TEMIC

MHS

29C462

Vehicle Area Network Data Link Controller Multiple Interfaces

Overview

The 29C462 data link controller multiple interfaces is fully compliant with the ISO recommendation ISO/TC22/SC3/WG1 revision 4.000.

The 29C462 handles all specific module types (Autonomous, synchronous and slave).

The 29C462 handles all specific message types (Data frame, reply request frame with deferred reply as well as reply request with immediate reply).

The 29C462 features a powerful CPU interface that offers flexibility to directly interface to many different CPUs. It can be configured to interface with CPUs using an 8-bit multiplexed, 16-bit multiplexed, or 8-bit non-multiplexed address/data bus for multiple architectures such as Intel, Motorola. A flexible serial interface is also available when a parallel CPU interface is not needed. This serial interface is fully compatible with the SPI protocol of Motorola

The 29C462 provides 14 Identifier registers with all bits individually maskable and 128 bytes of general purpose RAM.

The 29C462 includes an automatic diagnosis line selection to prevent fault such as open or short on the VAN bus.

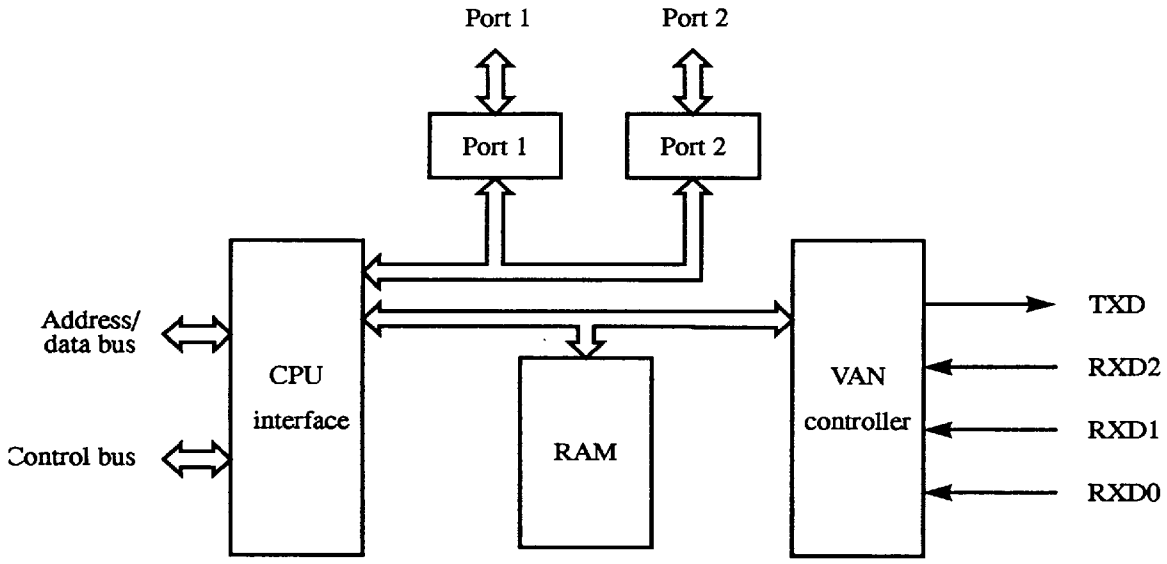
The 29C462 includes an integrated crystal or resonator oscillator with internal Baud rate generator and a buffered clock output.

The 29C462 is manufactured in 0.8µm, 5v technology and is available in 44-lead PLCC (The micro interface is pin-out compatible with Intel 82527) as well as 28-lead PLCC or 20 lead PLCC with reduced interface. Package such as 28 PLCC or 20 PLCC would be considered for dedicated micro interface if necessary.

