Motorola M68HC08 microcontroller family.

The module is a communication controller implementing the CAN 2.0A/B protocol as defined in BOSCH specifications.

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field- real time processing, reliable operation in the electromagnetic interference environment of a vehicle, cost effectiveness and required bandwidth.

It has an advanced buffer arrangement.

FEATURES

- MSCAN08 enable bit is software controlled by bit MSCANEN in configuration register (CONFIG2)
- Modular architecture
- CAN protocol – CAN 2.0A/B
  - ✓ Standard and extended data frames
  - ✓ 0-8 bytes data length
  - ✓ Programmable bit rate up to 1 Mbps
- Support for remote frames
- Double buffered receive storage scheme
- Triple buffered transmit storage scheme with internal prioritization using LOCAL PRIORITY concept
- Flexible maskable identifier filter supports alternatively one full size extended identifier filter or two 16 bit filters or four 8 bit filters
- Programmable wake-up functionality with integrated low pass filter
- Programmable loop-back mode supports self test operation
- Separate signaling and interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer interface module 2 for network synchronization
- Low- power sleep mode

CAN SYSTEM IN MICROCONTROLLER

The MSCAN has two external pins.

- One input- CANrx
- One output- CANtx

- The CANtx pin represents the logic level on the CAN,'0' is for a DOMINANT state and a '1' is for a RECESSIVE state.
- Each CAN station is connected physically to the CAN bus lines through a transceiver chip.
- Transceiver is capable of driving the large current needed for CAN and has current protection against defected stations.

MESSAGE STORAGE

Any CAN node can send out a stream of scheduled messages without releasing the bus.

The nodes will arbitrate for the bus right after sending the previous message.

The internal message queue within the CAN node is organized such that the highest priority message will be sent out first.

A single transmit buffer will not be sufficient because the buffer must be reloaded right after the previous message is send.

A double buffer scheme would decouple the reloading the transmit buffers from the actual message being send.

MSCAN08 has three transmit buffers to make the requirements.

RECEIVE STRUCTURES

- Received messages are stored in a two stage input First-in-First-out (FIFO)
- Has two buffers- a) background receive buffer (RXBG)
  b) foreground receive buffer (RXFG)
- It also has receiver full flag (RXF) which is set if a message is received
- Buffer size is 13 bytes

TRANSMIT STRUCTURES

- There are three transmit buffers
- The transmit buffers are 13 bytes in size
- An additional transmit buffer priority register (TBPR) contains an 8 bit local priority field
- It has a transmitter buffer empty (TXE) flag and the data is transmitted by clearing the flag

IDENTIFIER ACCEPTANCE FILTER

The filter can be programmed to operate in four different modes.

- ✓ Single identifier acceptance filter
- ✓ Two identifier acceptance filter
- ✓ Four identifier acceptance filter
- ✓ Closed filter

INTERRUPTS

MSCAN08 supports four interrupt vectors mapped on to eleven different interrupt sources.

- ✓ Transmit interrupt
- ✓ Receive interrupt
- ✓ Wake-up interrupt
- ✓ Error interrupt (maybe due to over-run, warning or error)

5.4 MONITOR ROM

The monitor ROM allows complete testing of the microcontroller unit (MCU) through a single wire interface with a hot computer.

FEATURES

- Normal user mode pin functionality
- One pin is dedicated to serial communication between monitor Read Only Memory (ROM) and host computer
- Standard baud rate of 7200 @ 8Mhz bus frequency
- FLASH memory security feature
- FLASH memory programming interface
- 350 bytes monitor ROM code size
- It is normal mode entry if high voltage is applied to IRQ`

All the communication between the host computer and the MCU is through the PTA0 pin. PTA0 is used in a wired – OR configuration and requires a pull-up resistor. A normal monitor mode is used here. A security feature discourages unauthorized reading of FLASH locations while in monitor mode.

6.1 POWER SUPPLY

The power supply circuit consists of a dc voltage of +12V to +17V which is given from the battery, a diode which acts as the rectifier and a regulator which regulates the high voltage to a constant value of 5V.

BATTERY VOLTAGE

The dc supply from the battery is about 12V to 17V. Since the microcontroller requires only 5V for its operation, the voltage is regulated to 5V.

RECTIFIERS

A device that is capable of converting the sinusoidal input waveform into unidirectional waveforms with non-zero average component is called rectifier.

The most common single phase rectifiers are –

1. half – wave rectifier
2. full – wave center tapped rectifier
3. full – wave bridge rectifier

The type of rectifier circuit used in this power supply is a half wave rectifier which consists of a single diode D1. The advantage of the half wave rectifier is its simplicity. The diode D1 used here provides reverse polarity protection.

FILTERS

Filters used are basically capacitors. Shunting the load with capacitor frequently affects filtering. The action of this system depends upon the fact that the capacitor stores energy during the conducting period. In this way, the time during which the current passes through the load and the ripple is considerably decreased. The ripple voltage is defined as the deviation for the load voltage from its average or dc value.

There are four capacitors used in this power circuit. All of them act as filters. C1 & C2 capacitors help in removing the spikes in the input, whereas C3 & C4 are also used for protection by removing the spikes in the output supply to the microcontroller.

VOLTAGE REGULATOR

The function of a voltage regulator is to provide a stable dc voltage independent of the load current, temperature and ac voltage variations. It is in this stage that the battery voltage is reduced o a stable dc voltage of 5V.

IC voltage regulators are versatile and inexpensive. The TLE4274 voltage regulator is simple to use. The common pin 2 is grounded. The pin 1 is connected to the cathode of D1. The pin 3 is connected to the 14^{th} pin of the microcontroller which is the power supply Vdd pin.

6.2 SPEED AND TACHO SENSORS:

There are two inputs, speed and tacho in RPM that are tested in our CAN based cluster controller. A pulse width modulated wave is given as the speed input. The are two resistors, R2 & R3 which are used to limit and provide resistance to the input.

The capacitor C5 removes the spikes in the speed input and hence acts as a filter. There is a zener diode which allows only 9V or lesser input to pass through. There is another diode D2 4148 with MELF package which acts as a rectifier. The transistor BC547 used here acts as a switch. When a high voltage comes through the base, the transistor goes to the ON state and hence is pulled to a high. But, when a low voltage is given to the input, the transistor is shorted. At every falling edge of the input, the transistor goes to the OFF state. Every edge is taken as a trigger and transistor goes to the ON state. The transistor has a capacitor C7 connected across it and hence connected to the 15^{th} pin of the microcontroller. This pin is the port D4 or the timer channel 0 pin. The timer hence senses the number of pulses and calculates the distance covered and hence the speed.

6.3 DIFFERENTIAL AIR CORE METER DRIVER

The SA5775 is a monolithic Serial Gauge Driver (SGD) which can be used to directly drive Air Core Meters (ACM), typically used in automobile dashboards. The circuit interfaces with the serial bus of a microcontroller, has 10 bit resolution (0.35 deg) and is guaranteed to be monotonic. Data can be shifted through the part so that several SA5775s can be cascaded with only one chip select line.

PRINCIPLE

In many meter applications hitherto D'Arsonval type of movement has been used. In this, a field coil is used along with a permanent magnet. When a current passes through the field coil, it generates a magnetic field. The magnetic field, due to the field coil, interacts with that due to the permanent magnet producing a resultant force that causes a needle to deflect, the deflection being controlled by a spring. Calibration of the current results in known deflections of the needle is used to indicate the desired reading. There are a number of disadvantages here.

a. The strength of the mechanical spring degrades with age and with temperature cycling
b. The permanent magnet ages with a lessening of the magnetic field. This makes it harder for it to pull the needle against the spring.
c. The mechanism is prone to damage due to vibrations from handling and other causes

To overcome some of these problems, Air Core Meters have been used quite successfully. In this type of movement, there are two coils which are

wound at 90 deg from one another. A needle is attached to a magnetized disc that is positioned between the two coils. When currents 90 deg out of phase with each other pass through the two coils, the resultant magnetic fields produce a force which moves the disc. The position of the disc is dependent upon the ratio of the currents in the two coils. By varying this ratio, the disc and hence the needle position can be varied.

ADVANTAGES

a. There are no springs
b. Only effect of magnetic strength degradation with age is in slowing of mechanical response time
c. Accuracy does not degrade with age
d. Not prone to damage due to vibration/handling on account of rugged sleeve bearings
e. Built-in damping

FEATURES OF THE SA5775 ACM DRIVER

1) 10 bit resolution (0.35 deg)
2) exceptional accuracy (0.25 deg, typically)
3) high torque capability
4) active differential drivers eliminate back-EMF issues
5) no RFI/EMI generation issues
6) simple serial interface
7) simple cascading capability for multiple meters
8) internal fault protection
9) only one external component required (bypass capacity)

APPLICATION

Instrumentation utilizing Air Core Meters.

6.4 SN75LBC031 – HIGH SPEED CONTROLLER AREA NETWORK (CAN) TRANSCEIVER:

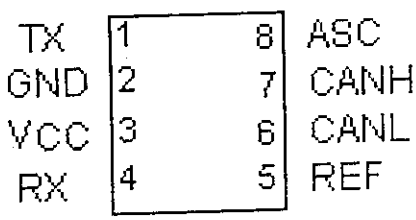
The SN75LBC031 is transceiver used as an interface between a CAN controller and the physical bus for high speed applications of up to 500 kbaud. The device provides transmit capability to the differential bus and differential receive capability to the controller.

The transmitter outputs (CAN-H, CAN-L), feature internal transition regulation to provide controlled symmetry resulting in low EMI emissions. Both the transmitter outputs are fully protected against battery short circuits and electrical transients that can occur on the bus lines. There is a thermal shut down circuitry to disable the output drivers in the event of a excessive power dissipation. For normal operation at 500 kbaud, the ASC terminal is open or tied to GND.

The receiver includes an integrated filter that suppresses the signal into pulses less than 30 ns wide. The SN75LBC031 is characterized for operation from -40 ° C to 85 ° C.

SN75LBC031 PIN DIAGRAM

D PACKAGE
(TOP VIEW)



TERMINAL FUNCTIONS DESCRIPTION

TX	-	Transmitter input
GND	-	Ground
RX	-	Receiver output
VCC	-	Supply voltage
REF	-	Reference output
CANL	-	Low side bus output driver
CANH	-	High side bus output driver
ASC	-	Adjustable slope control

6.5 CRYSTAL OSCILLATOR

The crystal oscillator module provides the reference clock for the Clock Generator Module (CGM), the Real Time Clock module (RTC) and other MCU sub systems. The oscillator module consists of two types of oscillator circuits-

1. internal RC oscillator
2. Crystal oscillator (X-tal)

The two oscillator pins OSC1 and OSC2 of the micro controller are the connections for an external crystal, resonator, or clock circuit. The internal RC clock is the output from the internal RC oscillator. This clock drives the SIM and COP modules.

The WAIT and STOP instructions put the MCU in the low power stand by mode. The WAIT instruction has no effect on the oscillator module. The CGMXFC is the external filter capacitor connection for the clock generator module.

6.6 MAX 232

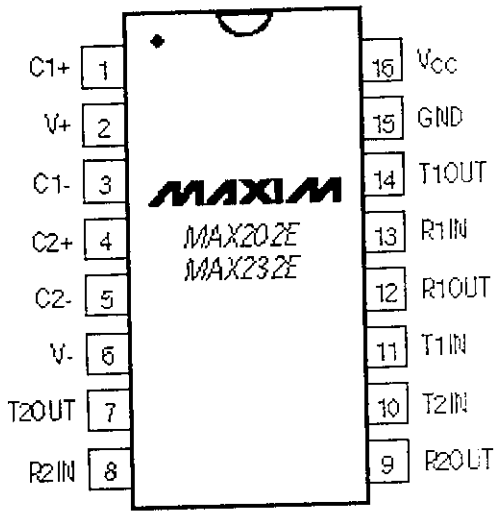
FEATURES

- High data rate – 250 kbits/sec under load
- 16 – pin DIP or SOIC package
- 20 – pin TSSOP package for height restricted applications
- Operate from single +5V power
- Meets all EIA-232E and V0.28 specifications
- Uses small capacitors 0.1μF
- Optional industrial temperature range available (-40°C to +85°C)

DESCRIPTION

The MAX232 is a dual RS-232 driver/receiver pair that generates RS-232 voltage levels from a single +5V power supply. Additional +12V supplies are not needed since the MAX232 uses on-board charge pumps to convert the +5V supply to +10V. The MAX232 is fully compliant with EIA RS-232E and V0.28/V0.24 standards. The MAX232 contains two drivers and two receivers. Driver slew rates and data rates are guaranteed up to 250 kbits/sec. The MAX232 operates with only one 0.1μF charge pump capacitors.

PIN CONFIGURATION



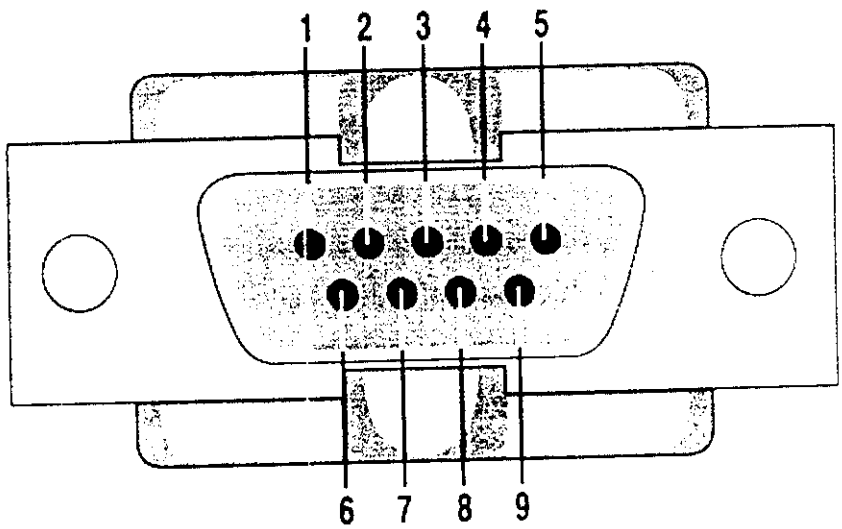
PIN DESCRIPTION

VCC	-	+5 Volt Supply
GND	-	Ground
V+	-	Positive Supply Output
V-	-	Negative Supply Output
T1IN, T2IN	-	RS-232 Driver Inputs
T1OUT, T2OUT	-	RS-232 Driver Outputs
R1IN, R2IN	-	Receiver Inputs
R1OUT, R2OUT	-	Receiver Outputs
C1+, C1-	-	Capacitor 1 Connections
C2+, C2-	-	Capacitor 2 Connections

6.6.1 RS-232

RS-232 stands for Recommended Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. It sends and receives data using twisted pair cable. The full RS-232C standard specifies a 25-pin 'D' connector of which 22 pins are used. Most of these pins are not needed for normal PC communications and indeed, most new PCs are equipped with male 'D' type connectors having only nine pins.

PIN SCHEMATIC



PIN DESCRIPTION

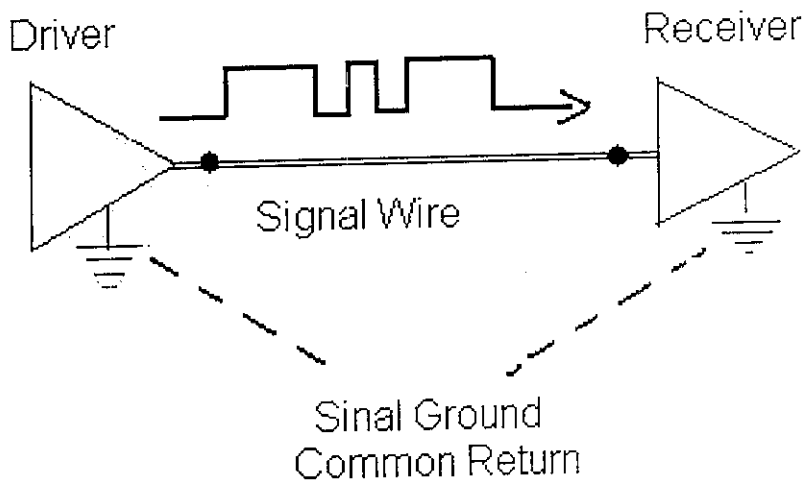
Pin Number	Direction of Signal
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request to Send (RTS) Outgoing flow control signal
8	Clear to Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

RS-232 uses an unbalanced signal communication method. That is, there is one signal wire for each circuit with a common return for all signals. This method is somewhat susceptible to electrical noise.

To communicate, the device sends a series of binary signals to the receiver. These binary pulses make up predefined words that indicate either commands or status conditions.

The RS-232 communication standard supports only one transmitter driver and a receiver. Distance is limited to 50 feet.

FLOW OF SIGNAL FROM DRIVER TO RECEIVER

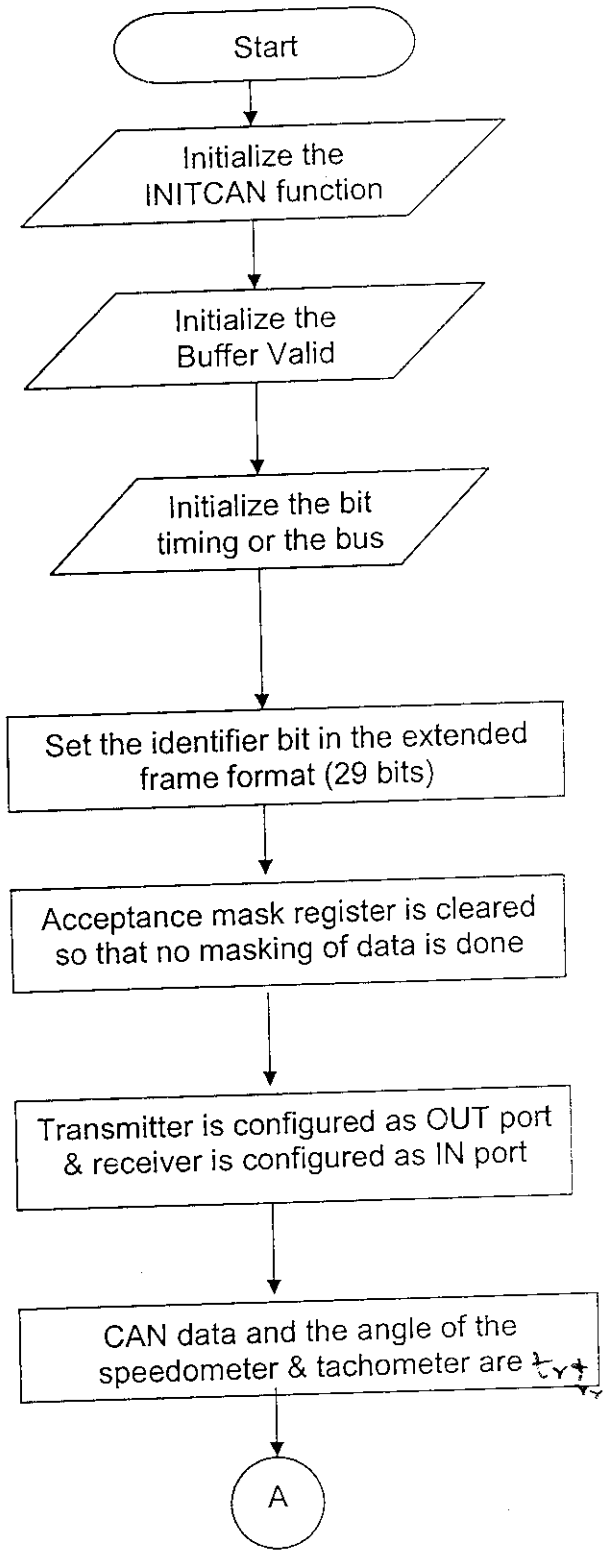


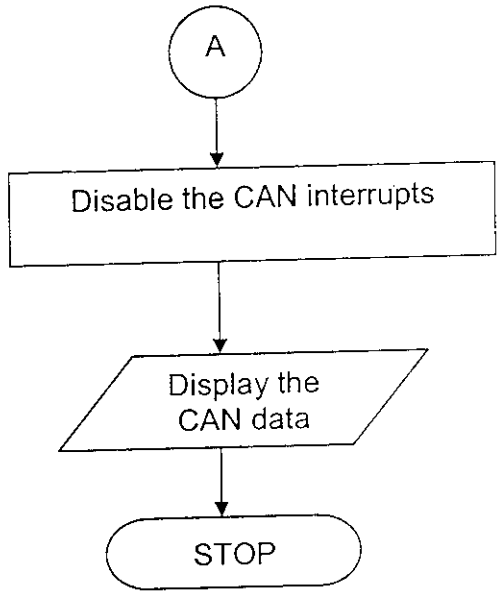
7.1 PROGRAMMING ASPECTS

'C' – An Insight

The software was programmed in Motorola's 'C' compiler. 'C' was preferred as a development medium due to its right popularity, ease of use as compared to assembly language, good programming constructs, rich reference language, rich reference sources, high performance and it offers very good hardware interaction due to its close relationship with assembly language. Summing it up, it offers the best of both low level assembly language, by providing raw programming power of assembly language and the understandable programming style of high level languages. By suitably changing the compiler options, the same 'C' code can be easily ported to different microcontrollers by making little or no changes in source code.