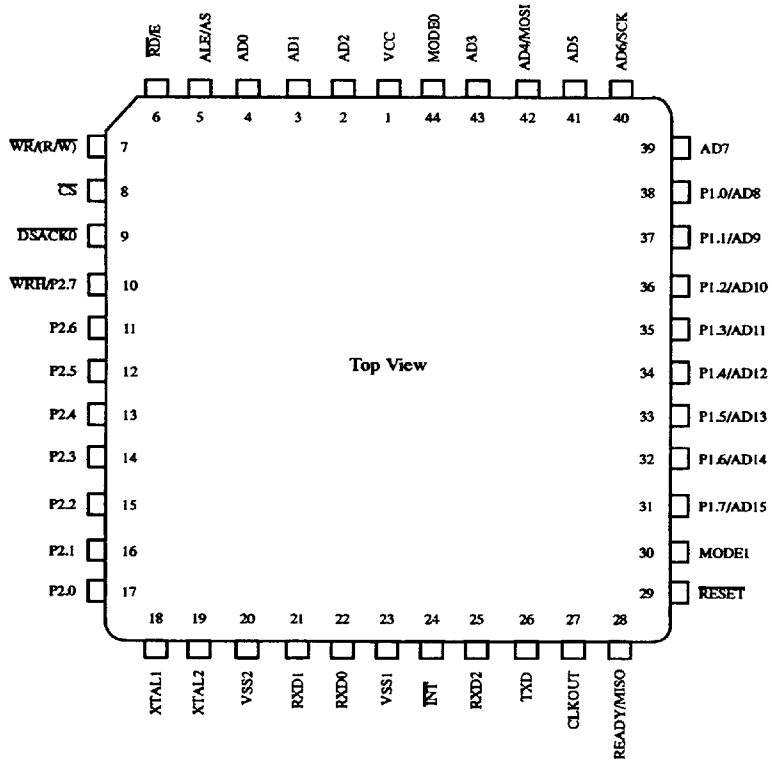
Features

- Fully Compliant with VAN specification ISO/TC22/SC3/WG1 Rev 4,000.
- 14 Identifier Registers with all bits individually maskable
- 1 Mbits/s Maximum Transfer Rate
- Flexible CPU interface
  - — 8 Bit Multiplexed
  - — 16 Bit Multiplexed
  - — 8 Bit Non-multiplexed (Synchronous/Asynchronous)
  - — Serial Interface
- 3 Seperate line inputs with automatic diagnosis and selection
- Idle and Sleep modes
- Manchester Enhanced or Impulsed Coding
- Two 8-bit bidirectional I/O ports
- 44 PLCC package (the micro interface is pin-out compatible with INTEL 82527)
- 28 package (reduced interface)
- 20 package (reduced interface)

1 — Block Diagram



2 — Pin Out and Package



2.1 — Pin Description

PIN	Nb	Direction	Type	Description
VSS1	23	Ground		Digital ground.
VSS2	20	Ground		Oscillator ground.
VCC	1	Power		Power for the entire chip.
XTAL1	18	I	CMOS	Either crystal connection (with XTAL2), or input for an external clock.
XTAL2	19	O	CMOS	Output from the internal oscillator. XTAL1 and XTAL2 are the crystal connections to an internal oscillator.
CLKOUT	27	O	CMOS	Programable clock output. This output can be used to drive the oscillator of the host microcontroller.
$\overline{\text{RESET}}$	29	I	TTL	A low level on this pin causes a hardware reset.
$\overline{\text{CS}}$	8	I	TTL	A low level on this pin enables the CPU to access the 29C462.
$\overline{\text{INT}}$	24	O/N	CMOS	Interrupt, this pin is an open drain, it requires external pullup resistor.
RXD0	22	I	CMOS PD	These three pins are the inputs with internal pull-down from the VAN bus driver.
RXD1	21			
RXD2	25			
TXD	26	O/Z	CMOS	Three state data output to the VAN bus driver. This output is in three state all along the IDLE mode.
AD0/CPOL AD1/CPHA AD2/CSAS AD3/STE AD4/MOSI AD5 AD6/SCK AD7	4 3 2 43 42 41 40 39	I/O	TTL	Address/Data bus in 8-bit multiplexed mode. Address bus in 8-bit non-multiplexed mode Low byte of Address/Data bus in 16-bit multiplexed mode. Serial interface: - AD0/CPOL : Idle Clock Polarity - AD1/CPHA : Clock Phase - AD2/CSAS: Chip Select Active State - AD3/STE : Sync Transmit Enable - AD4/MOSI : Serial Data Input - AD6/SCK : Serial Clock Input
AD8/P1.0 AD9/P1.1 AD10/P1.2 AD11/P1.3 AD12/P1.4 AD13/P1.5 AD14/P1.6 AD15/P1.7	38 37 36 35 34 33 32 31	I/O	TTL PU	Data bus in 8-bit non-multiplexed mode. High byte of Address/Data bus in 16-bit multiplexed mode. I/O port P1 in 8-bit multiplexed mode and serial mode.