7.2 FLOWCHART




A
Disable the CAN interrupts
Display the
CAN data
STOP

7.3 ALGORITHM

TRANSMISSION

1. Start the program
2. Initialize the function INITCAN in which all CAN registers are initialized
3. Configure the message Buffer Valid Register (BVALR) invalid
4. Configure the Identifier bits as the extended frame version with 29 bits. It is initialized as XXFF.
5. The first buffer out of the 16 buffers is configured for transmission
6. The acceptance mask selection register is cleared so that no masking of data is done
7. The control status register is configured as 0081 where the transmit buffer is the OUT port and the receiver buffer is the IN port
8. The message buffer is made valid
9. The Transmission Request Register (TREQR), Transmission Cancel Register (TCANR), Transmission Termination Register (TCR) are cleared
10. The speedometer, tachometer values are transmitted
11. The fuel angle is also transmitted in which only the first four bytes, two for speedometer and two for tachometer

12. The control status register is configured as 0080

13. The Transmission Request Register (TREQR) is set

RECEPTION

1. Start the program

2. Initialize the function INITCAN in which all CAN registers are initialized

3. Configure the message Buffer Valid Register (BVALR) invalid

4. Configure the Identifier bits as the extended frame version with 29 bits. It is initialized as XXFF.

5. The first buffer out of the 16 buffers is configured for reception

6. The acceptance mask selection register is cleared so that no masking of data is done

7. The control status register is configured as 0081 where the transmit buffer is the OUT port and the receiver buffer is the IN port

8. The message buffer is made valid

9. The Transmission Request Register (TREQR), Transmission Cancel Register (TCANR), Receiving Termination Register (RCR), Receiving Over Run Register (ROVRR) and Receiving Interrupt Permission Register (RIER) are cleared

10. The speedometer, tachometer values are received

11. The fuel angle is also received in which only the first four bytes, two for speedometer and two for tachometer

12. A formula is used to convert the angle into the value to be displayed

13. The Receiving Termination Register (RCR) is configured as XXFF

14. The Transmission Request Register (TREQR) is set

7.4 PROGRAM CODING

********** CAN TRANSMISSION BOARD **********

```
void InitCan(void)
{
        REG_WRITE(IO_BVALR.word, (U2)0);            /* message buffer invalid */
        REG_WRITE(IO_CANCT.BTR.word, (U2)0x7FFF);    /*  bit timing  */
        REG_WRITE(IO_CANCT.IDER.word, (U2)0xFFFF); /* IDE (Extended
frame)*/
        for(i = 0; i < 16; i++)
        {
                REG_WRITE(IO_CANID.IDR[i].lword, (U4)0);   /* ID */
                REG_WRITE(IO_CANID.DLCR[i].word, (U2)8);/* DLC */
                REG_WRITE(IO_CANID.DTR[i].BYTE0_3, (U4)0);      /* DTR */
                REG_WRITE(IO_CANID.DTR[i].BYTE4_7, (U4)0);      /* DTR */
        }
        REG_WRITE(IO_CANID.IDR[0].lword,(U1) 0xff); /* Identifier = 0xff  */
        REG_WRITE(IO_CANCT.AMSR.lword, (U4)0);  /* acceptance mask
selection register    */
        REG_WRITE(IO_BVALR.word, (U2)1);    /* message buffer valid      */
        REG_WRITE(IO_CANCT.TRTRR.word, (U2)0); /* Send data frame  */
        REG_WRITE(IO_CANCT.CSR.word, (U2)0x0081);     /* control status */

        REG_WRITE(IO_TREQR.word, (U2)0);  /* transmission request register */
        REG_WRITE(IO_TCANR.word, (U2)0);  /* transmission cancel register  */
        REG_WRITE(IO_TCR.word, (U2)0);      /* ????transmission termination
register  */
        REG_WRITE(IO_RCR.word, (U2)0);       /* receiving termination register */
        REG_WRITE(IO_RRTRR.word, (U2)0);            /* remote request receiving
register */
        REG_WRITE(IO_ROVRR.word, (U2)0); /* receiving overran register   */
        REG_WRITE(IO_RIER.word, (U2)0);      /* receiving interrupt permission
register */

        REG_WRITE(IO_CANCT.LEIR.word, (U2)0);   /* last event display */
        REG_WRITE(IO_CANCT.TRTRR.word, (U2)0);  /* transmission RTR */
        REG_WRITE(IO_CANCT.RFWTR.word, (U2)0xFFFF); /* remote frame */
        REG_WRITE(IO_CANCT.TIER.word, (U2)0);   /* transmission interrupt
permission    */
        REG_WRITE(IO_CANCT.AMR0.lword, (U4)0); /* acceptance mask 0 */
        REG_WRITE(IO_CANCT.AMR1.lword, (U4)0);  /* acceptance mask 1*/
}
```

```
static void CanRoutine(void)
{
        U2 u2_spd_angle;
        U2 u2_tacho_angle;
        U2 u2_fuel_angle;
        U2 u2_temp_angle;
        U4 CanData;
        u2_spd_angle = u2_GetSpdAngle();
        u2_tacho_angle = u2_GetTachoAngle();
        CanData = (U4) u2_spd_angle | (U4) ((U4) u2_tacho_angle << 16);
        REG_WRITE(IO_CANID.DTR[0].BYTE0_3,CanData);
        REG_WRITE(IO_CANCT.RFWTR.word, (U2)0x0);     /* remote frame */
        REG_WRITE(IO_CANCT.CSR.word, (U2)0x0080);    /* control status */
        REG_WRITE(IO_TREQR.word, (U2)1); /* transmission request register */
}

*****************************************************************************

********************************* CAN RECEPTION BOARD **************************

void InitCan(void)
{
        REG_WRITE(IO_BVALR.word, (U2)0);          /* message buffer invalid */
        REG_WRITE(IO_CANCT.BTR.word, (U2)0x7FFF);   /* bit timing   */
        REG_WRITE(IO_CANCT.IDER.word, (U2)0xFFFF);  /* IDE (Extended
frame)*/
        for(i = 0; i < 16; i++)
        {
                REG_WRITE(IO_CANID.IDR[i].lword, (U4)0);  /* ID   */
                REG_WRITE(IO_CANID.DLCR[i].word, (U2)8);/* DLC */
                REG_WRITE(IO_CANID.DTR[i].BYTE0_3, (U4)0);     /* DTR */
                REG_WRITE(IO_CANID.DTR[i].BYTE4_7, (U4)0);     /* DTR */
        }
        REG_WRITE(IO_CANID.IDR[0].lword,(U1) 0xff);       /* Identifier = 0xff*/
        REG_WRITE(IO_CANCT.AMSR.lword, (U4)0);            /* acceptance mask
selection register    */
        REG_WRITE(IO_BVALR.word, (U2)1);          /* message buffer valid   */
        REG_WRITE(IO_CANCT.TRTRR.word, (U2)0); /* Send data frame */
        REG_WRITE(IO_CANCT.CSR.word, (U2)0x0081);    /* control status */

        REG_WRITE(IO_TREQR.word, (U2)0);          /* transmission request
register */
        REG_WRITE(IO_TCANR.word, (U2)0);          /* transmission cancel
register */
```

```
        REG_WRITE(IO_TCR.word, (U2)0);      /* ????transmission termination
register */
        REG_WRITE(IO_RCR.word, (U2)0);      /* receiving termination register */
        REG_WRITE(IO_RRTRR.word, (U2)0); /* remote request receiving
register */
        REG_WRITE(IO_ROVRR.word, (U2)0); /* receiving overran register   */
        REG_WRITE(IO_RIER.word, (U2)0);             /* receiving interrupt
permission register */

        REG_WRITE(IO_CANCT.LEIR.word, (U2)0);   /* last event display */
        REG_WRITE(IO_CANCT.TRTRR.word, (U2)0); /* transmission RTR */
        REG_WRITE(IO_CANCT.RFWTR.word, (U2)0xFFFF);/* remote frame */
        REG_WRITE(IO_CANCT.TIER.word, (U2)0);   /* transmission interrupt
permission   */
        REG_WRITE(IO_CANCT.AMR0.lword, (U4)0); /* acceptance mask 0 */
        REG_WRITE(IO_CANCT.AMR1.lword, (U4)0); /* acceptance mask 1 */
}


static void CanRoutine(void)
{
        U4 u4_candata;
        U2 u2_Spd;
        U2 u2_tacho;
        REG_WRITE(IO_CANCT.CSR.word, (U2)0x0080);     /* control status */
        if(REG_READ(IO_RCR.bit.RC0) == ON)
        {
                u4_candata = REG_READ(IO_CANID.DTR[0].BYTE0_3);
                u2_Spd = (U2) (u4_candata & (U4) 0xffff);
                u2_tacho = (U2) (u4_candata >> 16);
                SpdMtrCanUpdate(u2_Spd);
                TachoCanUpdate(u2_tacho);
                u2_Spd -= (U2) 1800;
                SetDebugDisp(0, (U4) u2_Spd);
                SetDebugDisp(1, (U4) ((U4) 2000 * (U4) u2_Spd) / (U4) ((U4)
 32894-(U4) 1800));
                REG_WRITE(IO_RCR.bit.RC0, OFF);
        }
}
*****************************************************************************************
```

8 CONCLUSION

"*Necessity is the mother of invention*" – the saying goes...
The need to transmit data in a more convenient manner and to restructure the established standards of transmission by using just a twisted pair wire led to the implementation of this project.

This project provides the above stated feature by introducing the concept of Controller Area Network – CAN Bus in the automobile cluster. The CAN protocol with greater advantages compared to USB or Ethernet has a higher bit rate, immunity to electrical interference and data consistency.

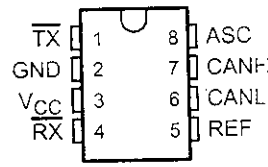
This project can be further developed by the inclusion of additional devices, interfacing them with the CAN bus and thus making the automobile cluster compatible with the various standards followed for communication by the vehicle manufacturers.

SN65LBC031, SN65LBC031Q, SN75LBC031
HIGH-SPEED CONTROLLER AREA NETWORK (CAN) TRANSCEIVERS

SLRS048A – MAY 1998 – REVISED APRIL 2000

- **SN75LBC031 Meets Standard ISO/DIS 11898 (up to 500 k Baud)**
- **Driver Output Capability at 50 mA**
- **Wide Positive and Negative Input/output Bus Voltage Range**
- **Bus Outputs Short-Circuit-Protected to Battery Voltage and Ground**
- **Thermal Shutdown**
- **Available in Q-Temp Automotive**
  - **HighRel Automotive Applications**
  - **Configuration Control/Print Support**
  - **Qualification to Automotive Standards**

D PACKAGE
(TOP VIEW)



TERMINAL FUNCTIONS

TERMINAL	DESCRIPTION
$\overline{TX}$	Transmitter input
GND	Ground
V_{CC}	Supply voltage
$\overline{RX}$	Receiver output
REF	Reference output
CANL	Low side bus output driver
CANH	High side bus output driver
ASC	Adjustable slope control

FUNCTION TABLE

$\overline{TX}$	CANH	CANL	BUS STATE	$\overline{RX}$
L	H	L	Dominant	L
High or floating	Floating	Floating	Recessive	H

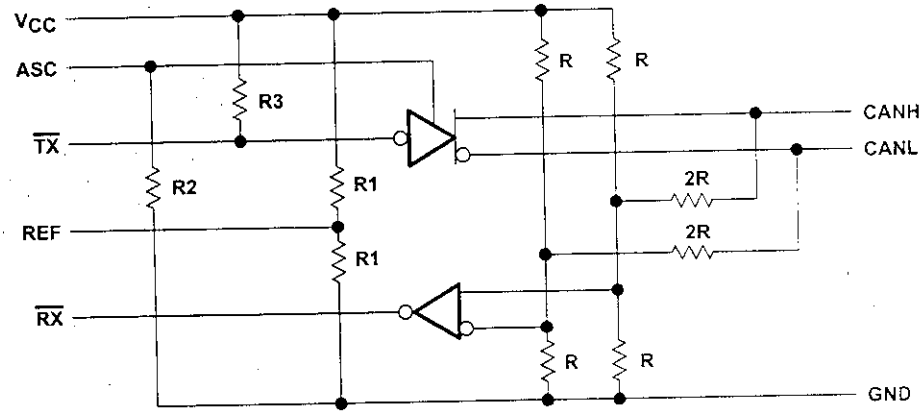
L = low, H = high

description

The SN75LBC031 is a CAN transceiver used as an interface between a CAN controller and the physical bus for high speed applications of up to 500 kBaud. The device provides transmit capability to the differential bus and differential receive capability to the controller. The transmitter outputs (CANH and CANL), feature internal transition regulation to provide controlled symmetry resulting in low EMI emissions. Both transmitter outputs are fully protected against battery short circuits and electrical transients that can occur on the bus lines. In the event of excessive device power dissipation the output drivers are disabled by the thermal shutdown circuitry at a junction temperature of approximately 160°C. The inclusion of an internal pullup resistor on the transmitter input ensures a defined output during power up and protocol controller reset. For normal operation at 500 kBaud the ASC terminal is open or tied to GND. For slower speed operation at 125 kBaud the bus output transition times can be increased to reduce EMI by connecting the ASC terminal to V_{CC}. The receiver includes an integrated filter that suppresses the signal into pulses less than 30 ns wide.

The SN75LBC031 is characterized for operation from –40°C to 85°C. The SN65LBC031 is characterized for operation from –40°C to 125°C. The SN65LBC031Q is characterized for operation over the automotive temperature range of –40°C to 125°C.

logic diagram



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Logic supply voltage, V_{CC} (see Note 1)	7 V
Bus terminal voltage	−5 V to 20 V
Input current at $\overline{TX}$ and ASC terminal, I_I	±10 mA
Input voltage at $\overline{TX}$ and ASC terminal, V_I	$2 \times V_{CC}$
Operating free-air temperature range, T_A: SN65LBC031, SN65LBC031Q	−40°C to125°C
SN75LBC031	−40°C to 85°C
Operating juncation range, T_J	−40°C to 150°C
Continuous total power dissipation at (or below) 25°C free-air temperature	See Dissipation Rating Table
Storage temperature range, T_{stg}	−65°C to 150°C
Case temperature for 10 sec T_C, D package	260°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values, except differential bus voltage, are measured with respect to GND.

DISSIPATION RATING TABLE

PACKAGE	$T_A \leq 25°C$ POWER RATING	OPERATING FACTOR ABOVE $T_C = 25°C$	$T_C = 125°C$ POWER RATING
D	725 mW	5.8 mW/°C	145 mW

DISSIPATION DERATING CURVE vs FREE-AIR TEMPERATURE

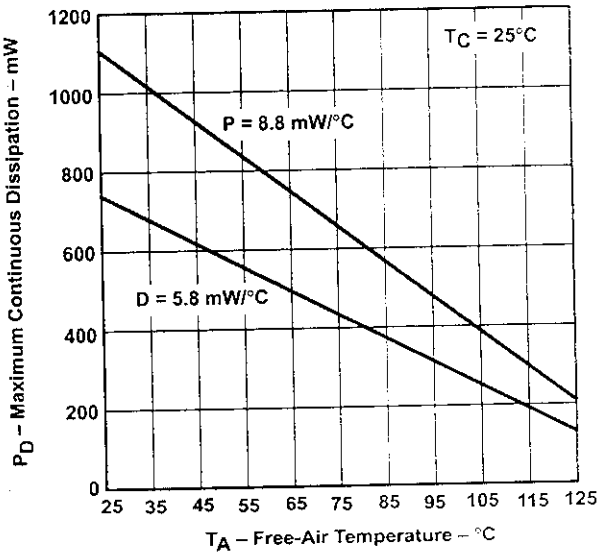


Figure 1

recommended operating conditions

		MIN	NOM	MAX	UNIT
Logic supply voltage, V_{CC}		4.5	5	5.5	V
Voltage at any bus terminal (separately or common mode), V_I or V_{IC} (see Note 3)		–2		7	V
High-level input voltage, V_{IH}	$\overline{TX}$	2		V_{CC}	V
Low-level input voltage, V_{IL}	$\overline{TX}$	0		0.8	V
High-level output current, I_{OH}	Transmitter			–50	mA
	Receiver			–400	µA
Low-level output current, I_{OL}	Transmitter			50	mA
	Receiver			1	
Operating free-air temperature, T_A	SN75LBC031	–40		85	°C
	SN65LBC031, SN65LBC031Q	–40		125	

NOTES: 2. All voltage values, except differential bus voltage, are measured with respect to the ground terminal.
3. For bus voltages from –5 V to –2 V and 7 V to 20 V the receiver output is stable.

SYMBOL DEFINITION

DATA SHEET PARAMETER	DEFINITION
$V_{O(CANHR)}$	CANH bus output voltage (recessive state)
$V_{O(CANLR)}$	CANL bus output voltage (recessive state)
$V_{O(CANHD)}$	CANH bus output voltage (dominant state)
$V_{O(CANLD)}$	CANL bus output voltage (dominant state)
$V_{O(DIFFR)}$	Bus differential output voltage (recessive state)
$V_{O(DIFFD)}$	Bus differential output voltage (dominant state)
$V_{I(ASC)}$	Adjustable slope control input voltage

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$V_{O(REF)}$	Reference source output voltage	I_{REF} = ±20 µA	0.45V_{CC}		0.55V_{CC}	V
$R_{O(REF)}$	Reference source output resistance		5		10	kΩ
$I_{CC(REC)}$	Logic supply current, recessive state	See Figure 2. S1 closed		12	20	mA
$I_{CC(DOM)}$	Logic supply current, dominant state			55	80	

transmitter electrical characteristics over recommended ranges of supply and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$V_{O(CANHR)}$ $V_{O(CANLR)}$	Output voltage (recessive state)	See Figure 2, S1 open	2	0.5V_{CC}	3	V
$V_{O(DIFFR)}$	Differential output voltage (recessive state)		–500	0	50	mV
$V_{O(CANHD)}$	Output voltage (dominant state)	See Figure 2, S1 closed	2.75	3.5	4.5	V
$V_{O(CANLD)}$	Output voltage (dominant state)		0.5	1.5	2.25	
$V_{O(DIFFD)}$	Differential output voltage (dominant state)		1.5	2	3	
$I_{IH(TX)}$	High-level input current (TX)	V_{IH} = 2.4 V		–100	–135	µA
		V_{IH} = V_{CC}			±2	
$I_{IH(ASC)}$	High-level input current (ASC)	V_{IH} = 2.4 V		100	165	µA
		V_{IH} = V_{CC}		200	340	
$I_{IL(TX)}$	Low-level input current ($\overline{TX}$)	V_{IL} = 0.4 V		–180	–400	µA
$I_{IL(ASC)}$	Low-level input current (ASC)	V_{IL} = 0.4 V		15	25	µA
$C_{I(TX)}$	$\overline{TX}$ input capacitance			8		pF
$I_{O(ssH)}$	CANH short circuit output current	$V_{O(CANH)}$ = –2 V to 20 V		–95	–200	mA
$I_{O(ssL)}$	CANL short circuit output current	$V_{O(CANL)}$ = 20 V to –2 V		140	250	mA

NOTE 2: All voltage values, except differential bus voltage, are measured with respect to the ground terminal.

transceiver dynamic characteristics over recommended operating free-air temperature range and V_{CC} = 5 V

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{(loop)}$	Loop time	See Figures 2 and 3, S1 closed, $V_{I(ASC)}$ = 0 V or open circuit, S2 open			280	ns
		See Figures 2 and 3, S1 closed, $V_{I(ASC)}$ = V_{CC}, S2 closed			430	ns
$SR_{(RD)}$	Differential-output slew rate (recessive to dominant)	See Figures 2 and 4, S1 closed, $V_{I(ASC)}$ = 0 or open circuit, S2 open		35		V/µs
		See Figures 2 and 4, S1 closed, $V_{I(ASC)}$ = V_{CC}, S2 closed		10		V/µs
$SR_{(DR)}$	Differential-output slew rate (dominant to recessive)	See Figures 2 and 4, S1 closed, $V_{I(ASC)}$ = 0 or open circuit, S2 open		10		V/µs
		See Figures 2 and 4, S1 closed, $V_{I(ASC)}$ = V_{CC}, S2 closed		10		V/µs
$t_{d(RD)}$	Differential-output delay time	See Figure 2, S1 closed		55		ns
$t_{d(DR)}$				160		ns
$t_{pd(RECRD)}$	Receiver propagation delay time	See Figures 2 and 5		90		ns
$t_{pd(RECDR)}$				55		ns

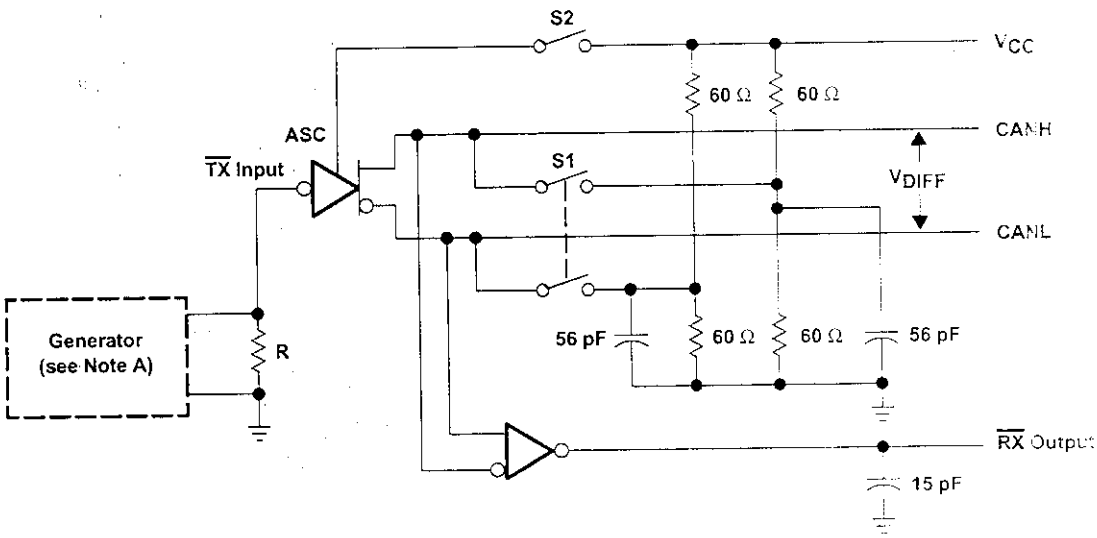
NOTE 4: Receiver input pulse width should be >50 ns. Input pulses of <30 ns are suppressed.

receiver electrical characteristics over recommended ranges of common-mode input voltage, supply voltage, and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$V_{IT(REC)}$	Differential input threshold voltage for recessive state	V_{IC} = –2 V to 7 V			500	mV
$V_{IT(DOM)}$	Differential input threshold voltage for dominant state		900			
V_{hys}	Recessive-dominant input hysteresis		100	180		mV
$V_{OH(RX)}$	High-level output voltage	$V_{O(DIFF)}$ = 500 mV, I_{OH} = –400 µA	V_{CC}–0.5 V		V_{CC}	V
$V_{OL(RX)}$	Low-level output voltage	$V_{O(DIFF)}$ = 900 mV, I_{OL} = 1 mA	0		0.5	V
$r_{I(REC)}$	CANH and CANL input resistance in recessive state	dc, no load	5		50	kΩ
$r_{I(DIFF)}$	Differential CANH and CANL input resistance in recessive state	dc, no load	10		100	kΩ
C_i	CANH and CANL input capacitance			20		pF
$C_{i(DHL)}$	Differential CANH and CANL input capacitance			10		pF

NOTE 2: All voltage values, except differential bus voltage, are measured with respect to the ground terminal.

PARAMETER MEASUREMENT INFORMATION



NOTE A: The input pulse is supplied to $\overline{TX}$ by a generator having a t_r and t_f = 5 ns.

Figure 2. Test Circuit

PARAMETER MEASUREMENT INFORMATION

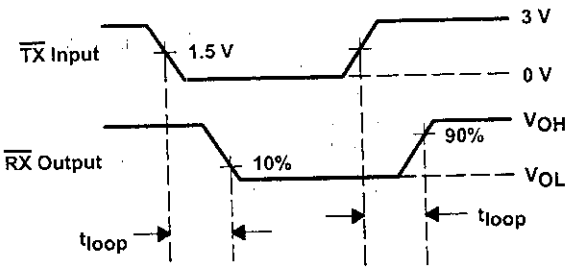


Figure 3. Loop Time

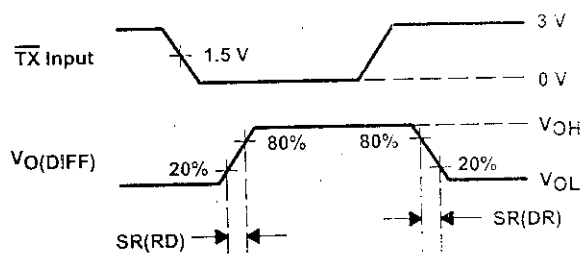
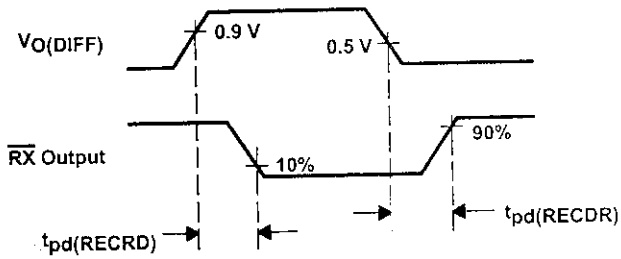


Figure 4. Slew Rate

NOTE A: The input pulse is supplied to $\overline{TX}$ by a generator having a t_r and t_f = 5 ns.



NOTE A: The input pulse is supplied as V_{DIFF} using CANH and CANL respectively by a generator having a t_r and t_f = 5 ns.

Figure 5. Receiver Delay Times

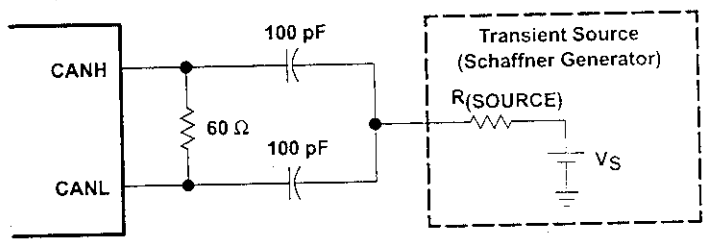


Figure 6. Transient Stress Capability Test Circuit

PARAMETER MEASUREMENT INFORMATION

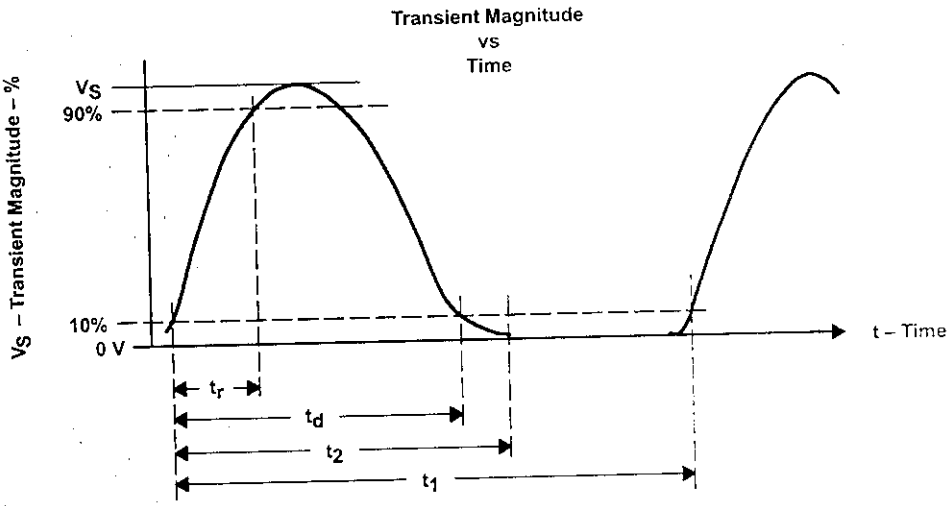


Figure 7. Transient Stress Capability Waveform

Table 1. Test Circuit Results According to DIN 40839

TEST PULSE	TRANSIENT MAGNITUDE V_S	SOURCE IMPEDANCE R_{SOURCE}	PULSE WIDTH t_d (see Note 5)	PULSE RISE TIME, t_r (see Note 6)	PULSE TIME, t_2 (see Figure 7)	REPETITION PERIOD, t_1 (see Figure 7)	NUMBER OF PULSES
1	–100 V	10 Ω	2 ms	1 µs	200 ms	5 s	5000
2	100 V	10 Ω	50 µs	1 µs	200 ms	5 s	5000
3a	–150 V	50 Ω	0.1 µs	5 ns	100 µs	100 µs	See Note 7
3b	100 V	50 Ω	0.1 µs	5 ns	100 µs	100 µs	See Note 7
5	60 V	1 Ω	400 ms	5 ms	—	—	

NOTES: 5. Measured from 10% on rising edge to 10% on falling edge
6. Measured from 10% to 90% of pulse
7. Pulse package for a period of 3600 s, 10 ms pulse time, 90 ms stop time

APPLICATION INFORMATION

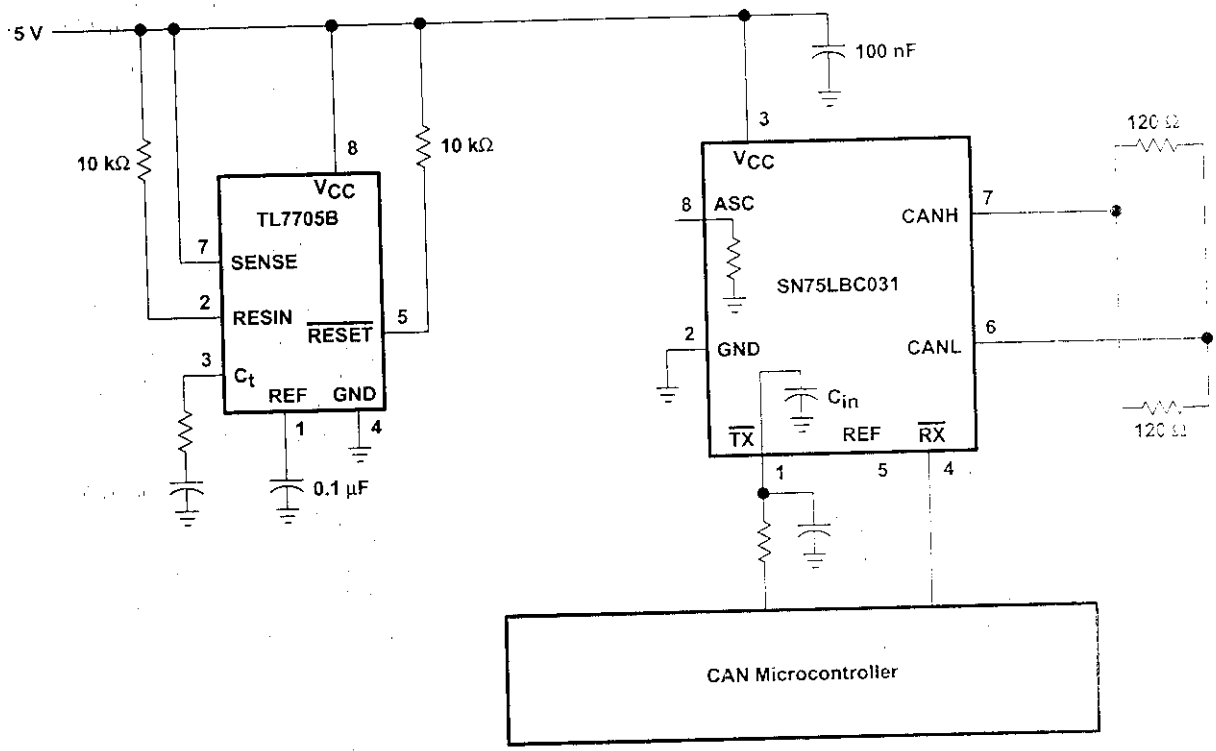
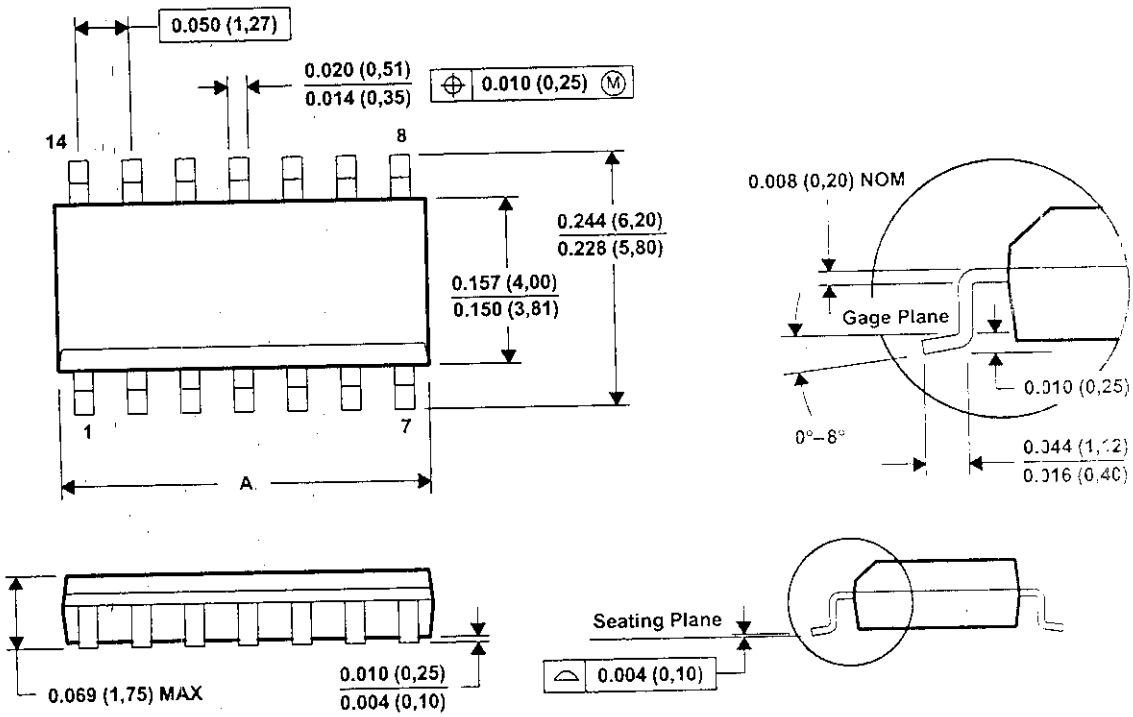


Figure 8. Typical SN75LBC031 Application

MECHANICAL DATA

D (R-PDSO-G**) PLASTIC SMALL-OUTLINE PACKAGE

14 PIN SHOWN



DIM \ PINS **	8	14	16
A MAX	0.197 (5,00)	0.344 (8,75)	0.394 (10,00)
A MIN	0.189 (4,80)	0.337 (8,55)	0.386 (9,80)

4040047/D 10/96

NOTES: A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion, not to exceed 0.006 (0,15).
D. Falls within JEDEC MS-012

Application Report
SLOA101 - August 2002

Introduction to the Controller Area Network (CAN)

Steve Corrigan

HPL

ABSTRACT

A controller area network (CAN) is ideally suited to the many high-level industrial protocols embracing CAN and ISO 11898 as their physical layer. Its cost, performance, and upgradeability provide for tremendous flexibility in system design. This paper presents a brief introduction to the CAN operating principles, the implementation of a basic CAN bus using Texas Instrument's CAN transceivers and DSPs, and a discussion of the robust error detection and fault confinement mechanisms. Some of the properties of CAN, especially relating to the electrical layer and features of transceiver products, are then discussed at a tutorial level.

Contents

1 **Introduction** 3
2 **The CAN Standard** 3
3 **Standard CAN or Extended CAN** 4
3.1 The Bit Fields of Standard CAN and Extended CAN 4
3.1.1 Standard CAN 4
3.1.2 Extended CAN 5
4 **A CAN Message** 6
4.1 Arbitration 6
4.2 Message Types 7
4.2.1 The Data Frame 7
4.2.2 The Remote Frame 8
4.2.3 The Error Frame 8
4.2.4 The Overload Frame 8
4.3 Error Checking and Fault Confinement 8
5 **The CAN Bus** 9
5.1 CAN Transceiver Features 12
5.1.1 Supply Voltage 12
5.1.2 High Short-Circuit Protection 12
5.1.3 High ESD Protection 12
5.1.4 Wide Common-Mode Range 13
5.1.5 Common-Mode Rejection 13
5.1.6 High Input Impedance 14
5.1.7 Controlled Driver Output Transition Times 14
5.1.8 Low Current Standby and Sleep Modes 14
5.1.9 Thermal Shutdown Protection 15
5.1.10 Glitch Free Power Up and Power Down 15

5.1.11 Unpowered Node Does Not Disturb the Bus 15
5.2 The Relationship Between Bus Length and Signaling Rate 15
6 Conclusion 16
7 Additional Reading 16

List of Figures

1 The Layered ISO 11898 Standard Architecture 3
2 Standard CAN: 11-Bit Identifier 4
3 Extended CAN: 29-Bit Identifier 5
4 The Inverted Logic of a CAN Bus 6
5 Arbitration on a CAN Bus 7
6 Details of an Electronic Control Unit 9
7 CAN Dominant and Recessive Bus States of the SN65HVD230 10
8 CAN Bus Traffic 11
9 CAN Test Bus 11
10 3.3-V CAN Transceiver Power Savings 12
11 Common-Mode Noise Coupled onto 4 Twisted-Pair Bus Lines 14

List of Tables

1 CAN Versions 4
2 Maximum Signaling Rates for Various Cable Lengths 15

1 Introduction

The CAN bus was developed by BOSCH[1] as a multi-master, message broadcast system that specifies a maximum signaling rate of 1M bit per second (bps). Unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network many short messages like temperature or RPM are broadcast to the entire network, which allows for data consistency in every node of the system.

This application report explains the CAN message format, message identifiers, and bit-wise arbitration—a major benefit of the CAN signaling scheme. CAN bus implementation is examined and typical waveforms are presented.

2 The CAN Standard

CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for signaling rates up to 1 Mbps, high immunity to electrical interference, and an ability to self-diagnose and repair data errors. These features have led to CAN's popularity in a variety of industries including automotive, marine, medical, manufacturing, and aerospace.

The CAN communications protocol, ISO 11898, describes how information is passed between devices on a network, and conforms to the Open Systems Interconnection (OSI) model that is defined in terms of layers. Actual communication between devices connected by the physical medium is defined by the physical layer of the model. The ISO 11898 architecture defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer in Figure 1.

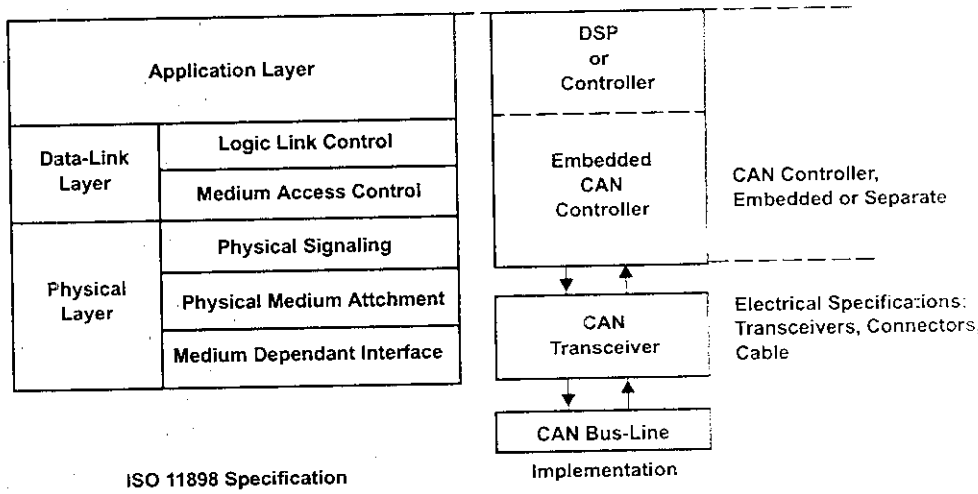


Figure 1. The Layered ISO 11898 Standard Architecture

[1] Robert Bosch GmbH, www.bosch.com

In Figure 1, the application layer establishes the communication link to an upper-level application specific protocol such as the vendor independent CANopen protocol. This protocol is supported by the international users and manufacturers group, CAN in Automation (CiA). Additional CAN information is located at the CiA website, can-cia.de. There are many similar emerging protocols dedicated to particular applications like industrial automation or aviation. Examples of industry-standard CAN-based protocols are KVASER's CAN Kingdom, Allen-Bradley's DeviceNet™ and Honeywell's Smart Distributed System (SDS™).

3 Standard CAN or Extended CAN

The CAN communication protocol is a carrier-sense multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP). CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message. CD+AMP means that collisions are resolved through a bit-wise arbitration, based upon a preprogrammed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access.

The first version of the CAN standards listed in Table 1, ISO 11519 (Low-Speed CAN) is for applications up to 125 kbps with a *standard* 11-bit identifier. The second version, ISO 11898 (1993), also with 11-bit identifiers provides for signaling rates from 125 kbps to 1 Mbps while the more recent ISO 11898 amendment (1995) introduces the *extended* 29-bit identifier. The ISO 11898 11-bit version is often referred to as Standard CAN Version 2.0A, while the ISO 11898 amendment is referred to as Extended CAN Version 2.0B. The Standard CAN 11-bit identifier field in Figure 2 provides for 2^{11}, or 2048 different message identifiers, while the Extended CAN 29-bit identifier in Figure 3 provides for 2^{29}, or 537 million identifiers.

Table 1. CAN Versions

NOMENCLATURE	STANDARD	MAX. SIGNALING RATE	IDENTIFIER
Low–Speed CAN	ISO 11519	125 kbps	11-bit
CAN 2.0A	ISO 11898:1993	1 Mbps	11-bit
CAN 2.0B	ISO 11898:1995	1 Mbps	29-bit

3.1 The Bit Fields of Standard CAN and Extended CAN

3.1.1 Standard CAN

Figure 2. Standard CAN: 11-Bit Identifier

The meaning of the bit fields of Figure 2 are:

- SOF—The single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.
- Identifier—The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.

- RTR—The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way all data being used in a system is uniform.
- IDE—A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
- r0—Reserved bit (for possible use by future standard amendment).
- DLC—The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.
- Data—Up to 64 bits of application data may be transmitted.
- CRC—The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.
- ACK—Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after rearbitration. In this way each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgement bit and the second is a delimiter.
- EOF—This end-of-frame (EOF) 7-bit field marks the end of a CAN frame (message) and disables bit–stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is *stuffed* into the data.
- IFS—This 7-bit inter-frame space (IFS) contains the amount of time required by the controller to move a correctly received frame to its proper position in a message buffer area.

3.1.2 *Extended CAN*

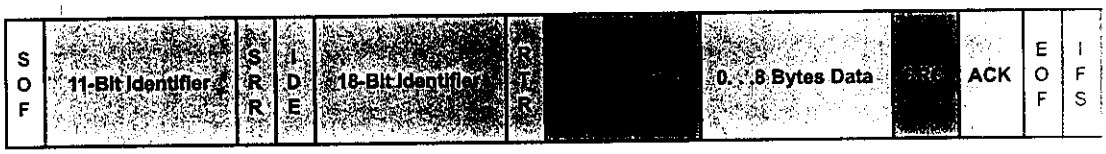


Figure 3. Extended CAN: 29-Bit Identifier

As shown in Figure 3, the Extended CAN message is the same as the Standard message with the addition of:

- SRR—The substitute remote request (SRR) bit replaces the RTR bit in the standard message location as a placeholder in the extended format.
- IDE—A recessive bit in the identifier extension (IDE) indicates that there are more identifier bits to follow. The 18-bit extension follows IDE.
- r1—Following the RTR and r0 bits, an additional reserve bit has been included ahead of the DLC bit.

4 A CAN Message

4.1 Arbitration

A fundamental CAN characteristic shown in Figure 4 is the opposite logic state between the bus, and the driver input and receiver output. Normally a logic high is associated with a one, and a logic low is associated with a zero—but not so on a CAN bus. This is why it is desirable to have the driver input and receiver output pins of a CAN transceiver passively pulled high internally, as in the SN65HVD230. In the absence of any input, the device automatically defaults to a recessive bus state on all input and output pins.

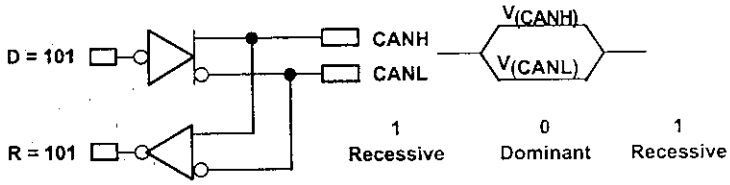


Figure 4. The Inverted Logic of a CAN Bus

Bus access is event-driven and takes place randomly. If two nodes try to occupy the bus simultaneously, access is implemented with a nondestructive, bit-wise arbitration. Nondestructive means that the node winning arbitration just continues on with the message, without the message being destroyed or corrupted by another node.

The allocation of priority to messages in the identifier is a feature of CAN that makes it particularly attractive for use within a real-time control environment. The lower the binary message identifier number, the higher its priority. An identifier consisting entirely of zeros is the highest priority message on a network since it holds the bus dominant the longest. Therefore, if two nodes begin to transmit simultaneously, the node that sends a zero (dominant) while the other nodes send a one (recessive) gets control of the CAN bus and goes on to complete its message. A dominant bit always overwrites a recessive bit on a CAN bus.

Note that a node constantly monitors its own transmission. This is the reason for the transceiver configuration of Figure 4 in which the CANH and CANL output pins of the driver are internally tied to the receiver's input. The propagation delay of a signal in the internal loop from the driver input to the receiver output is typically used as a qualitative measure of a CAN transceiver. This propagation delay is referred to as the loop time (t_{LOOP} in a TI data sheet), but takes on varied nomenclature from vendor to vendor.

Figure 5 displays the arbitration process. Since the nodes continuously monitor their own transmissions, when the node B recessive bit is overwritten by node C's higher priority dominant bit, B detects that the bus state does not match the bit that it transmitted. Therefore, node B halts transmission while node C continues on with its message. Another attempt to transmit the message is made by node B once the bus is released by node C. This functionality is part of the ISO 11898 physical signaling layer, which means that it is contained entirely within the CAN controller and is completely transparent to a CAN user.

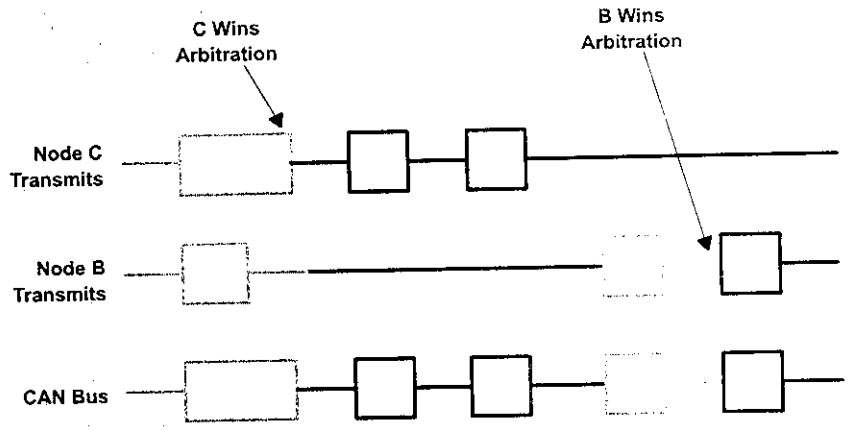


Figure 5. Arbitration on a CAN Bus

The allocation of message priority is up to a system designer, but industry groups mutually agree upon the significance of certain messages. For example, a manufacturer of motor drives may specify that message 0010 is a winding current feedback signal from a motor on a CAN network and that 0011 is the tachometer speed. Since 0010 has the lowest binary identifier, messages relating to current values always have a higher priority on the bus than those concerned with tachometer readings.

In the case of DeviceNet™, devices from many manufacturers such as proximity switches and temperature sensors can be incorporated into the same system. Since the messages generated by DeviceNet sensors have been predefined by their professional association, the Open DeviceNet Vendors Association (ODVA), a certain message always relates to the specific type of sensor such as temperature, regardless of the actual manufacturer.

4.2 Message Types

There are four different message types, or frames (Figures 2 and 3) that can be transmitted on a CAN bus: the data frame, the remote frame, the error frame, and the overload frame. A message is considered to be error free when the last bit of the ending EOF field of a message is received in the error–free recessive state. A dominant bit in the EOF field causes the transmitter to repeat a transmission.

4.2.1 The Data Frame

The data frame is the most common message type, and is made up by the arbitration field, the data field, the CRC field, and the acknowledgement field. The arbitration field determines the priority of a message when two or more nodes are contending for the bus. The arbitration field contains an 11-bit identifier for CAN 2.0A in Figure 2 and the RTR bit, which is dominant for data frames. For CAN 2.0B in Figure 3 it contains the 29-bit identifier and the RTR bit. Next is the data field which contains zero to eight bytes of data, and the CRC field which contains the 16-bit checksum used for error detection. Lastly, there is the acknowledgement field. Any CAN controller that is able to correctly receive a message sends a dominant ACK bit that overwrites the transmitted recessive bit at the end of correct message transmission. The transmitter checks for the presence of the dominant ACK bit and retransmits the message if no acknowledge is detected.