PIN	Nb	Direction	Type	Description
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7/$\overline{WRH}$	17 16 15 14 13 12 11 10	I/O	TTL PU	I/O Port P2 in all modes. P2.7/$\overline{WRH}$ is input $\overline{WRH}$ in 16-bit multiplexed mode.
MODE0 MODE1	44 30	I I	CMOS PD	These two pins with internal pull-down select one of the four parallel interfaces : MODE1 MODE0 0 0 8-bit multiplexed Intel 0 1 16-bit multiplexed Intel 1 0 8-bit multiplexed non-Intel 1 1 8-bit non-multiplexed NOTE: If upon reset, MODE1 = MODE0 = 0,$\overline{RD}$ = 0 and $\overline{WR}$ = 0, then the serial interface mode is entered.
ALE/AS	5	I	TTL	ALE used for Intel modes. AS used for non-Intel modes
$\overline{RD}$/E	6	I	TTL	$\overline{RD}$ used for Intel modes. E used for non-Intel modes.
$\overline{WR}$/ R/$\overline{W}$	7	I	TTL	$\overline{WR}$ used for Intel modes. R/$\overline{W}$ used for non-Intel modes.
READY/MISO	28	O	CMOS	READY is an output to synchronize accesses from the host microcontroller to the 29C462 in modes 0 & 1. This pin is an open drain. MISO is the serial data output for the serial interface.
$\overline{DSACK0}$	9	O/N	CMOS	$\overline{DASCK0}$ is an open drain output to synchronize accesses from the host microcontroller to the 29C462 in mode 3.

Address / Data Bus Function

	8-bit Intel multiplexed	8-bit Non-Intel multiplexed	16-bit Intel multiplexed	8-bit non-multiplexed	Serial interface
AD0	AD0	AD0	AD0	A0	CPOL
AD1	AD1	AD1	AD1	A1	CPHA
AD2	AD2	AD2	AD2	A2	CSAS
AD3	AD3	AD3	AD3	A3	STE
AD4	AD4	AD4	AD4	A4	MOSI
AD5	AD5	AD5	AD5	A5	unused
AD6	AD6	AD6	AD6	A6	SCK
AD7	AD7	AD7	AD7	A7	unused

	8-bit Intel multiplexed	8-bit Non-Intel multiplexed	16-bit Intel multiplexed	8-bit non-multiplexed	Serial interface
AD8	Port 1.0	Port 1.0	AD8	D0	Port 1.0
AD9	Port 1.1	Port 1.1	AD9	D1	Port 1.1
AD10	Port 1.2	Port 1.2	AD10	D2	Port 1.2
AD11	Port 1.3	Port 1.3	AD11	D3	Port 1.3
AD12	Port 1.4	Port 1.4	AD12	D4	Port 1.4
AD13	Port 1.5	Port 1.5	AD13	D5	Port 1.5
AD14	Port 1.6	Port 1.6	AD14	D6	Port 1.6
AD15	Port 1.7	Port 1.7	AD15	D7	Port 1.7

3 — Line Interface

There are three line inputs and one line output available on the 29C462. Which of the three inputs to use is either programmable by software or automatically selected by a diagnosis system.

The diagnosis system continuously monitors the data received through the three inputs, and compares it with each other and the selected bitrate. It then chooses the most reliable input according to the results.

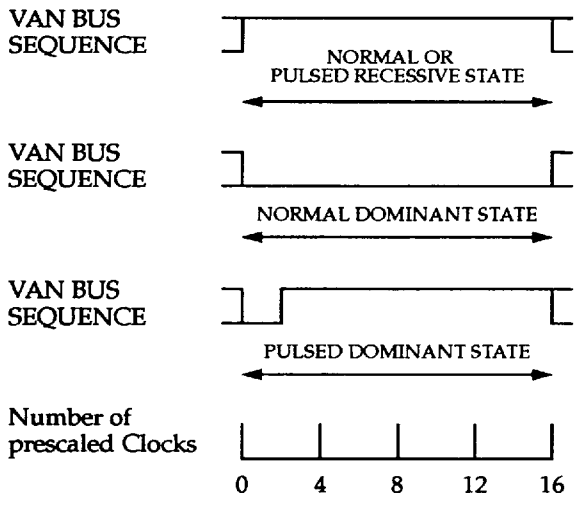
The data on the line is encoded according to the VAN specification ISO/TC22/SC3/WG1 revision 4,000. This means that the 29C462 is using a two level signal having a recessive (1) and a dominant (0) state. Furthermore, due to the simple medium used, all data transmitted on the bus is also received simultaneously.

The VAN protocol is hence a CSMA/CD (Carrier Sense Multiple Access/Collision Detection) protocol, allowing for continuous bitwise arbitration of the bus, and non-destructive (for the higher priority message) collision detection.

In addition to the VAN specification there is also a pulsed coding of the dominant and recessive states. This mode is intented to be used with an optical or radio link. In this mode the dominant state for the transmitter is a low pulse, and the recessive state is just a high level. When receiving in this mode it is not the state of the signal itself which is decoded but the edges. Also reception is imposed on the RXD0 input, and the diagnosis does not operate correctly.

In addition in this mode there is an internal loopback in the circuit since optical transceivers are not able to receive the signal that they transmit themselves .

Figure 1. State Encoding



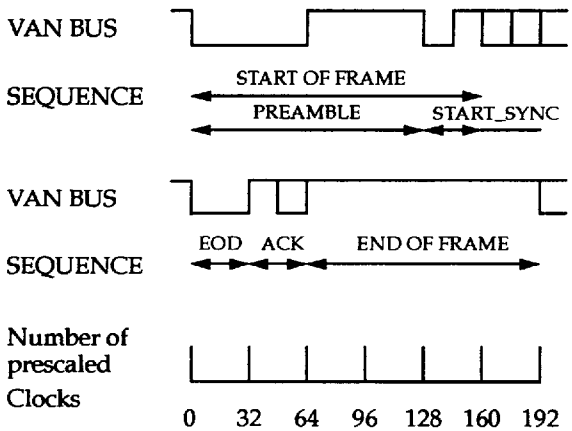
In figure 1 the pulsed waveforms are shown. In figures 3 through 5 the low "timeslots" (i.e blocks of 16 prescaled clocks) should be replaced by the dominant waveform showed in figure 1, if the correct representations for pulsed coding are to be seen.

The VAN bus supports three different module (unit) types:
- First, the autonomous module which is a bus master. It can emit Start Of Frame (SOF), initiate data transfers and receive messages,
- Second the synchronous access module, cannot emit any SOF sequences, but it can initiate data transfers and receive messages,
- Finally the slave module, which can only transmit using an in-frame response mechanism, and receive messages.

Figure 2. VAN bus frame

Figure 2 shows a normal VAN bus frame. It is initiated with a Start Of Frame (SOF) sequence shown in figure 3. The SOF can only be transmitted by an autonomous module. During the preamble the 29C462 will synchronize its bit rate clock to the data received.

Figure 3. Framing Sequences



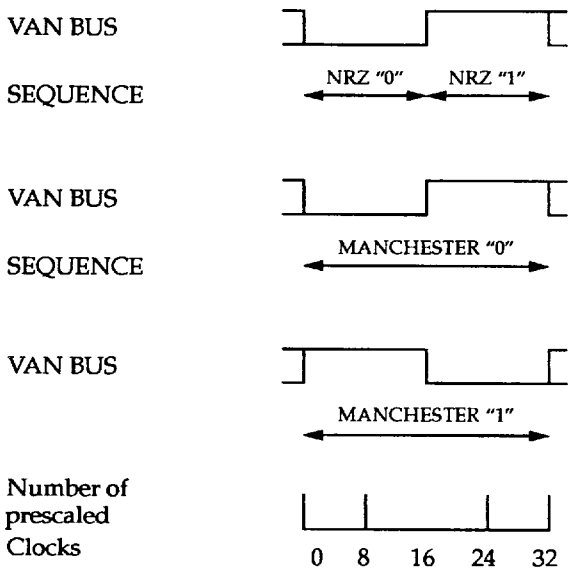
When the complete SOF sequence has been transmitted or received, the circuit will start the transmission or reception of the identifier field.

All data on the VAN bus, including the identifier and Frame Check Sum (FCS), are transmitted using enhanced Manchester code.