4.2.2 The Remote Frame

The intended purpose of the remote frame is to solicit the transmission of data from another node. The remote frame is similar to the data frame, with two important differences. First, this type of message is explicitly marked as a remote frame by a recessive RTR bit in the arbitration field, and secondly, there is no data.

4.2.3 The Error Frame

The error frame is a special message that violates the formatting rules of a CAN message. It is transmitted when a node detects an error in a message, and causes all other nodes in the network to send an error frame as well. The original transmitter then automatically retransmits the message. There is an elaborate system of error counters in the CAN controller which ensures that a node cannot tie up a bus by repeatedly transmitting error frames.

4.2.4 The Overload Frame

The overload frame is mentioned here for completeness. It is similar to the error frame with regard to the format, and it is transmitted by a node that becomes too busy. It is primarily used to provide for an extra delay between messages.

4.3 Error Checking and Fault Confinement

The robustness of CAN may be attributed in part to its abundant error checking procedures. The CAN protocol incorporates five methods of error checking: three at the message level and two at the bit level. If a message fails with any one of these error detection methods, it is not accepted and an error frame is generated from the receiving nodes, causing the transmitting node to resend the message until it is received correctly. However, if a faulty node hangs up a bus by continuously repeating an error, its transmit capability is removed by its controller after an error limit is reached.

At the message level are the CRC and the ACK slots displayed in Figures 2 and 3. The 16-bit CRC contains the checksum of the preceding application data for error detection with a 15-bit checksum and 1-bit delimiter. The ACK field is two bits long and consists of the acknowledge bit and an acknowledge delimiter bit. Finally, at the message level there is a form check. This check looks for fields in the message which must always be recessive bits. If a dominant bit is detected, an error is generated. The bits checked are the SOF, EOF, ACK delimiter, and the CRC delimiter bits.

At the bit level each bit transmitted is monitored by the transmitter of the message. If a data bit (not arbitration bit) is written onto the bus and its opposite is read, an error is generated. The only exceptions to this are with the message identifier field which is used for arbitration, and the acknowledge slot which requires a recessive bit to be overwritten by a dominant bit. The final method of error detection is with the bit stuffing rule where after five consecutive bits of the same logic level, if the next bit is not a compliment, an error is generated. Stuffing ensures rising edges available for on-going synchronization of the network, and that a stream of recessive bits are not mistaken for an error frame, or the seven-bit interframe space that signifies the end of a message. Stuffed bits are removed by a receiving node's controller before the data is forwarded to the application.

With this logic, an active error frame consists of six dominant bits—violating the bit stuffing rule. This is interpreted as an error by all of the CAN nodes which then generate their own error frame. This means that an error frame can be from the original six bits to twelve bits long with all the replies. This error frame is then followed by a delimiter field of eight recessive bits and a bus idle period before the corrupted message is retransmitted. It is important to note that the retransmitted message still has to contend for arbitration on the bus.

5 The CAN Bus

The data link and physical signaling layers of Figure 1, which are normally transparent to a system operator, are included in any controller that implements the CAN protocol, such as Texas Instruments' TMS320LF2407 3.3-V DSP with integrated CAN controller. Connection to the physical medium is then implemented through a line transceiver such as TI's SN65HVD230 3.3-V CAN transceiver to form what the ISO–11898 standard refers to as an electronic control unit (ECU) in Figure 6.

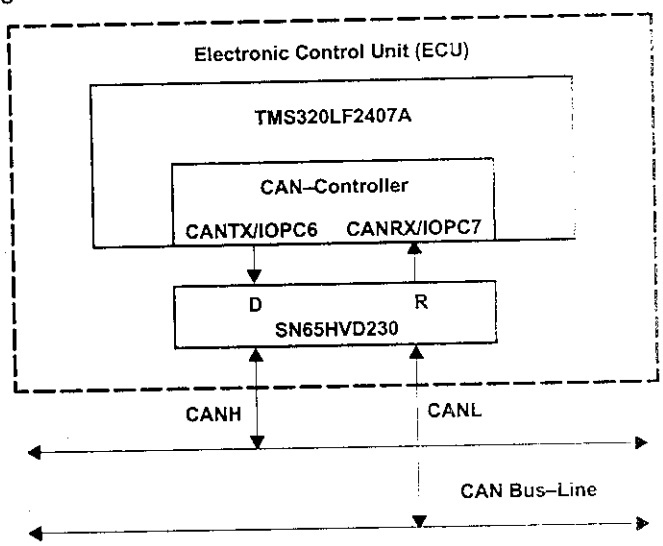


Figure 6. Details of an Electronic Control Unit

Signaling is differential which is where CAN derives its robust noise immunity and fault tolerance. Balanced differential signaling reduces noise coupling and allows for high signaling rates over twisted-pair cable. Balanced means that the current flowing in each signal line is equal but opposite in direction, resulting in a field-canceling effect that is a key to low noise emissions. The use of balanced differential receivers and twisted-pair cabling enhance the common-mode rejection and high noise immunity of a CAN bus.

The two signal lines of the bus, CANH and CANL, in the quiescent recessive state, are passively biased to ≈ 2.5 V. The dominant state on the bus takes CANH ≈ 1 V higher to ≈ 3.5 V, and takes CANL ≈ 1 V lower to ≈ 1.5 V creating a typical 2-V differential signal as displayed in Figure 7.

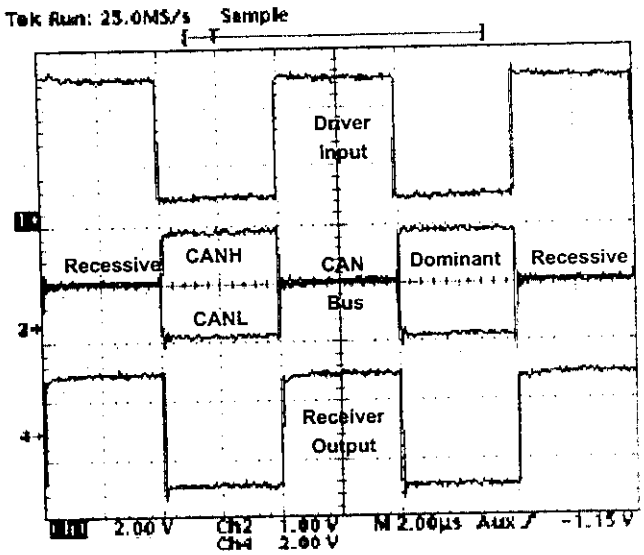


Figure 7. CAN Dominant and Recessive Bus States of the SN65HVD230

The CAN standard defines a communication network that links all the nodes connected to a bus and enables them to talk with one another. There may or may not be a central control node, and nodes may be added at any time, even while the network is operating (hot-plugging).

The nodes could theoretically be sending messages from smart sensing technology and a motor controller. An actual application may include a temperature sensor sending out a temperature update that is used to adjust the motor speed of a fan. If a pressure sensor node wants to send a message at the same time, arbitration assures that the message is sent. For instance, Node A in Figures 8 and 9 finishes sending its message as nodes B and C acknowledge a correct message being received. Nodes B and C then begin arbitration—node C wins the arbitration and sends its message. Nodes A and B acknowledge C's message, and node B then continues on with its message. Again note the opposite polarity of the driver input and output on the bus.

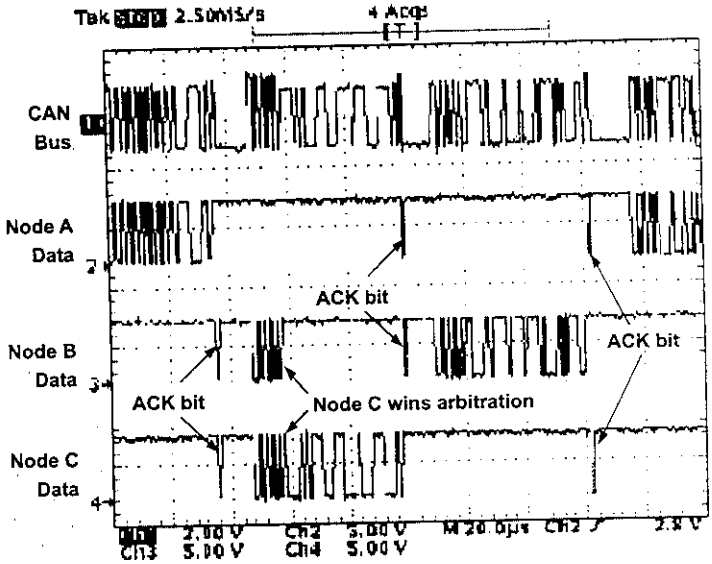


Figure 8. CAN Bus Traffic

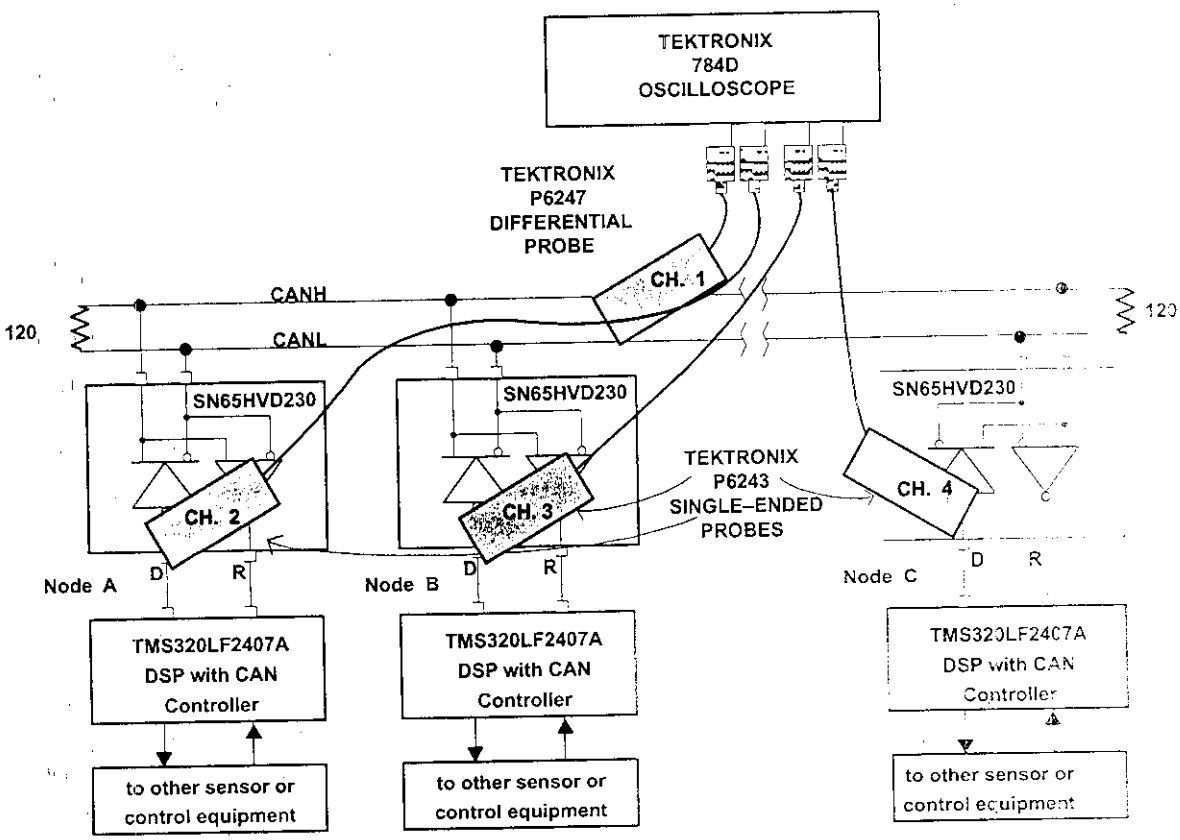


Figure 9. CAN Test Bus

5.1 CAN Transceiver Features

5.1.1 *Supply Voltage*

Most CAN transceivers like the SN65HVD251 require a 5-V power supply to reach the signal levels required by the ISO 11898 standard. By superior attention to high-efficiency circuit design, the Texas Instruments SN65HVD23x CAN transceiver family operates with a 3.3-V power supply, and is fully interoperable with 5-V CAN transceivers on the same bus. This allows designers to reduce total node power by 50% or more (Figure 10).

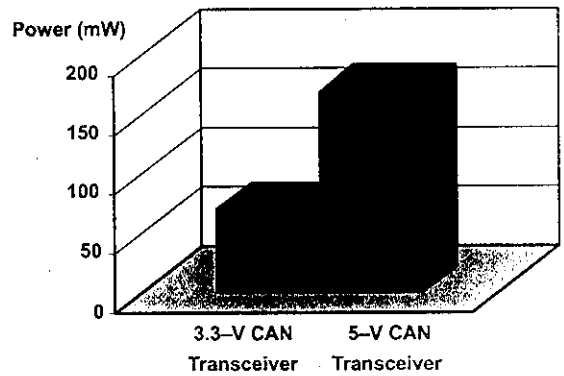


Figure 10. 3.3-V CAN Transceiver Power Savings

For applications using 3.3-V technology, such as the Texas Instruments TMS320C240x™ family of DSPs with integrated CAN controllers, the need for a 5 V power supply can be eliminated. In addition to the inherent power savings of using a 3.3-V transceiver, this lowers the overall part count for the node, thereby reducing system cost and increasing system reliability.

For designers with an existing system design requiring a 5-V powered transceiver, the SN65HVD251 is available as an improved alternate for the commonly-used PCA82C250 and PCA82C251. While the 'HVD251 is a drop-in replacement for the '82C250 and '82C251, it offers the additional features of higher ESD protection, shorter loop delay, and wider common-mode range.

5.1.2 *High Short-Circuit Protection*

The CAN standard recommends that a transceiver survive bus wire short-circuits to each other, to the power supply, and to ground. This ensures that transceivers are not damaged by bus cable polarity reversals, cable crush, and accidental shorts to high power supplies. While the 'HVD23x can survive short-circuits to voltages in the range of –4 V to 16 V, the 'HVD251 provides an extended short-circuit protection to voltages in the –36 V to 36 V range.

5.1.3 *High ESD Protection*

Static charge is an unbalanced electrical charge at rest, typically created by the physical contact of different materials. One surface gains electrons, while the other surface loses electrons. This results in an unbalanced electrical condition known as a static charge. When a static charge moves from one surface to another, it is referred to as an electrostatic discharge (ESD). It can occur only when the voltage differential between the two surfaces is sufficiently high to break down the dielectric strength of the medium separating the two surfaces.

ESD can occur in any one of four ways: a charged body can touch an IC, a charged IC can touch a grounded surface, a charged machine can touch an IC, or an electrostatic field can induce a voltage across a dielectric sufficient to break it down. It becomes readily apparent that a high ESD rating not only indicates a robust transceiver, but a robust circuit design as well.

While the earliest CAN transceivers on the market have only a 2 kV or 4 kV ESD maximum on the bus pins, the HVD2xx families of CAN transceivers have ESD ratings as high as 16 kV on the bus pins when tested in accordance with the human body model (HBM) of JEDEC Standard 22, Test Method A114-A. With these ESD ratings, the TI HVD2xx CAN transceivers are much better suited to the harsh electrical environments of automotive and industrial applications than the earlier transceiver versions of other vendors.

5.1.4 *Wide Common-Mode Range*

Common-mode noise is the difference in potential between grounds of sending and receiving nodes on a bus. This is often the case in the networked equipment typically found in a CAN application. Possible effects of this problem are intermittent reboots, lockups, bad data transfer, or physical damage to the transceiver.

Network interface cards, parallel ports, serial ports, and especially transceivers are prime targets for some form of failure if not designed to accommodate high levels of ground noise and power supply imbalance between typical CAN nodes.

With this in mind the 'HVD230 family is designed to operate with complete safety over the bus voltage range of –2 V to 7 V required by the standard. However, the 'HVD251 extends this safe operating range even farther with operation from –7 V to 12 V.

5.1.5 *Common-Mode Rejection*

Common-mode noise of varied magnitudes exists within the networks associated with CAN applications. Noise from pulsing motor controllers, switch-mode power supplies, or from fluorescent lighting are the typical sources of noises that couple onto bus lines (displayed in Figure 11).

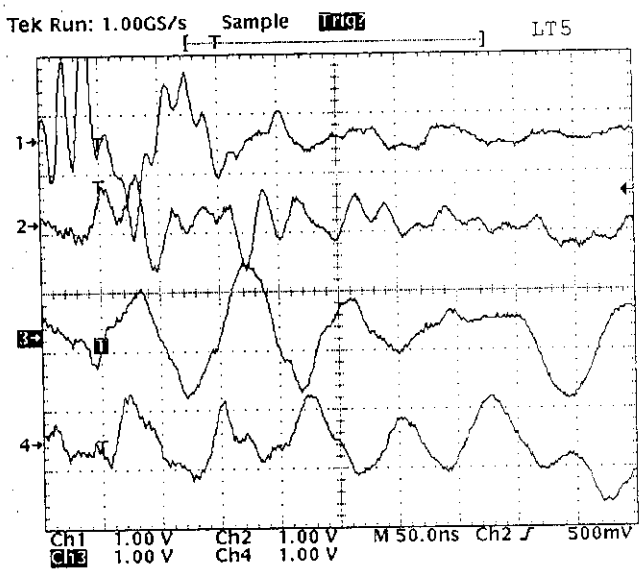


Figure 11. Common-Mode Noise Coupled onto 4 Twisted-Pair Bus Lines

A CAN transceiver not specifically designed to reject this coupled noise can respond to common-mode noise as if it were data on a bus and send meaningless data to a controller. The 'HVD23x and 'HVD251 are specifically designed and tested for their ability to reject this common-mode noise.

5.1.6 *High Input Impedance*

A high bus input impedance increases the number of nodes that can be added to a bus above the standard's 30 nodes. The high impedance restricts the amount of current that a receiver sinks or sources onto a bus over common-mode voltage conditions. This ensures that a driver transmitting a message into such a condition is not required to sink or source an excessive amount of current from the sum of the receiver currents on the bus.

5.1.7 *Controlled Driver Output Transition Times*

Controlling driver output slew-rate dampens the rise-time of a dominant bit to improve signal quality and provides for longer stub lengths and a better bit-error rate.

5.1.8 *Low Current Standby and Sleep Modes*

Many applications are looking to lower-power opportunities as more electronics are added to designs. The standby mode of the 'HVD230 is generally referred to as the *listen only* mode, since in standby, the 'HVD230's driver circuitry is switched off while the receiver continues to monitor bus activity. In the occurrence of a dominant bit on the bus, the receiver passes this information along to its DSP/CAN controller which in turn activates the circuits that are in standby.

The difference between the 'HVD230 and 'HVD231 is that both driver and receiver circuits can be switched off in the 'HVD231 to create an extremely low-power sleep mode.

5.1.9 Thermal Shutdown Protection

Another desirable safety feature for a CAN transceiver is the thermal shutdown circuitry of the 'HVD23x and 'HVD251. This feature protects a device against the destructive currents and resulting heat that can occur in a short–circuit condition. Once thermal shutdown is activated, the device remains shut down until the circuitry is allowed to cool.

5.1.10 Glitch Free Power Up and Power Down

This feature provides for hot–plugging onto a powered bus without disturbing the network. The 'HVD23x's and 'HVD251's driver and receiver pins are passively pulled high internally while the bus pins are biased internally to a high-impedance recessive state. This provides for a power-up into a known recessive condition without disturbing ongoing bus communication.

5.1.11 Unpowered Node Does Not Disturb the Bus

Several CAN transceivers on the market today have very low output impedance when un–powered. This low impedance causes the device to sink any signal present on the bus and shuts down all data transmission. The 'HVD23x and 'HVD251 have a very high output impedance in powered and unpowered conditions, and maintain the integrity of the bus when power or ground is removed from the circuit.

5.2 The Relationship Between Bus Length and Signaling Rate

The basics of arbitration require that the front wave of the first bit of a message travel to the most remote node on a network and back again before the bit is designated by the receiver of the sending node as dominant or recessive (typically this *sample* is made at about two–thirds the bit width). With this limitation, the maximum bus length and signaling rate are determined by network parameters.

Factors to be considered in network design include the ≈ 5 ns/m propagation delay of typical twisted-pair bus cable, signal amplitude loss due to the loss mechanisms of the cable, and the input impedance of the bus transceivers. Under strict analysis, variations among the different oscillators in a system also need to be accounted for with adjustments in signaling rate and bus length. Maximum signaling rates achieved with the SN65HVD230 in high-speed mode with several bus lengths are listed in Table 2.

Table 2. Maximum Signaling Rates for Various Cable Lengths

BUS LENGTH (m)	SIGNALING RATE (kbps)
30	1000
100	500
250	250
500	125
1000	62.5

The ISO 11898 standard specifications are given for a maximum bus length of 40 m and maximum stub length of 0.3 m with a maximum of 30 nodes. However, with careful design, longer cables, longer stub lengths, and many more nodes can be added to a bus—always with a trade-off in signaling rate. A transceiver with high input impedance such as the HVD230 is needed to increase the number of nodes on a bus.

The cable is specified to be a shielded or unshielded twisted-pair with a 120-Ω characteristic impedance (Zo). The standard defines the interconnection to be a single twisted-pair cable. The interconnection is terminated at both ends with a resistor equal to the characteristic impedance of the line to prevent signal reflections. Nodes are then connected to the bus with unterminated drop cables, or stubs, which should be kept as short as possible to minimize signal reflections.

Connectors, while not specified by the standard should not affect standard operating parameters such as the minimum VOD. Although unshielded cable is used in many applications, data transmission circuits employing CAN transceivers are used in applications requiring a rugged interconnection with a wide common-mode voltage range. Therefore, shielded cable is recommended in these electronically harsh environments, and when coupled with the standard's –2 V to 7 V common-mode range of tolerable ground noise, helps to ensure data integrity.

6 Conclusion

CAN is ideally suited in applications requiring a high number of short messages in a short period of time with high reliability in rugged operating environments. Since CAN is message based and not address based, it is especially suited when data is needed by more than one location and system-wide data consistency is mandatory.

Fault confinement is also a major benefit of CAN. Faulty nodes are automatically dropped from the bus, which prevents any single node from bringing a network down, and assures that bandwidth is always available for critical message transmission. This error containment also allows nodes to be added to a bus while the system is in operation, otherwise known as hot-plugging.

The many features of the 'HVD23x and 'HVD251 make them ideally suited for the many rugged applications to which the CAN protocol is being adapted. Among the applications finding solutions with CAN are automobiles, trucks, trains, busses, airplanes, and agriculture, construction, mining, and marine vehicles. CAN-based control systems are being used in factory and building automation and embedded control systems for machines, medical devices, domestic appliances and many other applications.

7 Additional Reading

1. Controller Area Network, Basics Protocols, Chips and Applications; Dr. Konrad Etschberger; ISBN 3–00–007376–0 (www.ixxat.com)
2. CAN Systems Engineering, From Theory to Practical Applications; Wolfhard Lawrenz. ISBN 0–387–94939–9

Section 15. Monitor ROM (MON)

15.1 Introduction

This section describes the monitor ROM (MON) and the monitor mode entry methods. The monitor ROM allows complete testing of the microcontroller unit (MCU) through a single-wire interface with a host computer. Monitor mode entry can be achieved without use of the higher test voltage, V_{TST}, as long as vector addresses $FFFE and $FFFF are blank, thus reducing the hardware requirements for in-circuit programming.

15.2 Features

Features of the monitor ROM include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor read-only memory (ROM) and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- Standard communication baud rate (7200 @ 8-MHz crystal frequency)
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security feature[1]
- FLASH memory programming interface
- 350 bytes monitor ROM code size ($FE20 to $FF7D)
- Monitor mode entry without high voltage, V_{TST}, if reset vector is blank ($FFFE and $FFFF contain $FF)
- Normal monitor mode entry if high voltage is applied to $\overline{IRQ}$

15.3 Functional Description

Figure 15-1 shows a simplified diagram of the monitor mode.

The monitor ROM receives and executes commands from a host computer. **Figure 15-2** and **Figure 15-3** show example circuits used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the FLASH difficult for unauthorized users.

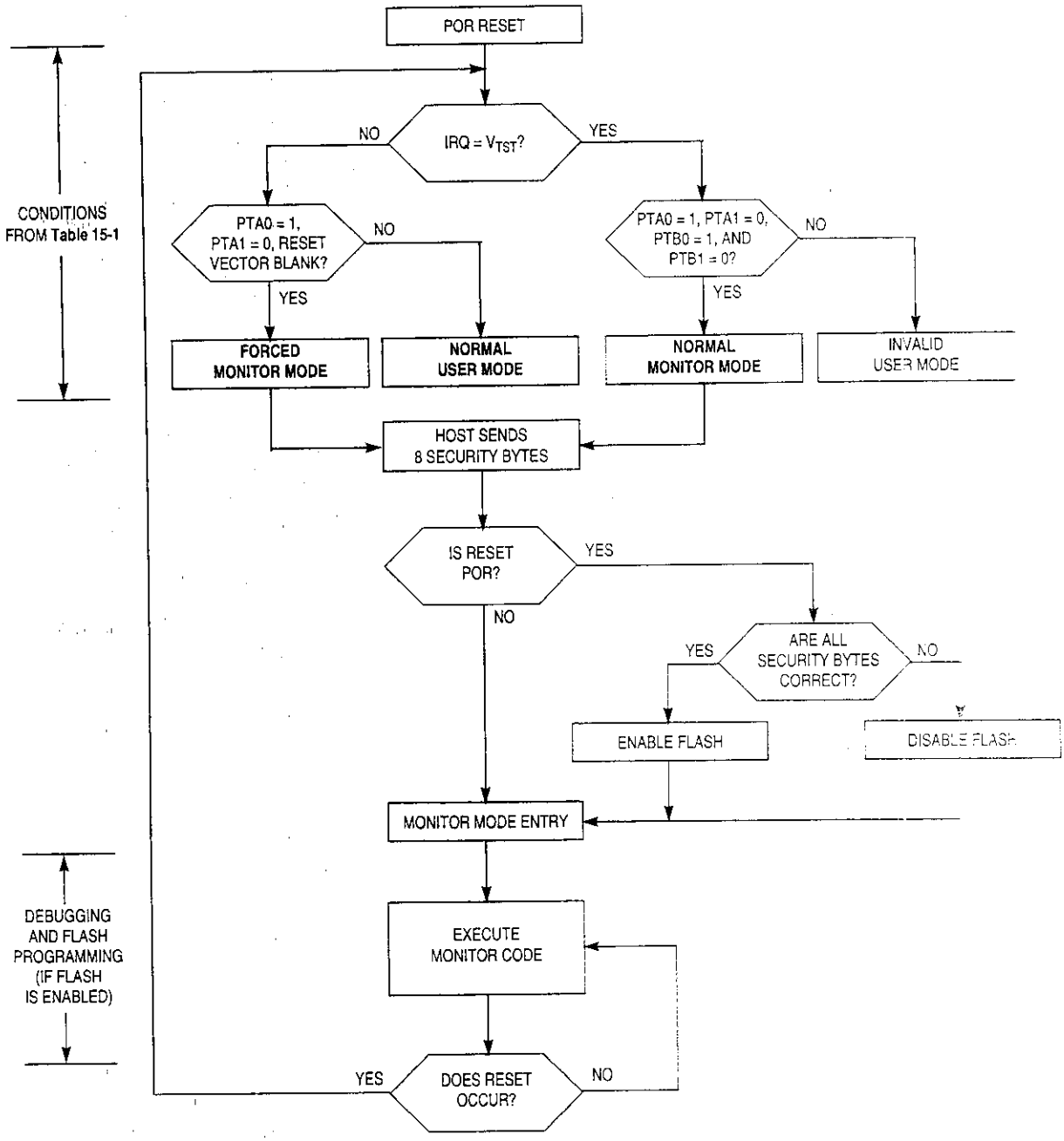


Figure 15-1. Simplified Monitor Mode Entry Flowchart

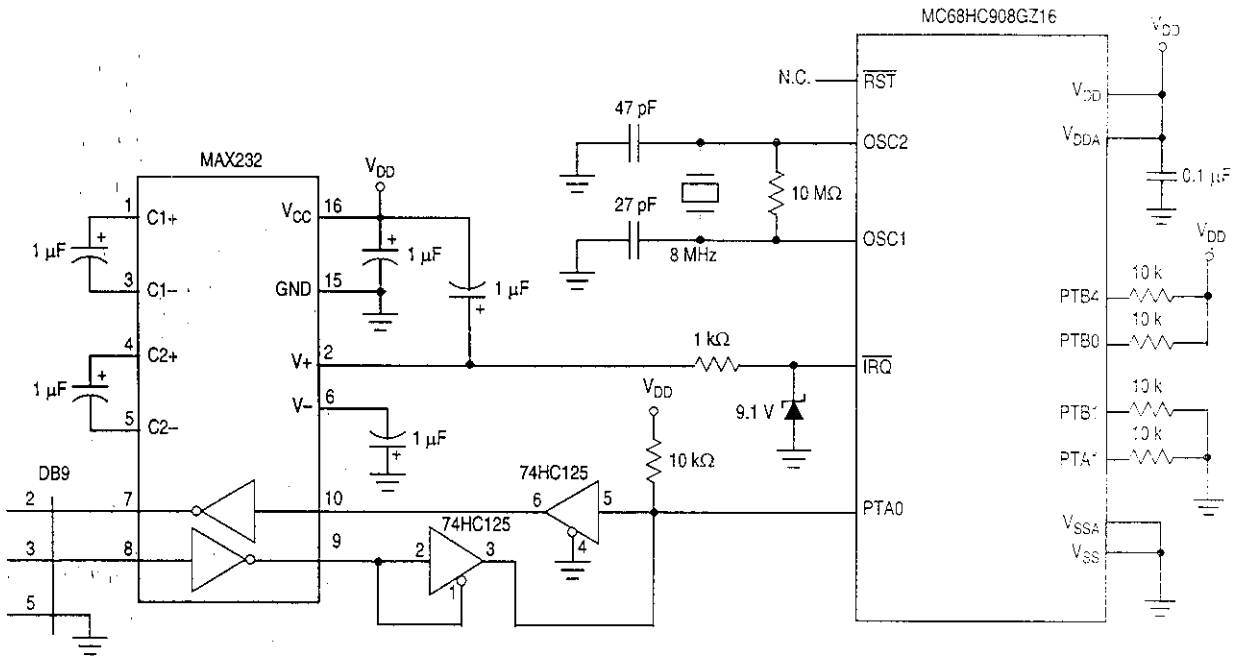


Figure 15-2. Normal Monitor Mode Circuit

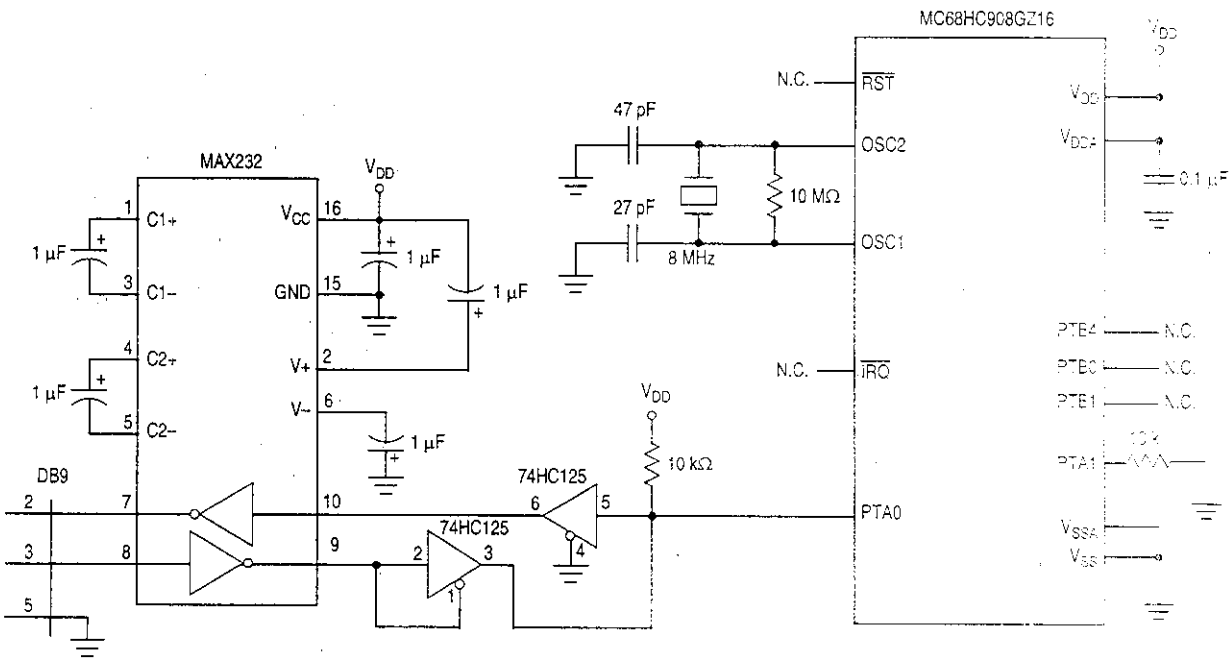


Figure 15-3. Forced Monitor Mode

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute code downloaded into RAM by a host computer while most MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

Table 15-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on reset (POR) and will allow communication at 7200 baud provided one of the following sets of conditions is met:

- If $FFFE and $FFFF does not contain $FF (programmed state):
  - The external clock is 4 MHz (7200 baud)
  - PTB4 = low
  - $\overline{IRQ} = V_{TST}$
- If $FFFE and $FFFF do not contain $FF (programmed state):
  - The external clock is 8 MHz (7200 baud)
  - PTB4 = high
  - IRQ = V_{TST}
- If $FFFE and $FFFF contain $FF (erased state):
  - The external clock is 8 MHz (7200 baud)
  - $\overline{IRQ} = V_{DD}$ (this can be implemented through the internal $\overline{IRQ}$ pullup) or V_{SS}

Enter monitor mode with pin configuration shown in **Table 15-1** by pulling $\overline{RST}$ low and then high. The rising edge of $\overline{RST}$ latches monitor mode. Once monitor mode is latched, the values on the specified pins can change.

Once out of reset, the MCU waits for the host to send eight security bytes (see **15.4 Security**). After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host, indicating that it is ready to receive a command.

Table 15-1. Monitor Mode Signal Requirements and Options

Mode	$\overline{IRQ}$	$\overline{RST}$	Reset Vector	Serial Communication		Mode Selection		Divider	PLL	COP	Communication Speed			Comments
				PTA0	PTA1	PTB0	PTB1	PTB4			External Clock	Bus Frequency	Baud Rate	
—	X	GND	X	X	X	X	X	X	X	X	X	X	X	Reset condition
Normal Monitor	V_{TST}	V_{DD} or V_{TST}	X	1	0	1	0	0	OFF	Disabled	4 MHz	2 MHz	7200	
	V_{TST}	V_{DD} or V_{TST}	X	1	0	1	0	1	OFF	Disabled	8 MHz	2 MHz	7200	
Forced Monitor	V_{DD} or GND	V_{DD}	$FF (blank)	1	0	X	X	X	OFF	Disabled	8 MHz	2 MHz	7200	
User	V_{DD} or GND	V_{DD} or V_{TST}	Not $FF	X	X	X	X	X	X	Enabled	X	X	X	
MON08 Function [Pin No.]	V_{TST} [6]	$\overline{RST}$ [4]	—	COM [8]	SSEL [10]	MOD0 [12]	MOD1 [14]	DIV4 [16]	—	—	OSC1 [13]	—	—	

1. PTA0 must have a pullup resistor to V_{DD} in monitor mode.
2. Communication speed in the table is an example to obtain a baud rate of 7200. Baud rate using external oscillator is bus frequency / 278.
3. External clock is an 4.0 MHz or 8.0 MHz crystal on OSC1 and OSC2 or a canned oscillator on OSC1.
4. X = don't care
5. MON08 pin refers to P&E Microcomputer Systems' MON08-Cyclone 2 by 8-pin connector.

NC	1	2	GND
NC	3	4	$\overline{RST}$
NC	5	6	$\overline{IRQ}$
NC	7	8	PTA0
NC	9	10	PTA1
NC	11	12	PTB0
OSC1	13	14	PTB1
V_{DD}	15	16	PTB4

15.3.1 Normal Monitor Mode

If V_{TST} is applied to $\overline{IRQ}$ and PTB4 is low upon monitor mode entry, the bus frequency is a divide-by-two of the input clock. If PTB4 is high with V_{TST} applied to $\overline{IRQ}$ upon monitor mode entry, the bus frequency will be a divide-by-four of the input clock. Holding the PTB4 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator *only if V_{TST} is applied to $\overline{IRQ}$*. In this event, the CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.

When monitor mode was entered with V_{TST} on $\overline{IRQ}$, the computer operating properly (COP) is disabled as long as V_{TST} is applied to either $\overline{IRQ}$ or $\overline{RST}$.

This condition states that as long as V_{TST} is maintained on the $\overline{IRQ}$ pin after entering monitor mode, or if V_{TST} is applied to $\overline{RST}$ after the initial reset to get into monitor mode (when V_{TST} was applied to $\overline{IRQ}$), then the COP will be disabled. In the latter situation, after V_{TST} is applied to the $\overline{RST}$ pin, V_{TST} can be removed from the $\overline{IRQ}$ pin in the interest of freeing the $\overline{IRQ}$ for normal functionality in monitor mode.

15.3.2 Forced Monitor Mode

If entering monitor mode without high voltage on $\overline{IRQ}$, then all port B pin requirements and conditions, including the PTB4 frequency divisor selection, are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

NOTE: *If the reset vector is blank and monitor mode is entered, the chip will see an additional reset cycle after the initial power-on reset (POR). Once the reset vector has been programmed, the traditional method of applying a voltage, V_{TST}, to $\overline{IRQ}$ must be used to enter monitor mode.*

An external oscillator of 8 MHz is required for a baud rate of 7200, as the internal bus frequency is automatically set to the external frequency divided by four.

When the forced monitor mode is entered the COP is always disabled regardless of the state of $\overline{IRQ}$ or $\overline{RST}$.

15.3.3 Monitor Vectors

In monitor mode, the MCU uses different vectors for reset, SWI (software interrupt), and break interrupt than those for user mode. The alternate vectors are in the $FE page instead of the $FF page and allow code execution from the internal monitor firmware instead of user code.

Table 15-2 summarizes the differences between user mode and monitor mode.

Table 15-2. Mode Differences

Modes	Functions					
	Reset Vector High	Reset Vector Low	Break Vector High	Break Vector Low	SWI Vector High	SWI Vector Low
User	$FFFE	$FFFF	$FFFC	$FFFD	$FFFC	$FFFD
Monitor	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

15.3.4 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical.

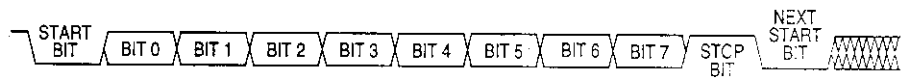


Figure 15-4. Monitor Data Format

15.3.5 Break Signal

A start bit (logic 0) followed by nine logic 0 bits is a break signal. When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits and then echoes back the break signal.

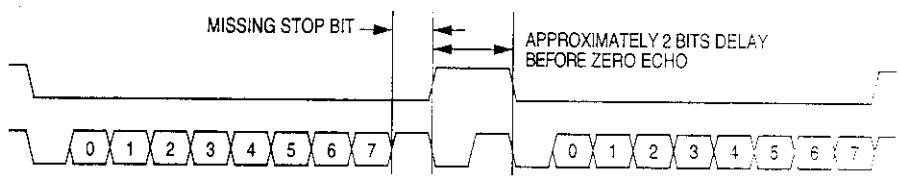


Figure 15-5. Break Transaction

15.3.6 Baud Rate

The communication baud rate is controlled by the crystal frequency or external clock and the state of the PTB4 pin (when $\overline{IRQ}$ is set to V_{TST}) upon entry into monitor mode. If monitor mode was entered with V_{DD} on $\overline{IRQ}$ and the reset vector blank, then the baud rate is independent of PTB4.

Table 15-1 also lists external frequencies required to achieve a standard baud rate of 7200 bps. The effective baud rate is the bus frequency divided by 278. If using a crystal as the clock source, be aware of the upper frequency limit that the internal clock module can handle. See **24.7 5.0-Volt Control Timing** or **24.6 3.3-Vdc Electrical Characteristics** for this limit.

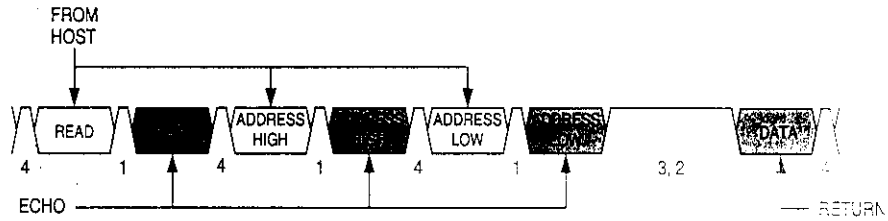
15.3.7 Commands

The monitor ROM firmware uses these commands:

- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

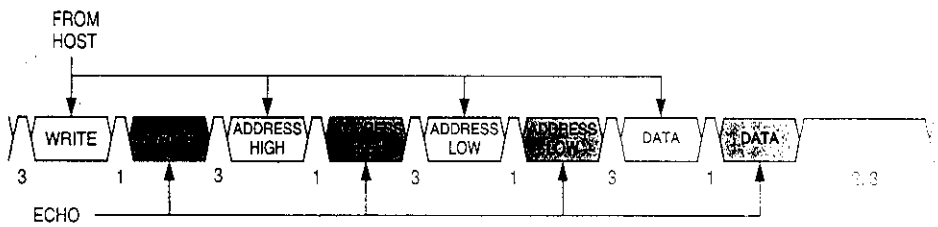
The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

NOTE: *Wait one bit time after each echo before sending the next byte.*



Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

Figure 15-6. Read Transaction



Notes:
1 = Echo delay, approximately 2 bit times
2 = Cancel command delay, 11 bit times
3 = Wait 1 bit time before sending next byte.

Figure 15-7. Write Transaction

A brief description of each monitor mode command is given in Table 15-3 through Table 15-8.

Table 15-3. READ (Read Memory) Command

Description	Read byte from memory
Operand	2-byte address in high-byte:low-byte order
Data Returned	Returns contents of specified address
Opcode	$4A

Command Sequence

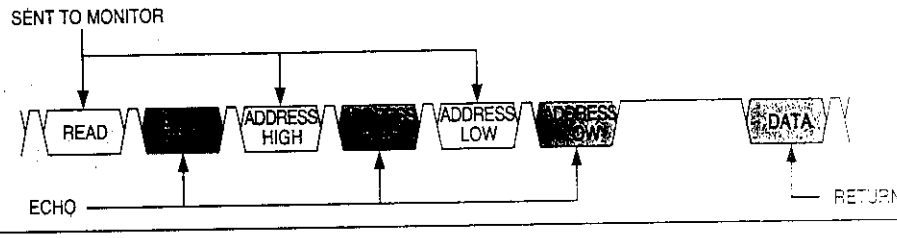


Table 15-4. WRITE (Write Memory) Command

Description	Write byte to memory
Operand	2-byte address in high-byte:low-byte order; low byte followed by data byte
Data Returned	None
Opcode	$49

Command Sequence

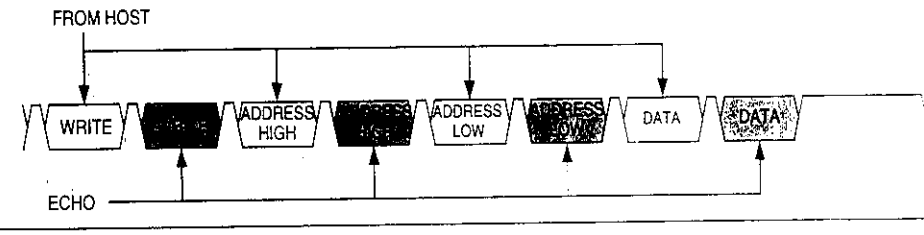


Table 15-5. IREAD (Indexed Read) Command

Description	Read next 2 bytes in memory from last address accessed
Operand	2-byte address in high byte:low byte order
Data Returned	Returns contents of next two addresses
Opcode	$1A

Command Sequence

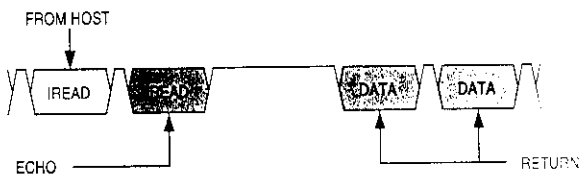
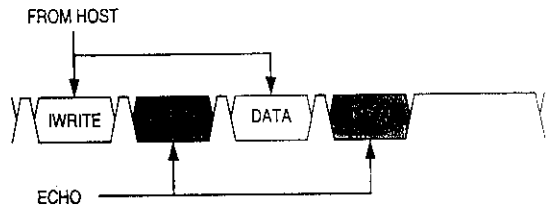


Table 15-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Single data byte
Data Returned	None
Opcode	$19

Command Sequence



A sequence of IREAD or IWRITE commands can access a block of memory sequentially over the full 64-Kbyte memory map.

Table 15-7. READSP (Read Stack Pointer) Command

Description	Reads stack pointer
Operand	None
Data Returned	Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order
Opcode	$0C

Command Sequence

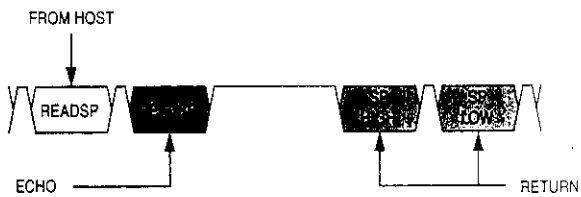
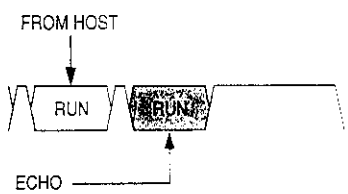


Table 15-8. RUN (Run User Program) Command

Description	Executes PULH and RTI instructions
Operand	None
Data Returned	None
Opcode	$28

Command Sequence



The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.