In enhanced Manchester code three NRZ bits are transmitted first followed by one Manchester bit, then three more NRZ bits followed by one Manchester bit and so on.

Since the high state is recessive and the low state is dominant, the bus arbitration can be done. If a module wants access to the bus, it must first listen to the bus during one full End Of Frame (EOF) and one full Inter-Frame Spacing (IFS) period, to determine whether the bus is free or not (i.e no dominant states received).

Figure 4. Data Encoding



Once the bus has been determined free, the module must now, if it is a synchronous access module, wait until it detects a preamble sequence.

Up till this point there can be several modules transmitting on the bus, and there is no possibility of knowing if this is the case or not. Therefore the first field in which arbitration can be performed is the identifier field. Since the logical zeroes on the bus are dominant, and all data is transmitted with the most significant bit first, the first module to transmit a logical zero on the bus will be prioritized module, i.e the message that is tagged with the lowest identifier will have priority over the other messages.

It is, however, conceivable that two messages transmitted on the bus will have the same identifier. The 29C462 therefore continues the arbitration of the bus throughout the whole frame. More, if the identifier in transmission has been programmed for reception as well, it transmits and receives messages simultaneously, right up till the FCS. Only then, if the 29C462 has transmitted the whole message, does it discard the message received.

Arbitration loss in the FCS field is considered as a CRC error during transmission.

This feature is called full data field arbitration, and it enables the user to extend the identifier. For instance it can be used to transmit the emitting modules address in the first bytes of the data field, thus enabling the identifier to specify the contents of the frame and the data field to specify the source of information.

The first field of the command is the extension bit (EXT). This bit is defined by the user on transmission and is received and retained by the 29C462. To conform with the revision 4.000 it should be set to 1 (recessive) by the user.

The second bit is the request acknowledge bit (RAK). If this bit is a logical one, the receiving module must acknowledge the transfer with an in-frame acknowledgement in the ACK field. If it is set to logical zero, then the ACK field must contain an acknowledge absent sequence.

Third we have the read/write bit (RNW). This bit indicates the direction of the data in a frame.
- If set to zero it is a "write" message, i.e data transmitted by one module to be received by another module.
- If set to one it implies a "read" message, i.e a request that another module should transmit data to be received by the one that requested the data (reply request message).

Last in the command field is the remote transmission request bit (RTR). This bit is a logical zero if the frame contains data and a logical one if the frame does not contain data.

All the bits in the command field are automatically handled by the 29C462, so the user need not to be concerned for the encoding and decoding of these. The command bits transmitted on the VAN bus are calculated from the current status of the active message.

After the command field comes the data field. This is just a sequence of bytes transmitted MSB first. In the revision 4.000 VAN recommendation the maximum message length is set to 28 bytes, but the 29C462 handles messages up to 30 bytes.

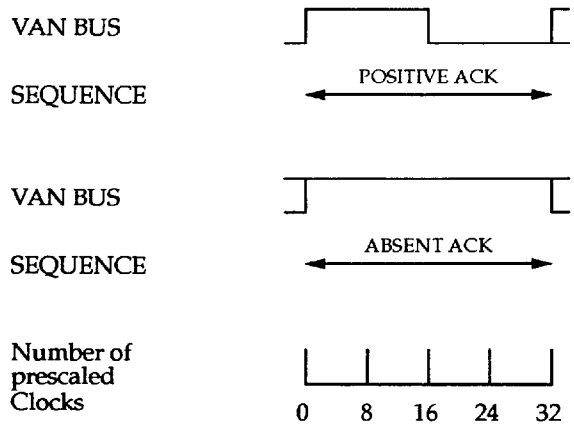
The next field is the FCS field. This field is a 15 bit CRC checksum defined by the following generator polynomial g(x) of order 15:

$g(x)= x^{15}+ x^{11}+ x^{10}+ x^9+ x^8+ x^7+ x^4+ x^3+ x^2 +1$

The division is done with a rest initialized to 0x7FFF, and an inversion of the CRC bits is performed before transmission.

However, since the CRC is calculated automatically from the identifier, command and data fields by the 29C462, it need not concern the user of the circuit. When the frame check sequence has been transmitted, the transmitting module must transmit an End Of Data (EOD) sequence, followed by the acknowledge field (ACK) and the End Of Frame (EOF) sequence to terminate the transfer.