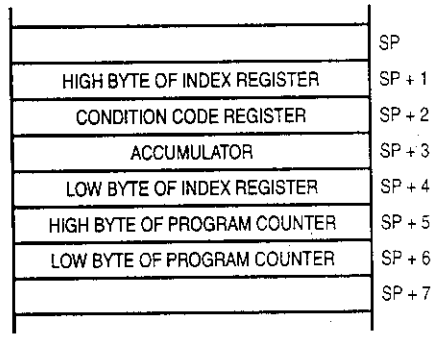


Figure 15-8. Stack Pointer at Monitor Mode Entry

15.4 Security

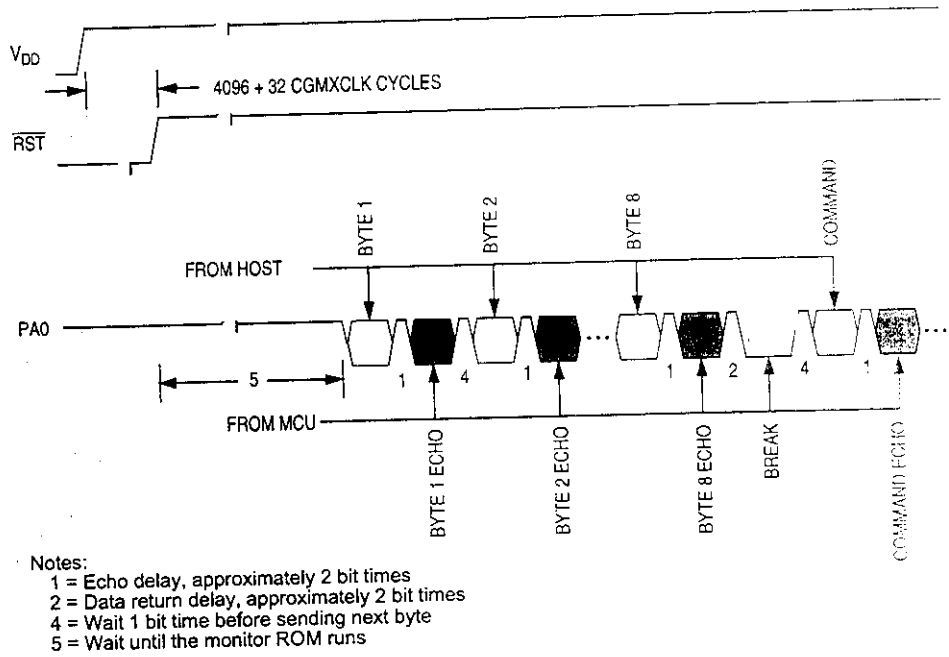
A security feature discourages unauthorized reading of FLASH locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. Locations $FFF6–$FFFD contain user-defined data.

NOTE: *Do not leave locations $FFF6–$FFFD blank. For security reasons, program locations $FFF6–$FFFD even if they are not used for vectors.*

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. See **Figure 15-9**.

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE: *The MCU does not transmit a break character until after the host sends the eight security bytes.*



Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
4 = Wait 1 bit time before sending next byte
5 = Wait until the monitor ROM runs

Figure 15-9. Monitor Mode Entry Timing

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $40 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

Section 16. MSCAN08 Controller (MSCAN08)

16.1 Introduction

The MSCAN08 is the specific implementation of the Motorola scalable controller area network (MSCAN) concept targeted for the Motorola M68HC08 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September, 1991.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the electromagnetic interference (EMI) environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN08 utilizes an advanced buffer arrangement, resulting in a predictable real-time behavior, and simplifies the application software.

16.2 Features

Basic features of the MSCAN08 are:

- MSCAN08 enable is software controlled by bit (MSCANEN) in configuration register (CONFIG2)
- Modular architecture
- Implementation of the CAN Protocol — Version 2.0A/B
  - Standard and extended data frames
  - 0–8 bytes data length.
  - Programmable bit rate up to 1 Mbps depending on the actual bit timing and the clock jitter of the phase-locked loop (PLL)
- Support for remote frames
- Double-buffered receive storage scheme
- Triple-buffered transmit storage scheme with internal prioritization using a "local priority" concept
- Flexible maskable identifier filter supports alternatively one full size extended identifier filter or two 16-bit filters or four 8-bit filters
- Programmable wakeup functionality with integrated low-pass filter
- Programmable loop-back mode supports self-test operation
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus off)

- Programmable MSCAN08 clock source either CPU bus clock or crystal oscillator output
- Programmable link to timer interface module 2, channel 0, for time-stamping and network synchronization
- Low-power sleep mode

16.3 External Pins

The MSCAN08 uses two external pins, one input (CAN_{RX}) and one output (CAN_{TX}). The CAN_{TX} output pin represents the logic level on the CAN: 0 is for a dominant state, and 1 is for a recessive state.

A typical CAN system with MSCAN08 is shown in **Figure 16-1**.

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection against defected CAN or defected stations.

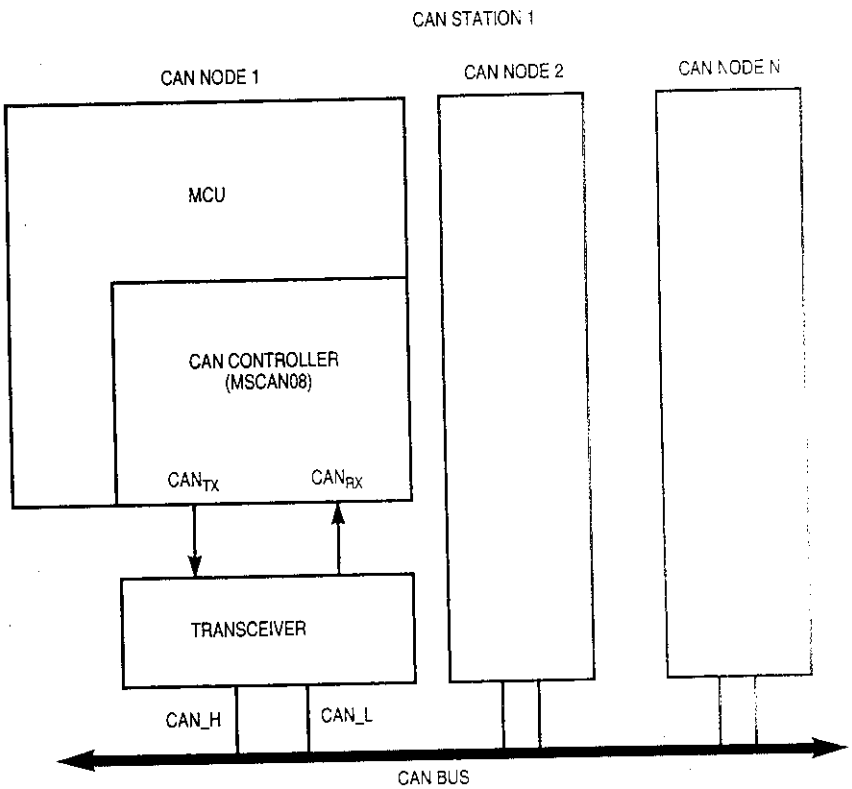


Figure 16-1. The CAN System

16.4 Message Storage

MSCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

16.4.1 Background

Modern application layer software is built under two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message being sent and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer. In that case, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN08 implements with the "local priority" concept described in 16.4.2 Receive **Structures**.

16.4.2 Receive Structures

The received messages are stored in a 2-stage input first in first out (FIFO). The two message buffers are mapped using a "ping pong" arrangement into a single memory area (see **Figure 16-2**). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the central processor unit (CPU08). This scheme simplifies the handler software, because only one address area is applicable for the receive process.

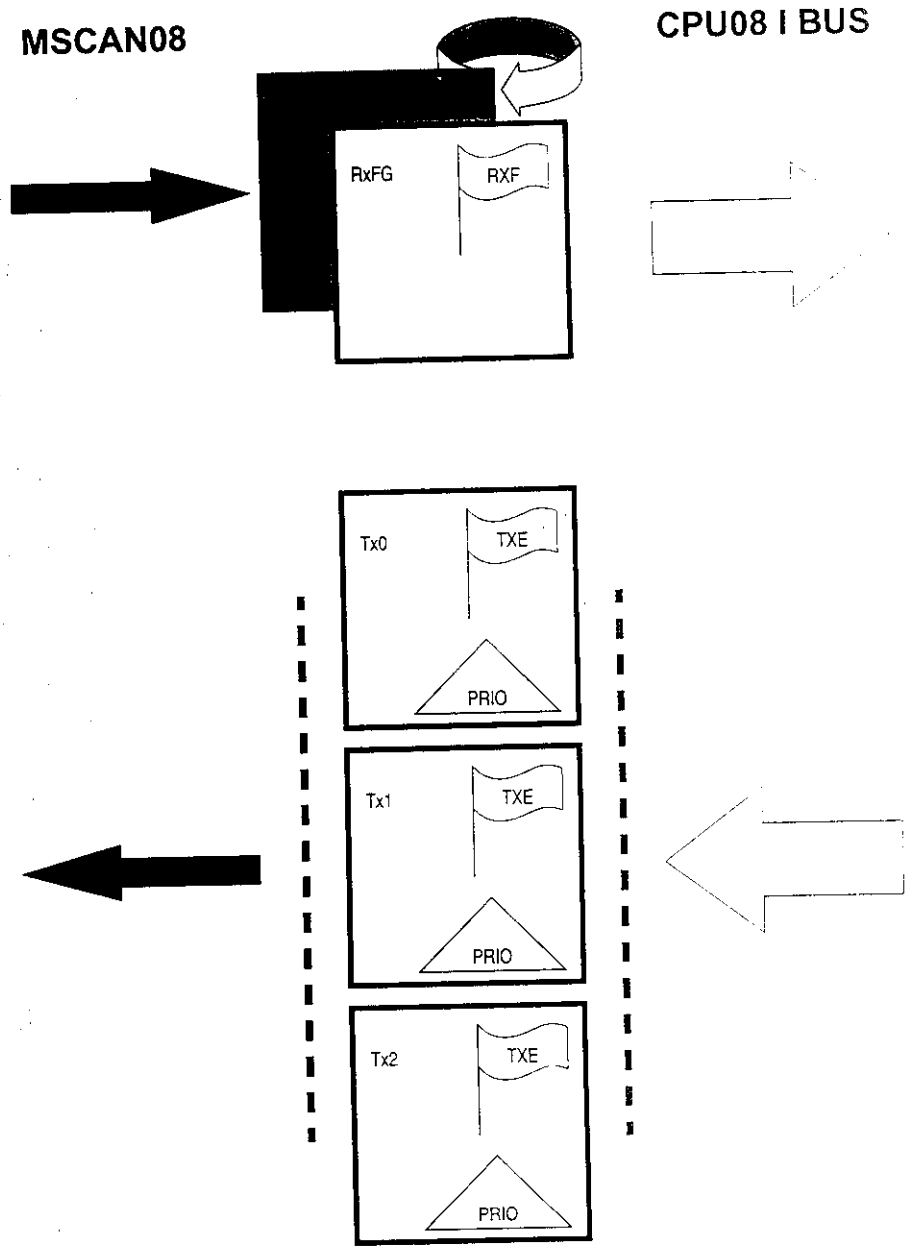


Figure 16-2. User Model for Message Buffer Organization

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended), and the data content. For details, see **16.12 Programmer's Model of Message Storage**.

The receiver full flag (RXF) in the MSCAN08 receiver flag register (CRFLG), signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set. See **16.13.5 MSCAN08 Receiver Flag Register (CRFLG)**

On reception, each message is checked to see if it passes the filter (for details see **16.5 Identifier Acceptance Filter**) and in parallel is written into RxBG. The MSCAN08 copies the content of RxBG into RxFG[1], sets the RXF flag, and generates a receive interrupt to the CPU[2]. The user's receive handler has to read the received message from RxFG and to reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message which can follow immediately after the IFS field of the CAN frame, is received into RxBG. The overwriting of the background buffer is independent of the identifier filter function.

When the MSCAN08 module is transmitting, the MSCAN08 receives its own messages into the background receive buffer, RxBG. It does NOT overwrite RxFG, generate a receive interrupt or acknowledge its own messages on the CAN bus. The exception to this rule is in loop-back mode (see **16.13.2 MSCAN08 Module Control Register 1**), where the MSCAN08 treats its own messages exactly like all other incoming messages. The MSCAN08 receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN08 must be prepared to become the receiver.

An overrun condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message will be discarded and an error interrupt with overrun indication will be generated if enabled. The MSCAN08 is still able to transmit messages with both receive message buffers filled, but all incoming messages are discarded.

16.4.3 Transmit Structures

The MSCAN08 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in **Figure 16-2**.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see **16.12 Programmer's Model of Message Storage**). An additional transmit buffer priority register (TBPR) contains an 8-bit "local priority" field (PRIO) (see **16.12.5 Transmit Buffer Priority Registers**).

---

1. Only if the RXF flag is not set.
2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

To transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the MSCAN08 transmitter flag register (CTFLG) (see **16.13.7 MSCAN08 Transmitter Flag Register**).

The CPU08 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged ready for transmission by clearing the TXE flag.

The MSCAN08 then will schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt is generated(1) when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN08 uses the local priority setting of the three buffers for prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined as the highest priority.

The internal scheduling process takes place whenever the MSCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission cannot be aborted, the user has to request the abort by setting the corresponding abort request flag (ABTRQ) in the transmission control register (CTCR). The MSCAN08 will then grant the request, if possible, by setting the corresponding abort request acknowledge (ABTAK) and the TXE flag in order to release the buffer and by generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK = 1) or sent (ABTAK = 0).

16.5 Identifier Acceptance Filter

The identifier acceptance registers (CIDAR0–CIDAR3) define the acceptance patterns of the standard or extended identifier (ID10–ID0 or ID28–ID0). Any of these bits can be marked 'don't care' in the identifier mask registers (CIDMR0–CIDMR3).

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

A filter hit is indicated to the application on software by a set RXF (receive buffer full flag, see **16.13.5 MSCAN08 Receiver Flag Register (CRFLG)**) and two bits in the identifier acceptance control register (see **16.13.9 MSCAN08 Identifier Acceptance Control Register**). These identifier hit flags (IDHIT1 and IDHIT0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

1. Single identifier acceptance filter, each to be applied to a) the full 29 bits of the extended identifier and to the following bits of the CAN frame: RTR, IDE, SRR or b) the 11 bits of the standard identifier plus the RTR and IDE bits of CAN 2.0A/B messages. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier. **Figure 16-3** shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces a filter 0 hit.
2. Two identifier acceptance filters, each to be applied to:
   a. The 14 most significant bits of the extended identifier plus the SRR and the IDE bits of CAN2.0B messages, or
   b. The 11 bits of the identifier plus the RTR and IDE bits of CAN 2.0A/B messages.

   Figure 16-4 shows how the 32-bit filter bank (CIDAR0–CIDAR3 and CIDMR0–CIDMR3) produces filter 0 and 1 hits.
3. Four identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements four independent filters for the first eight bits of a CAN 2.0A/B compliant standard identifier. **Figure 16-5** shows how the 32-bit filter bank (CIDAR0–CIDAR3 and CIDMR0–CIDMR3) produces filter 0 to 3 hits.
4. Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

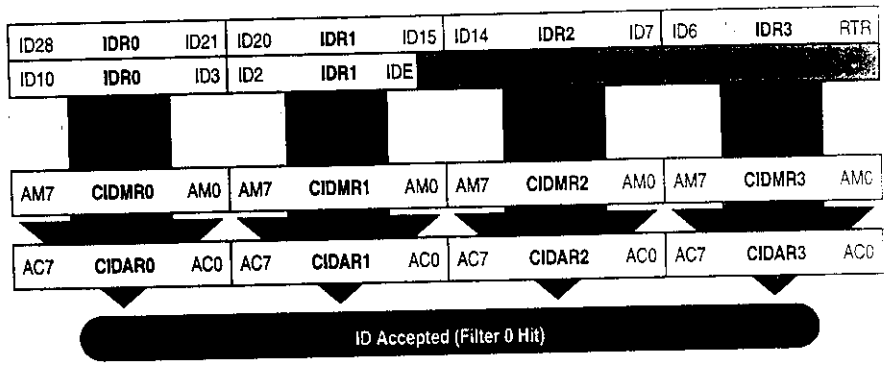


Figure 16-3. Single 32-Bit Maskable Identifier Acceptance Filter

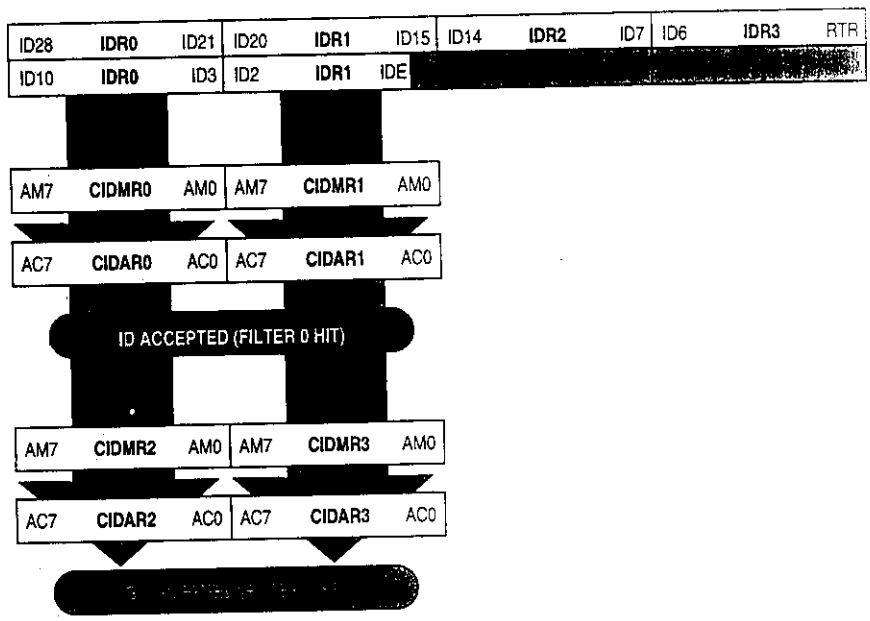


Figure 16-4. Dual 16-Bit Maskable Acceptance Filters

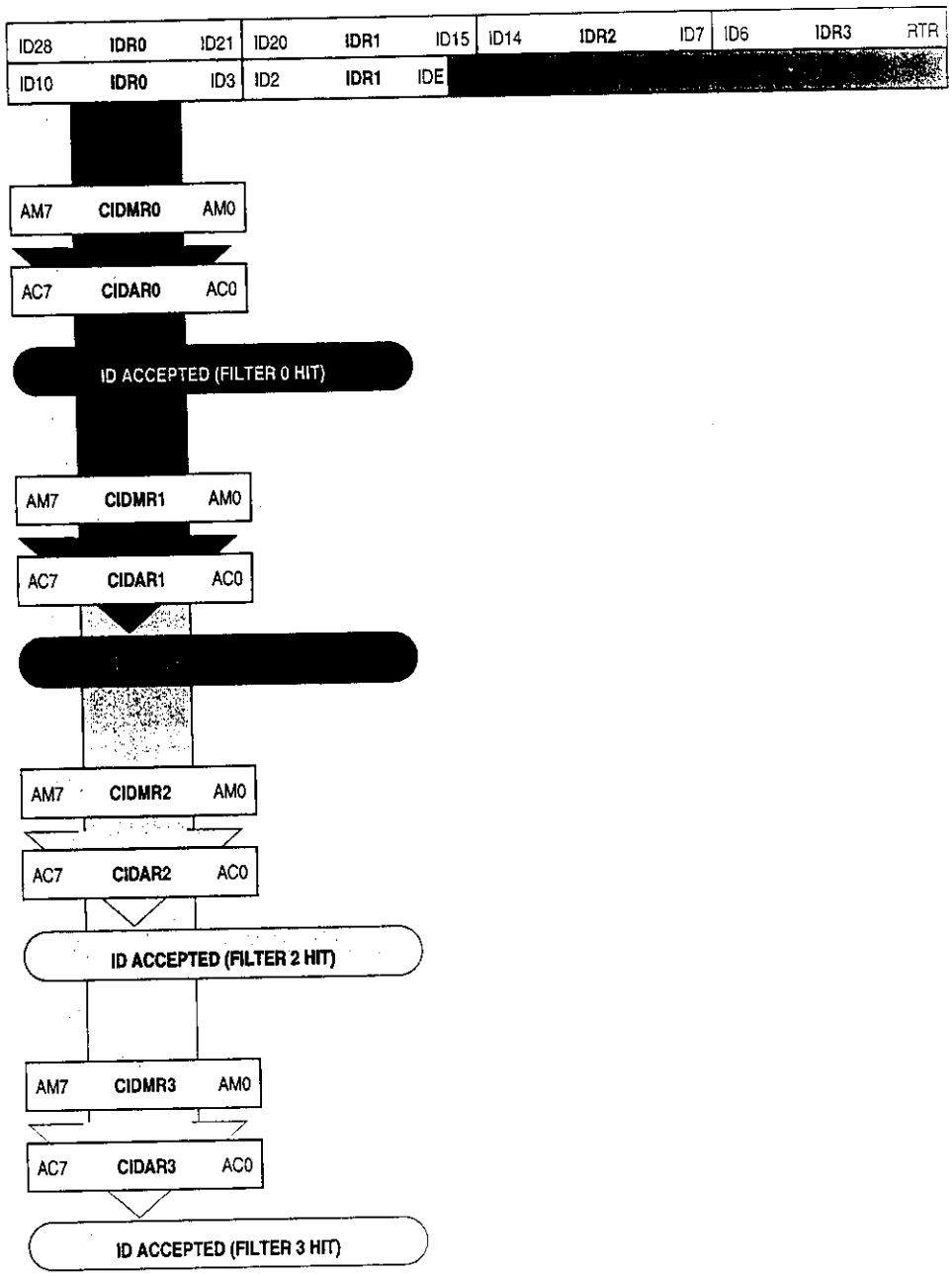


Figure 16-5. Quadruple 8-Bit Maskable Acceptance Filters

16.6.2 Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in **Table 16-1**. The vector addresses and the relative interrupt priority are dependent on the chip integration and to be defined.

Table 16-1. MSCAN08 Interrupt Vector Addresses

Function	Source	Local Mask	Global Mask
Wakeup	WUPIF	WUPIE	I bit
Error interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

16.7 Protocol Violation Protection

The MSCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 module control register (see **16.13.1 MSCAN08 Module Control Register 0**) serves as a lock to protect the following registers:
  - MSCAN08 module control register 1 (CMCR1)
  - MSCAN08 bus timing register 0 and 1 (CBTR0 and CBTR1)
  - MSCAN08 identifier acceptance control register (CIDAC)
  - MSCAN08 identifier acceptance registers (CIDAR0–3)
  - MSCAN08 identifier mask registers (CIDMR0–3)
- The CAN_{TX} pin is forced to recessive when the MSCAN08 is in any of the low-power modes.

16.8 Low-Power Modes

In addition to normal mode, the MSCAN08 has three modes with reduced power consumption: sleep, soft reset, and power down. In sleep and soft reset mode, power consumption is reduced by stopping all clocks except those to access the registers. In power-down mode, all clocks are stopped and no power is consumed.

The WAIT and STOP instructions put the MCU in low-power consumption stand-by modes. **Table 16-2** summarizes the combinations of MSCAN08 and CPU modes. A particular combination of modes is entered for the given settings of the bits SLPAK and SFTRES. For all modes, an MSCAN08 wakeup interrupt can occur only if SLPAK = WUPIE = 1.

Table 16-2. MSCAN08 versus CPU Operating Modes

MSCAN08 Mode	CPU Mode	
	STOP	WAIT or RUN
Power Down	SLPAK = X(1) SFTRES = X	
Sleep		SLPAK = 1 SFTRES = 0
Soft Reset		SLPAK = 0 SFTRES = 1
Normal		SLPAK = 0 SFTRES = 0

1. 'X' means don't care.

16.8.1 MSCAN08 Sleep Mode

The CPU can request the MSCAN08 to enter the low-power mode by asserting the SLPRQ bit in the module configuration register (see **Figure 16-6**). The time when the MSCAN08 enters sleep mode depends on its activity:

- If it is transmitting, it continues to transmit until there is no more message to be transmitted, and then goes into sleep mode
- If it is receiving, it waits for the end of this message and then goes into sleep mode
- If it is neither transmitting or receiving, it will immediately go into sleep mode

NOTE: *The application software must avoid setting up a transmission (by clearing or more TXE flags) and immediately request sleep mode (by setting SLPRQ). It then depends on the exact sequence of operations whether MSCAN08 starts transmitting or goes into sleep mode directly.*

During sleep mode, the SLPAK flag is set. The application software should use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode. When in sleep mode, the MSCAN08 stops its internal clocks. However, clocks to allow register accesses still run. If the MSCAN08 is in bus-off state, it stops counting the 128*11 consecutive recessive bits due to the stopped clocks. The CAN_{TX} pin stays in recessive state. If RXF = 1, the message can be read and RXF can be cleared. Copying of RxGB into RxFG doesn't take place while in sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort takes place while in sleep mode.

The MSCAN08 leaves sleep mode (wakes-up) when:

- Bus activity occurs, or
- The MCU clears the SLPRQ bit, or
- The MCU sets the SFTRES bit

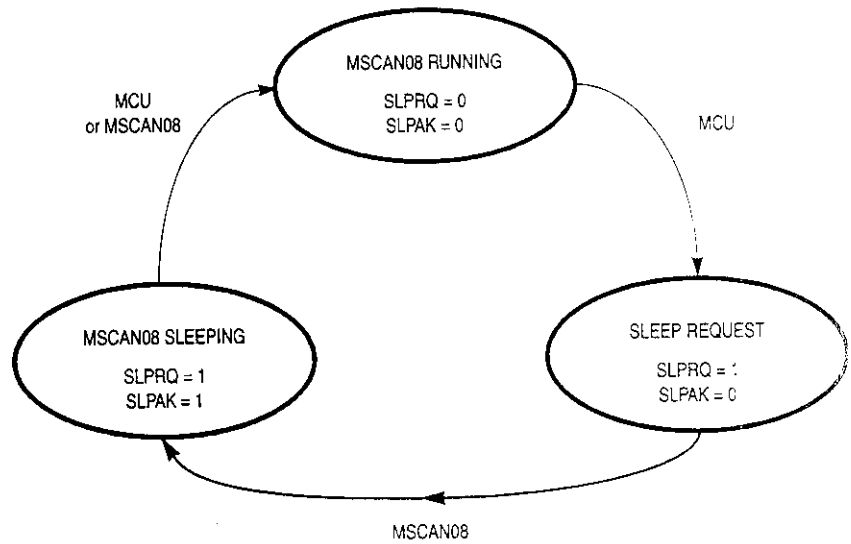


Figure 16-6. Sleep Request/Acknowledge Cycle

NOTE: *The MCU cannot clear the SLPRQ bit before the MSCAN08 is in sleep mode (SLPAK=1).*

After wakeup, the MSCAN08 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN08 is woken-up by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions are executed upon wakeup: copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN08 is still in bus-off state after sleep mode was left, it continues counting the 128*11 consecutive recessive bits.

16.8.2 MSCAN08 Soft Reset Mode

In soft reset mode, the MSCAN08 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing and the CAN message filter. See **16.13.1 MSCAN08 Module Control Register 0** for a complete description of the soft reset mode.

When setting the SFTRES bit, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

NOTE: *The user is responsible to take care that the MSCAN08 is not active when soft reset mode is entered. The recommended procedure is to bring the MSCAN08 into sleep mode before the SFTRES bit is set.*

16.8.3 MSCAN08 Power-Down Mode

The MSCAN08 is in power-down mode when the CPU is in stop mode.

When entering the power-down mode, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

NOTE: *The user is responsible to take care that the MSCAN08 is not active when power-down mode is entered. The recommended procedure is to bring the MSCAN08 into sleep mode before the STOP instruction is executed.*

To protect the CAN bus system from fatal consequences resulting from violations of the above rule, the MSCAN08 drives the CAN_{TX} pin into recessive state.

In power-down mode, no registers can be accessed.

MSCAN08 bus activity can wake the MCU from CPU stop/MSCAN08 power-down mode. However, until the oscillator starts up and synchronization is achieved the MSCAN08 will not respond to incoming data.

16.8.4 CPU Wait Mode

The MSCAN08 module remains active during CPU wait mode. The MSCAN08 will stay synchronized to the CAN bus and generates transmit, receive, and error interrupts to the CPU, if enabled. Any such interrupt will bring the MCU out of wait mode.

16.8.5 Programmable Wakeup Function

The MSCAN08 can be programmed to apply a low-pass filter function to the CAN_{RX} input line while in internal sleep mode (see information on control bit WUPM in **16.13.2 MSCAN08 Module Control Register 1**). This feature can be used to protect the MSCAN08 from wakeup due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

16.9 Timer Link

The MSCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN08 receiver engine also receives the frames being sent by itself, a timer signal also will be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip 2-channel timer interface module 2 (TIM2). This signal is connected to the TIM2 channel 0 input under the control of the timer link enable (TLNKEN) bit in CMCR0.

After the timer has been programmed to capture rising edge events, it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

16.10 Clock System

Figure 16-7 shows the structure of the MSCAN08 clock generation circuitry and its interaction with the clock generation module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

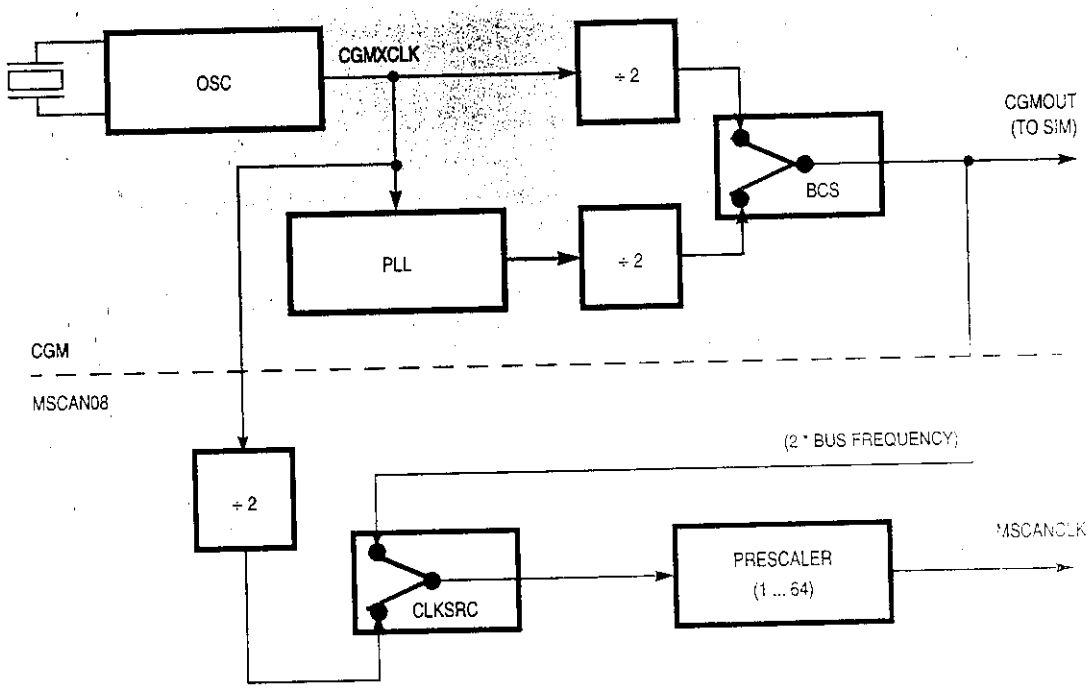


Figure 16-7. Clocking Scheme

The clock source bit (CLKSRC) in the MSCAN08 module control register (CMCR1) (see **16.13.1 MSCAN08 Module Control Register 0**) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the PLL output.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met.

NOTE: *If the system clock is generated from a PLL, it is recommended to select the crystal clock source rather than the system clock source due to jitter considerations, especially at faster CAN bus rates.*

A programmable prescaler is used to generate out of the MSCAN08 clock the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN08.

$$f_{Tq} = \frac{f_{MSCANCLK}}{\text{Presc value}}$$

A bit time is subdivided into three segments[1] (see **Figure 16-8**):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit rate} = \frac{f_{Tq}}{\text{No. of time quanta}}$$

The synchronization jump width (SJW) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The above parameters can be set by programming the bus timing registers. CBTR0 and CBTR1. See **16.13.3 MSCAN08 Bus Timing Register 0** and 16.13.4 **MSCAN08 Bus Timing Register 1**.

NOTE: *It is the user's responsibility to make sure that the bit timing settings are in compliance with the CAN standard,*

Table 16-8 gives an overview on the CAN conforming segment settings and the related parameter values.

1. For further explanation of the underlying concepts please refer to ISO/DIS 11 519-1, Section 10.3.

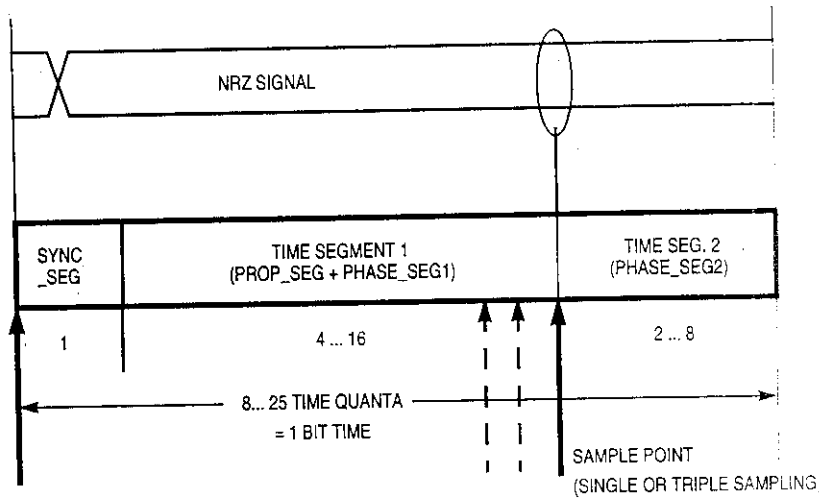


Figure 16-8. Segments Within the Bit Time

Table 16-3. Time Segment Syntax

SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit point	A node in transmit mode will transfer a new value to the CAN bus at this point.
Sample point	A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample.

Table 16-4. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronized Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

16.11 Memory Map

The MSCAN08 occupies 128 bytes in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128.

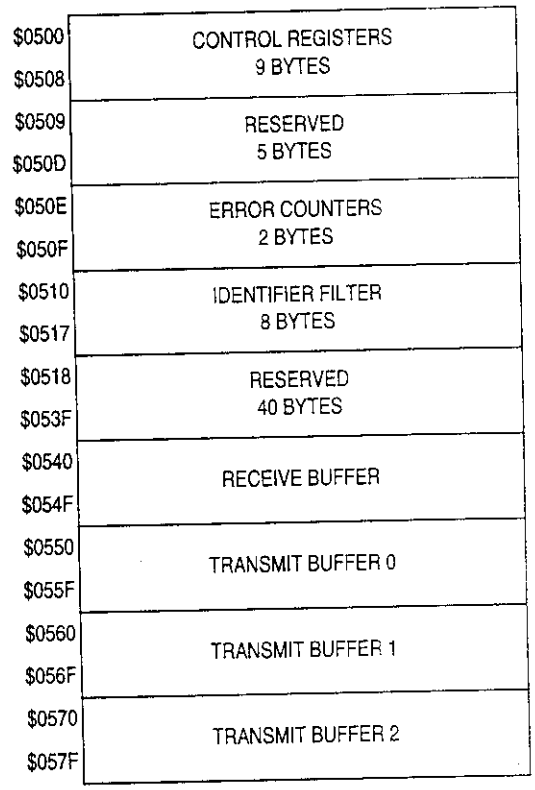


Figure 16-9. MSCAN08 Memory Map

16.12 Programmer's Model of Message Storage

This section details the organization of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure. An additional transmit buffer priority register (TBPR) is defined for the transmit buffers.

Addr[1]	Register Name
$05b0	IDENTIFIER REGISTER 0
$05b1	IDENTIFIER REGISTER 1
$05b2	IDENTIFIER REGISTER 2
$05b3	IDENTIFIER REGISTER 3
$05b4	DATA SEGMENT REGISTER 0
$05b5	DATA SEGMENT REGISTER 1
$05b6	DATA SEGMENT REGISTER 2
$05b7	DATA SEGMENT REGISTER 3
$05b8	DATA SEGMENT REGISTER 4
$05b9	DATA SEGMENT REGISTER 5
$05bA	DATA SEGMENT REGISTER 6
$05bB	DATA SEGMENT REGISTER 7
$05bC	DATA LENGTH REGISTER
$05bD	TRANSMIT BUFFER PRIORITY REGISTER[2]
$05bE	UNUSED
$05bF	UNUSED

1. Where b equals the following:
   b=4 for receive buffer
   b=5 for transmit buffer 0
   b=6 for transmit buffer 1
   b=7 for transmit buffer 2
2. Not applicable for receive buffers

Figure 16-10. Message Buffer Organization

16.12.1 Message Buffer Outline

Figure 16-11 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in **Figure 16-12**. All bits of the 13-byte data structure are undefined out of reset.

NOTE: *The foreground receive buffer can be read anytime but cannot be written. The transmit buffers can be read or written anytime.*

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$05b0	IDR0	Read: Write:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
$05b1	IDR1	Read: Write:	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
$05b2	IDR2	Read: Write:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
$05b3	IDR3	Read: Write:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
$05b4	DSR0	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b5	DSR1	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b6	DSR2	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b7	DSR3	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b8	DSR4	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b9	DSR5	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05bA	DSR6	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05bB	DSR7	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05bC	DLR	Read: Write:					DLC3	DLC2	DLC1	DLC0

= Unimplemented

Figure 16-11. Receive/Transmit Message Buffer Extended Identifier (IDRn)

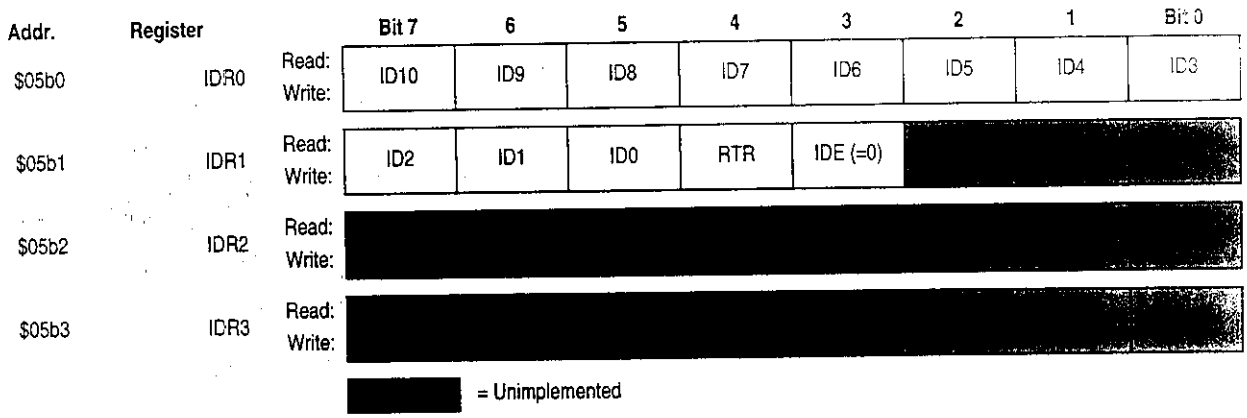


Figure 16-12. Standard Identifier Mapping

16.12.2 Identifier Registers

The identifiers consist of either 11 bits (ID10–ID0) for the standard, or 29 bits (ID28–ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

SRR — Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer, the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer, the flag indicates to the MSCAN08 what type of identifier to send.

1 = Extended format, 29 bits
0 = Standard format, 11 bits

RTR — Remote Transmission Request

This flag reflects the status of the remote transmission request bit in the CAN frame. In case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

1 = Remote frame
0 = Data frame

16.12.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

DLC3–DLC0 — Data Length Code Bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. **Table 16-5** shows the effect of setting the DLC bits.

Table 16-5. Data Length Codes

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

16.12.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

16.12.5 Transmit Buffer Priority Registers

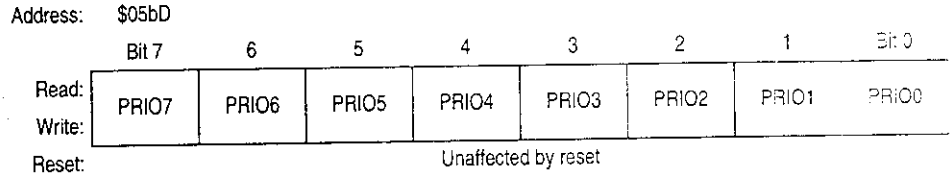


Figure 16-13. Transmit Buffer Priority Register (TBPR)

PRIO7–PRIO0 — Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN08 and is

defined to be highest for the smallest binary number. The MSCAN08 implements the following internal prioritization mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritization right before the SOF is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.
- In case more than one buffer has the same lowest priority, the message buffer with the lower index number wins.

16.13 Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency. **Figure 16-14** gives an overview on the control register block of the MSCAN08.

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$0500	CMCR0	Read:	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		Write:								
$0501	CMCR1	Read:	0	0	0	0	0	LOOPB	WUPM	CLKSRC
$0502	CBTR0	Read:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		Write:								
$0503	CBTR1	Read:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		Write:								
$0504	CRFLG	Read:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		Write:								
$0505	CRIER	Read:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		Write:								
$0506	CTFLG	Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		Write:								
$0507	CTCR	Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		Write:								
$0508	CIDAC	Read:	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
		Write:								
$0509	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$050E	CRXERR	Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		Write:								

= Unimplemented R = Reserved

Figure 16-14. MSCAN08 Control Register Structure

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$050F	CTXERR	Read: Write:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
$0510	CIDAR0	Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0511	CIDAR1	Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0512	CIDAR2	Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0513	CIDAR3	Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0514	CIDMR0	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
$0515	CIDMR1	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
$0516	CIDMR2	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
$0517	CIDMR3	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0

= Unimplemented R = Reserved

Figure 16-14. MSCAN08 Control Register Structure (Continued)

16.13.1 MSCAN08 Module Control Register 0

Address: $0500

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-15. Module Control Register 0 (CMCR0)

SYNCH — Synchronized Status

This bit indicates whether the MSCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 = MSCAN08 synchronized to the CAN bus

0 = MSCAN08 not synchronized to the CAN bus

TLNKEN — Timer Enable

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see **16.9 Timer Link**).

1 = The MSCAN08 timer signal output is connected to the timer input.

0 = The port is connected to the timer input.

SLPAK — Sleep Mode Acknowledge

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see **16.8.1 MSCAN08 Sleep Mode**). If the MSCAN08 detects bus activity while in sleep mode, it clears the flag.

1 = Sleep – MSCAN08 in internal sleep mode

0 = Wakeup – MSCAN08 is not in sleep mode

SLPRQ — Sleep Request, Go to Internal Sleep Mode

This flag requests the MSCAN08 to go into an internal power-saving mode (see **16.8.1 MSCAN08 Sleep Mode**).

1 = Sleep — The MSCAN08 will go into internal sleep mode.

0 = Wakeup — The MSCAN08 will function normally.

SFTRES — Soft Reset

When this bit is set by the CPU, the MSCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers enter and stay in their hard reset state:

CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0–CIDAR3, and CIDMR0–CIDMR3 can only be written by the CPU when the MSCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN08 tries to synchronize to the CAN bus. If the MSCAN08 is not in bus-off state, it will be synchronized after 11 recessive bits on the bus; if the MSCAN08 is in bus-off state, it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 = MSCAN08 in soft reset state

0 = Normal operation

16.13.2 MSCAN08 Module Control Register 1

Figure 16-16. Module Control Register (CMCR1)

LOOPB — Loop Back Self-Test Mode

When this bit is set, the MSCAN08 performs an internal loop back which can be used for self-test operation: the bit stream output of the transmitter is fed back to the receiver internally. The CAN_{RX} input pin is ignored and the CAN_{TX} output goes to the recessive state (logic 1). The MSCAN08 behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state the MSCAN08 ignores the bit sent during the ACK slot of the CAN frame Acknowledge field to insure proper reception of its own message. Both transmit and receive interrupts are generated.

1 = Activate loop back self-test mode
0 = Normal operation

WUPM — Wakeup Mode

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wakeups (see **16.8.5 Programmable Wakeup Function**).

1 = MSCAN08 will wakeup the CPU only in cases of a dominant pulse on the bus which has a length of at least t_{wup}.
0 = MSCAN08 will wakeup the CPU after any recessive-to-dominant edge on the CAN bus.

CLKSRC — Clock Source

This flag defines which clock source the MSCAN08 module is driven from (see **16.10 Clock System**).

1 = The MSCAN08 clock source is CGMOUT (see **Figure 16-7**).
0 = The MSCAN08 clock source is CGMXCLK/2 (see **Figure 16-7**).

NOTE: *The CMCR1 register can be written only if the SFTRES bit in the MSCAN08 module control register is set*

16.13.3 MSCAN08 Bus Timing Register 0

Address: $0502

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Reset:	0	0	0	0	0	0	0	0

Figure 16-17. Bus Timing Register 0 (CBTR0)

SJW1 and SJW0 — Synchronization Jump Width

The synchronization jump width (SJW) defines the maximum number of time quanta (T_q) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 16-6).

Table 16-6. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 T_q cycle
0	1	2 T_q cycle
1	0	3 T_q cycle
1	1	4 T_q cycle

BRP5–BRP0 — Baud Rate Prescaler

These bits determine the time quanta (T_q) clock, which is used to build up the individual bit timing, according to Table 16-7.

Table 16-7. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler Value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	1	1	64

NOTE: *The CBTR0 register can be written only if the SFTRES bit in the MSCAN08 module control register is set.*

16.13.4 MSCAN08 Bus Timing Register 1

Address: $0503

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
Reset:	0	0	0	0	0	0	0	0

Figure 16-18. Bus Timing Register 1 (CBTR1)

SAMP — Sampling

This bit determines the number of serial bus samples to be taken per bit time. If set, three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates, SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit(1)
0 = One sample per bit

TSEG22–TSEG10 — Time Segment

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in **Table 16-8**.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (T_q) clock cycles per bit as shown in Table 16-4).

$$\text{Bit time} = \frac{\text{Pres value}}{f_{\text{MSCANCLK}}} \cdot \text{number of time quanta}$$

NOTE: *The CBTR1 register can only be written if the SFTRES bit in the MSCAN08 module control register is set.*

Table 16-8. Time Segment Values

TSEG13	TSEG12	TSEG11	TSEG10	Time Segment 1
0	0	0	0	1 T_q Cycle(1)
0	0	0	1	2 T_q Cycles(1)
0	0	1	0	3T_q Cycles(1)
0	0	1	1	4 T_q Cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 T_q Cycles

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 T_q Cycle(1)
0	0	1	2 T_q Cycles
.	.	.	.
.	.	.	.
1	1	1	8T_q Cycles

1. This setting is not valid. Please refer to **Table 16-4** for valid settings.

1. In this case PHASE_SEG1 must be at least 2 time quanta.

16.13.5 MSCAN08 Receiver Flag Register (CRFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can be cleared only when the condition which caused the setting is valid no more. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

Address: $0504

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
Reset:	0	0	0	0	0	0	0	0

Figure 16-19. Receiver Flag Register (CRFLG)

WUPIF — Wakeup Interrupt Flag
If the MSCAN08 detects bus activity while in sleep mode, it sets the WUPIF flag. If not masked, a wakeup interrupt is pending while this flag is set.
1 = MSCAN08 has detected activity on the bus and requested wakeup.
0 = No wakeup interrupt has occurred.