Figure 5. Acknowledge Sequences



4 — Diagnosis System

The diagnosis system is based on the assumption that three seperate line receivers are connected to the VAN bus.

One of the receivers is connected in differential mode, sensing both the DATA and the $\overline{\text{DATA}}$ signals and is connected to the RXD0 input. The other two receivers are operating in single wire mode and are sensing only one of the two VAN bus signals. The receiver sensing DATA is connected to RXD1 and the one sensing $\overline{\text{DATA}}$ to RXD2.

The purpose is to detect if there are any short or open circuits on either the DATA or $\overline{\text{DATA}}$ lines.
If the diagnosis system finds a failure on either of the VAN bus signals it changes from nominal to degraded mode, and connects the line receiver not connected to the failing signal to the reception logic.
When the diagnosis system finds that the failing signal is working again, it returns to nominal mode and connects the differential line receiver to the reception logic.
If the user wishes to disable the diagnosis system, he only needs to select one of the fixed modes as listed in Table 1, and disregard the status bits in the Line status register.

The system performs five different types of analysis on the VAN bus signals:
- Input comparison,
- Asynchronous diagnosis,
- Synchronous diagnosis,
- Transmission diagnosis,
- Protocol anomaly.

In order to perform these analysis three signals are generated internally in the 29C462:
- The Return to Idle (RI), returns the diagnosis system to the nominal mode immediately.
- The Synchronous Diagnosis Clock (SDC), controls the cycle time of the synchronous diagnosis.
NOTE : Typically, there should be at least one message on the bus per SDC period.
- The Transmission In Progress (TIP), tells the diagnosis system to enable transmission diagnosis.

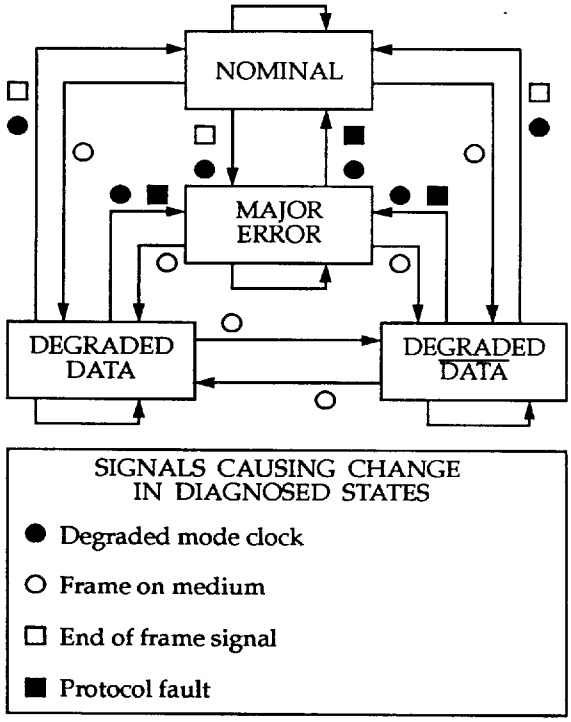
4.1 — Input Comparison

The input comparison is done once every "timeslot" (one timeslot is about 16 prescaled clock periods. The about is due to the synchronization of the timeslots to the data received).

If one of the three inputs differs from the others in this comparison, the high order bit of the status code is set (S2).

The only way of resetting this status bit are through the RI signal or a general reset. Every time RI goes high this status bit is reset.

Figure 6. Diagnosis System Status



4.2 — Asynchronous Diagnosis

The asynchronous diagnosis is done by comparing the number of edges on the DATA and the $\overline{DATA}$ inputs (RXD1 and RXD2).

If four edges are detected on one input and no edges on the other during the same timeperiod, the second input is considered faulty, and the status will change to one of the degraded modes.

4.3 — Synchronous Diagnosis

The synchronous diagnosis is counting the number of edges on the data input connected to the reception logic during one SDC period.

If there are less than four edges during one SDC period the status will change to the major error mode.

4.4 — Transmission Diagnosis

The transmission diagnosis compares the RXD1 and RXD2 inputs with the data transmitted on the TXD output.

If the TXD output becomes recessive during transmission and only one of the inputs are dominant, the recessive input is considered faulty (due to an open circuit) and the status is changed to reflect that.

4.5 — Protocol Anomaly

The protocol anomaly is detected by counting the number of consecutive dominant "timeslots".

If eight consecutive "timeslots" are dominant the status will change into the major error mode.

4.6 — Returning to Nominal Mode

There are two ways of returning to the nominal mode. The first is if the diagnosis system detects no errors during two consecutive SDC periods. This functionality allows the system to return to nominal mode if some error is due to, for instance, a bad connector.

The second way of returning to nominal mode is through the RI signal (or a general reset).

4.7 — Automatically Selected Input

If the mode of the diagnosis system is set to automatic selection, the input connected to the reception logic is decided by the operation mode.

Signal connected to reception logic is RXD0 (differential) in nominal mode
RXD1 in degraded DATA mode
RXD2 in degraded $\overline{DATA}$ mode.

In the major error mode the RXD1 and RXD2 inputs are connected alternatively to the reception logic. The input connected is changed every SDC period until a different diagnosis mode is entered.

Table 1 : Status and Mode Codes

M1	M0		S1	S0	
0	0	Com. on RXD0	0	0	Nominal
1	0	Com. on RXD1	0	1	Degraded DATA
0	1	Com. on RXD2	1	0	Degraded $\overline{DATA}$
1	1	Auto. select	1	1	Major error

NOTE: Automatic selection is only recommended if SDC is active,. RI is disabled in automatic mode and enabled in the fixed modes.

4.8 — RI, SDC and TIP GENERATION

The RI signal can be generated at the end of each frame. The automatic RI generator allows the user to make a diagnosis per frame, and is always enabled in the fixed modes, but always disabled in the automatic mode.

The SDC can be generated by a divider chain connected to the "timeslot" clock and controlled by the SDC divider if the Enable SDC bit is set (ESDC). The general idea is to generate one SDC period per frame on the VAN bus. One SDC period can also be generated through the manual SDC command.

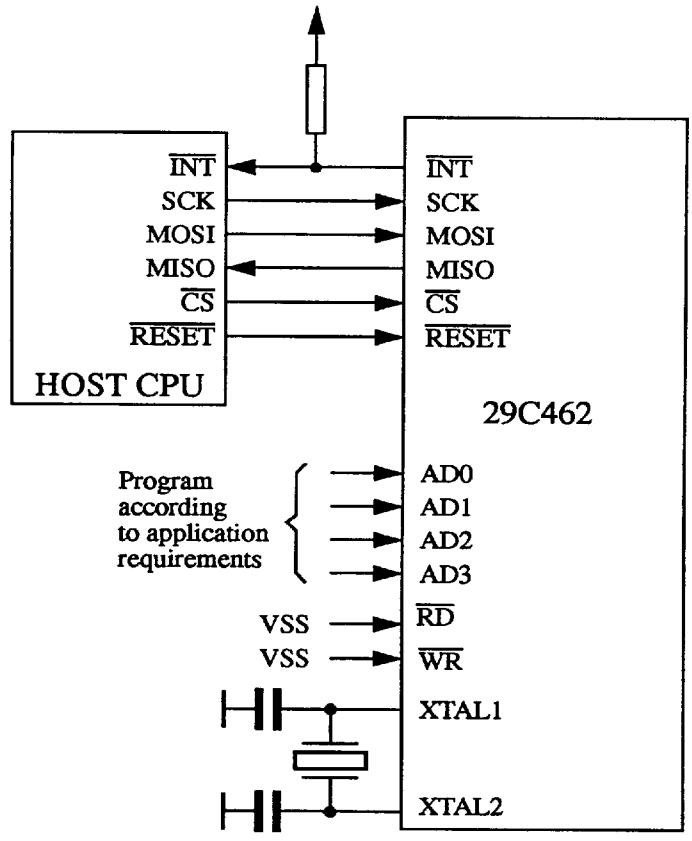
The TIP signal is generated by the 29C462 if the Enable TIP bit is set (ETIP), thus enabling transmission diagnosis.

5 — Serial Peripheral Interface

The serial peripheral interface (SPI) is fully compatible to the SPI protocol of Motorola.The interface will be implemented for the slave-mode only.

The serial peripheral interface allows an interconnection of several CPUs and peripheral devices on the same printed circuit board.

In a SPI, separate wires are required for data and clock, so the clock is not included in the data stream as shown below.