RWRNIF — Receiver Warning Interrupt Flag
This flag is set when the MSCAN08 goes into warning status due to the receive error counter (REC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(1). If not masked, an error interrupt is pending while this flag is set.
1 = MSCAN08 has gone into receiver warning status.
0 = No receiver warning status has been reached.

TWRNIF — Transmitter Warning Interrupt Flag
This flag is set when the MSCAN08 goes into warning status due to the transmit error counter (TEC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(2). If not masked, an error interrupt is pending while this flag is set.
1 = MSCAN08 has gone into transmitter warning status.
0 = No transmitter warning status has been reached.

RERRIF — Receiver Error Passive Interrupt Flag
This flag is set when the MSCAN08 goes into error passive status due to the receive error counter exceeding 127 and the bus-off interrupt flag is not set(3). If not masked, an error interrupt is pending while this flag is set.
1 = MSCAN08 has gone into receiver error passive status.
0 = No receiver error passive status has been reached.

1. Condition to set the flag: RWRNIF = (96 ð REC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
2. Condition to set the flag: TWRNIF = (96 ð TEC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
3. Condition to set the flag: RERRIF = (127 ð REC ð 255) & $\overline{\text{BOFFIF}}$

TERRIF — Transmitter Error Passive Interrupt Flag

This flag is set when the MSCAN08 goes into error passive status due to the transmit error counter exceeding 127 and the bus-off interrupt flag is not set[1]. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into transmit error passive status.

0 = No transmit error passive status has been reached.

BOFFIF — Bus-Off Interrupt Flag

This flag is set when the MSCAN08 goes into bus-off status, due to the transmit error counter exceeding 255. It cannot be cleared before the MSCAN08 has monitored 128 times 11 consecutive 'recessive' bits on the bus. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08has gone into bus-off status.

0 = No bus-off status has been reached.

OVRIF — Overrun Interrupt Flag

This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.

1 = A data overrun has been detected since last clearing the flag.

0 = No data overrun has occurred.

RXF — Receive Buffer Full

The RXF flag is set by the MSCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.

0 = The receive buffer is released (not full).

NOTE: *To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.*

The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

1. Condition to set the flag: TERRIF = (128 ð TEC ð 255) & $\overline{\text{BOFFIF}}$

16.13.6 MSCAN08 Receiver Interrupt Enable Register

Address: $0505

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
Reset:	0	0	0	0	0	0	0	0

Figure 16-20. Receiver Interrupt Enable Register (CRIER)

WUPIE — Wakeup Interrupt Enable
1 = A wakeup event will result in a wakeup interrupt.
0 = No interrupt will be generated from this event.

RWRNIE — Receiver Warning Interrupt Enable
1 = A receiver warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable
1 = A transmitter warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable
1 = A receiver error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable
1 = A transmitter error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

BOFFIE — Bus-Off Interrupt Enable
1 = A bus-off event will result in an error interrupt.
0 = No interrupt is generated from this event.

OVRIE — Overrun Interrupt Enable
1 = An overrun event will result in an error interrupt.
0 = No interrupt is generated from this event.

RXFIE — Receiver Full Interrupt Enable
1 = A receive buffer full (successful message reception) event will result in a receive interrupt.
0 = No interrupt will be generated from this event.

NOTE: *The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.*

16.13.7 MSCAN08 Transmitter Flag Register

The abort acknowledge flags are read only. The transmitter buffer empty flags are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. The transmitter buffer empty flags each have an associated interrupt enable bit in the CTCR register. A hard or soft reset will resets the register.

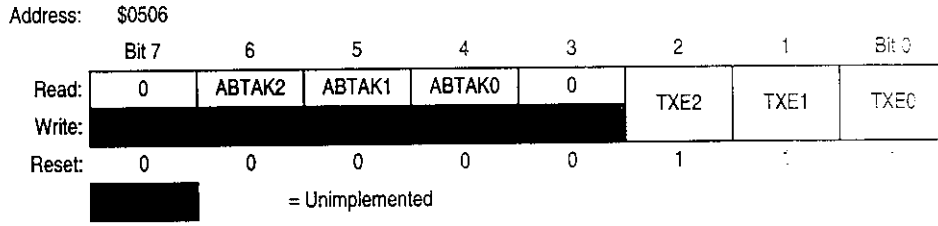


Figure 16-21. Transmitter Flag Register (CTFLG)

ABTAK2–ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent. The ABTAKx flag is cleared implicitly whenever the corresponding TXE flag is cleared.

1 = The message has been aborted.

0 = The message has not been aborted, thus has been sent out.

TXE2–TXE0 — Transmitter Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 sets the flag after the message has been sent successfully. The flag is also set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (see **16.12.5 Transmit Buffer Priority Registers**). If not masked, a receive interrupt is pending while this flag is set.

Clearing a TXEx flag also clears the corresponding ABTAKx flag (ABTAK, see above). When a TXEx flag is set, the corresponding ABTRQx bit (ABTRQ) is cleared. See **16.13.8 MSCAN08 Transmitter Control Register**

1 = The associated message buffer is empty (not scheduled).

0 = The associated message buffer is full (loaded with a message due for transmission).

NOTE: *To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.*

The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

16.13.8 MSCAN08 Transmitter Control Register

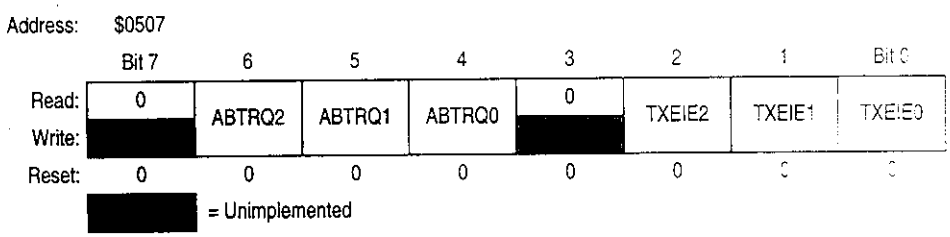


Figure 16-22. Transmitter Control Register (CTCR)

ABTRQ2–ABTRQ0 — Abort Request

The CPU sets an ABTRQx bit to request that an already scheduled message buffer (TXE = 0) be aborted. The MSCAN08 will grant the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted the associated TXE and the abort acknowledge flag (ABTAK) (see **16.13.7 MSCAN08 Transmitter Flag Register**) will be set and an TXE interrupt is generated if enabled. The CPU cannot reset ABTRQx. ABTRQx is cleared implicitly whenever the associated TXE flag is set.

1 = Abort request pending
0 = No abort request

NOTE: *The software must not clear one or more of the TXE flags in CTFLG and simultaneously set the respective ABTRQ bit(s).*

TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable

1 = A transmitter empty (transmit buffer available for transmission) event results in a transmitter empty interrupt.
0 = No interrupt is generated from this event.

NOTE: *The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.*

16.13.9 MSCAN08 Identifier Acceptance Control Register

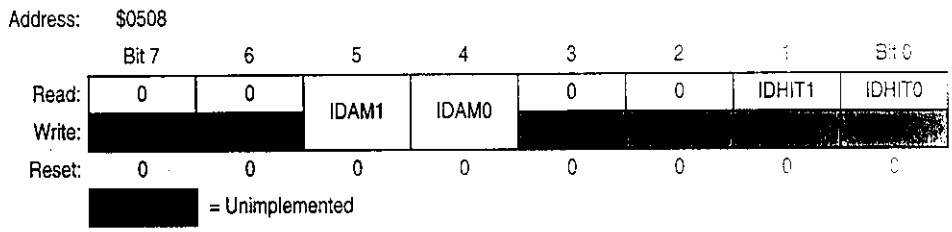


Figure 16-23. Identifier Acceptance Control Register (CIDAC)

IDAM1–IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organization (see **16.5 Identifier Acceptance Filter**). **Table 16-9** summarizes the different settings. In "filter closed" mode no messages will be accepted so that the foreground buffer will never be reloaded.

Table 16-9. Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Single 32-bit acceptance filter
0	1	Two 16-bit acceptance filter
1	0	Four 8-bit acceptance filters
1	1	Filter closed

IDHIT1–IDHIT0— Identifier Acceptance Hit Indicator

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see **16.5 Identifier Acceptance Filter**). **Table 16-9** summarizes the different settings.

Table 16-10. Identifier Acceptance Hit Indication

IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	Filter 0 hit
0	1	Filter 1 hit
1	0	Filter 2 hit
1	1	Filter 3 hit

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer, the indicators are updated as well.

NOTE: *The CIDAC register can be written only if the SFTRES bit in the CMCR0 is set.*

16.13.10 MSCAN08 Receive Error Counter

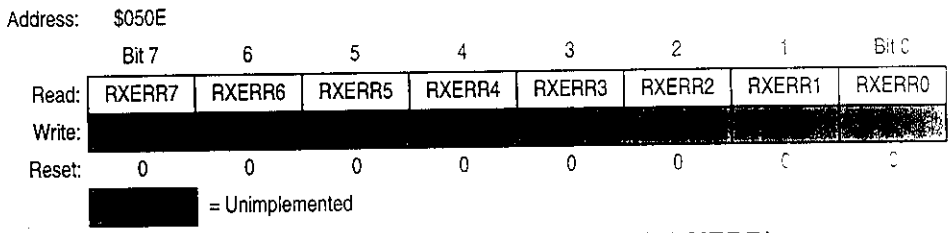


Figure 16-24. Receiver Error Counter (CRXERR)

This read-only register reflects the status of the MSCAN08 receive error counter.

16.13.11 MSCAN08 Transmit Error Counter

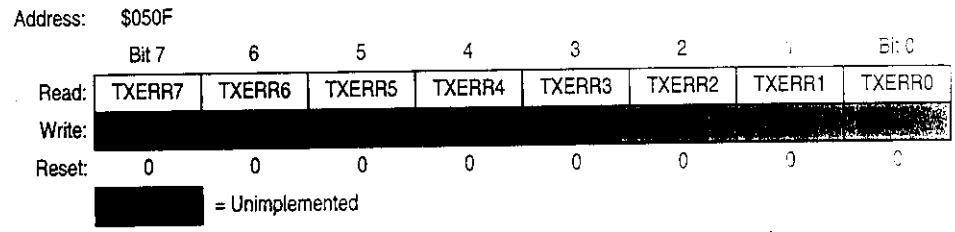


Figure 16-25. Transmit Error Counter (CTXERR)

This read-only register reflects the status of the MSCAN08 transmit error counter.

NOTE: *Both error counters may only be read when in sleep or soft reset mode.*

16.13.12 MSCAN08 Identifier Acceptance Registers

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message, however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the MSCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (CIDMR0/CIDMR1 and CIDAR0/CIDAR1) are applied.

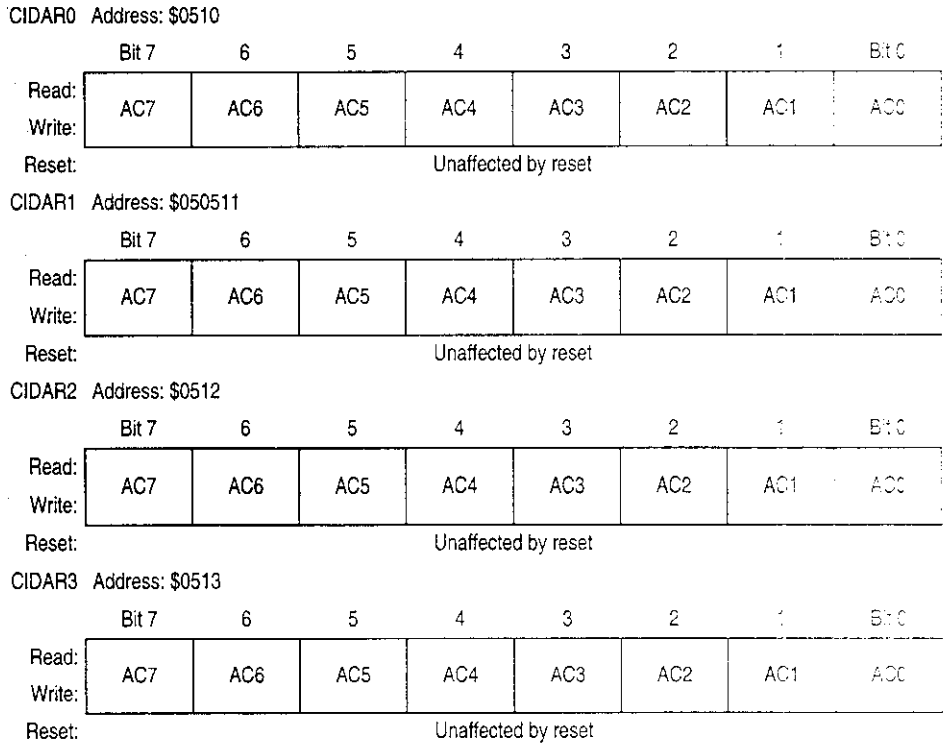


Figure 16-26. Identifier Acceptance Registers (CIDAR0–CIDAR3)

AC7–AC0 — Acceptance Code Bits

AC7–AC0 comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

NOTE: *The CIDAR0–CIDAR3 registers can be written only if the SFTRES bit in CMCR0 is set*

16.13.13 MSCAN08 Identifier Mask Registers (CIDMR0–CIDMR3)

The identifier mask registers specify which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. For standard identifiers it is required to program the last three bits (AM2–AM0) in the mask register CIDMR1 to 'don't care'.

CIDMR0 Address: $0514

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

CIDMR1 Address: $0515

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

CIDMR2 Address: $0516

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

CIDMR3 Address: $0517

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Figure 16-27. Identifier Mask Registers (CIDMR0–CIDMR3)

AM7–AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.
0 = Match corresponding acceptance code register and identifier bits.

NOTE: *The CIDMR0–CIDMR3 registers can be written only if the SFTRES bit in the CMCR0 is set*

SA5775 Air core meter driver applications information

AN1761

INTRODUCTION

The SA5775 is a monolithic Serial Gauge Driver (SGD) which can be used to directly drive Air Core Meters (ACM), typically used in automobile dashboards (Figure 1). The circuit interfaces with the serial bus of a microprocessor, has 10-bit resolution (0.35 degrees) and is guaranteed to be monotonic. Data can be shifted through the part so that several SA5775s can be cascaded with only one chip select line. On-board circuitry protects the circuit from external faults.

PRINCIPLES OF AIR CORE METERS

In many meter applications hitherto, D'Arsonval type of movements have been used. In this, a field coil is used along with a permanent magnet. When a current passes through the field coil, it generates a magnetic field. The magnetic field, due to the field coil, interacts with that due to the permanent magnet producing a resultant force that causes a needle to deflect, the deflection being controlled by a spring. Calibration of the current results in known deflections of the needle which is used to indicate the desired reading. While the advantage of such a system lies in its low cost and well-known behavior, there are also a number of disadvantages.

1. The strength of the mechanical spring degrades with age and with temperature cycling.
2. The permanent magnet ages with a lessening of the magnetic field. This makes it harder for it to pull the needle against the spring.
3. The mechanism is prone to damage due to vibrations from handling and other causes.

To overcome some of these problems, Air Core Meters have been used quite successfully. In this type of movement, there are two coils which are wound at 90° from one another (see Figures 2 and 3). A needle is attached to a magnetized disk that is positioned between the two coils. When currents 90° out-of-phase with each other pass through the two coils, the resultant magnetic fields produce a force which moves the disk. The position of the disk is dependent upon the ratio of the currents in the two coils. By varying this ratio, the disk and hence the needle position can be varied.

The advantage of such a type of movement lies in the fact that:

1. There are no springs
2. Only effect of magnetic strength degradation with age is in slowing of mechanical response time
3. Accuracy does not degrade with age
4. Not prone to damage due to vibration/handling on account of rugged sleeve bearings
5. Built-in damping

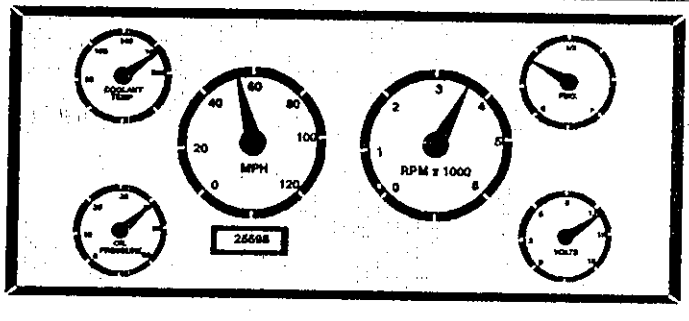


Figure 1. Analog Instrument Cluster

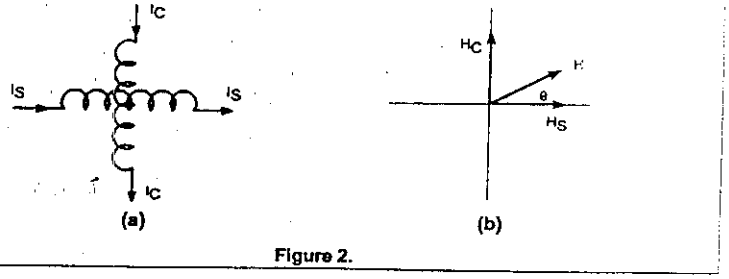


Figure 2.

THEORY OF OPERATION

In a solenoid, the strength of the magnetic field produced is given by

$$H = nI$$

where n = number of turns

I = current through the coil

When two coils are placed at right angles as shown in Figure 3 with the currents through the two coils being I_C and I_S, the resultant magnetic fields are

$$H_C = n\,I_C \quad (1a)$$

$$H_S = n\,I_S \quad (1b)$$

Then the magnitude of the resultant magnetic field H is given by the vector sum of H_C and H_S, namely

$$|H| = \sqrt{H_C^2 + H_S^2} \quad (2)$$

and the angle of the magnetic field is

$$\angle H = Arc\ \tan\left(\frac{H_S}{H_C}\right) \quad (3)$$

The currents I_C and I_S have the form and relationship as shown in Figure 4, so that

$$I_C = I\ Cos\left(\frac{360°}{1024}\ Code\right) \quad (4a)$$

$$I_S = I\ Sin\left(\frac{360°}{1024}\ Code\right) \quad (4b)$$

as 2^{10} bits are used for resolution of 360° and where code is the input code.

Substituting 4a and 4b in (1), (2) and (3) yields

$$|H| = n\,I \quad (5)$$

and

$$\angle H = Arc\ \tan\left(\frac{Sin\ \theta}{Cos\ \theta}\right) = \theta$$

where

$$\theta = \left(\frac{360°}{1024}\ Code\right) \quad (6)$$

From Equations (5) and (6), it is clear that the torque which is dependent upon $|H|$ is a constant if I is constant and θ is linearly dependent upon the input code.

The primary function of the SGD IC is to generate the transfer function that maps an input code into the correct voltages for linearly controlling the SIN and COS coils of an air core display. This circuit receives commands via an internal serial data interface port which is SPI compatible. The parts can be serially cascaded to minimize the necessary interface signals in multi-chip systems. The SGD has 10-bit resolution (0.35 degrees) and is guaranteed to be monotonic. The input data is directly proportional to the displayed angle in degrees. An input code of all 0's gives an output angle of 0.176 degrees; all 1's will generate a full-scale output of 359.82 degrees (see Table 2). The SGD output buffers are capable of sourcing up to 80mA per differential driver to control a single air core display directly.

FUNCTIONAL DESCRIPTION

The SA5775 which is housed in a 16-Pin package has terminal connections as shown in Figure 6. The function of each pin is described in Table 1.

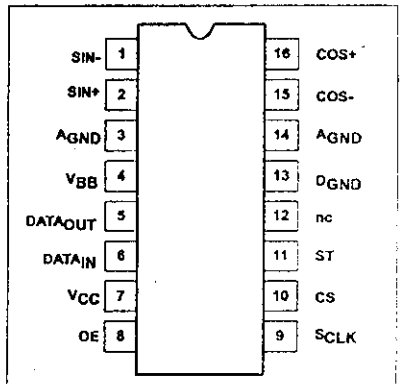


Figure 6. Pin Configuration

Table 1. SA5775 Pin Descriptions

Pin #	Name	Function
1	SIN–	Negative output connection to the SIN coil of the gauge.
2	SIN+	Positive output connection to the SIN coil of the gauge.
3	A_{GND}	Ground for V_{IGN} supply. Pins 3, 13 and 14 connected on the ckt board.
4	V_{BB}	Analog supply. Nominally 14.0V.
5	$DATA_{OUT}$	Serial data output. Output of the internal shift register. When a new data word is shifted in, the old word is shifted out the $DATA_{OUT}$ pin.
6	$DATA_{IN}$	Serial data input. A new data word is serially shifted into the part on the rising edge of S_{CLK}. The data is shifted in MSB first.
7	V_{CC}	5V logic supply. The internal latches and registers are set to zero on the rising edge of this signal.
8	OE	Output drivers are turned off when this input is low. Current draw is minimized.
9	S_{CLK}	Serial clock input. Data is loaded into the part on the rising edge of S_{CLK}.
10	CS	Active high chip select input. When CS is high, the part is enabled to receive a new serial input word. The high-to-low transition of CS loads the new 10-bit word into the DAC registers and updates the output.
11	ST	Status output from this IC to indicate that the outputs have been disabled. The outputs may be disabled due to shorted outputs, over temperature conditions, power up reset, or output enable control pin. This output is an open drain output. Multiple status outputs may be wire OR'ed together. This output is low when the outputs are disabled due to a fault condition.
12	nc	Not connected
13	D_{GND}	Ground for V_{CC} supply. Connect to Pins 3 and 14.
14	A_{GND}	Ground for V_{BB} supply. Connect to Pins 3 and 13.
15	COS–	Negative output connection to the COS coil of the gauge.
16	COS+	Positive output connection to the COS coil of the gauge.

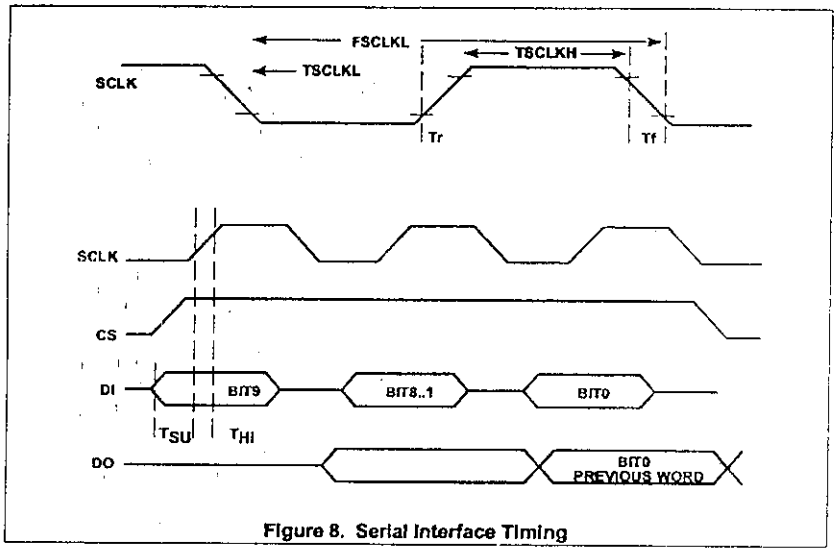


Figure 8. Serial Interface Timing

BIBLIOGRAPHY

- David A. Bell – " Electronic devices & circuits "
- Ayala – " 8051 Microcontroller " (for MAX232 module)
- D. Roy Choudhury, Shail Jain – " Linear Integrated Circuits " (for Power supply)
- MC68HC908GZ16 series manual
- http://www.texasinstruments.com
- http://ewww.motorola.com
- http://www.philips.com