MOSI: Master Out Slave In

The MOSI Pin is the data output of the master (CPU) device and the data input of the slave device. So data is transferred serially from the master to the slave on this line; most significant bit first, least significant bit last.

MISO: Master In Slave Out

The MISO pin is configured as the output of a slave device and as input of the master device (CPU).

$\overline{CS}$: Chip Select (used as slave select for the SPI interface)

An asserted state on the slave select input enables the 29C462 to accept data on the MOSI pin. The $\overline{CS}$ must not toggle between each transmitted byte. The 29C462 will only drive data to the serial data register if this pin is asserted.

SCK: Serial Clock

The master device provides the serial clock for the slave device. Data is transferred synchronous to this clock in both directions. The master and the slave devices exchange a data byte during a sequence of eight clock pulses.

5.1 — Serial Interface Protocol

The general format of the data exchange between the 29C462 and the host is a bit-for-bit exchange on each SCK clock pulse. Data is arranged in the 29C462 such that the significance of a bit is determined by its position from the start for output and from the end for input, most significant bit is sent first. The order is such that bit exchanges in multiples of 8 bit up to 15 bytes of data are allowed. A maximum of 17 bytes can be send to the 29C462 including 1 address byte, 1 SPI control byte and 15 data bytes.

At the beginning of a transmission over the serial interface, the first byte will be the address of the 29C462 register to be accessed. The next byte transmitted is a Control byte, which contains the number of bytes to be transmitted (consecutive bytes- Address, Address + 1,..., Address +(n), n=0 to 14) and whether this is to be a Read or Write access to the 29C462.

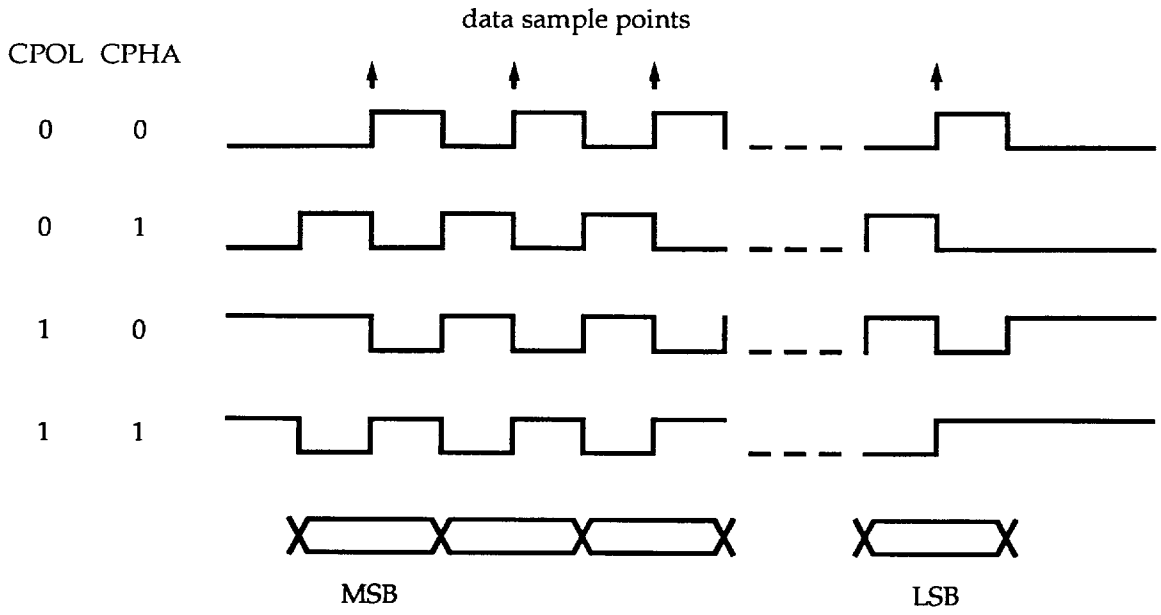
To ensure that the 29C462 device is not out of synchronisation, the 29C462 will transmit through the MISO pin during Address and Control byte times data "0xAA" and "0x55". This can be enabled and disabled depending on the state of the AD3 pin. This way the master will always know if the 29C462 is not synchronised. If the 29C462 is out of sync, the master SPI can re-sync by transmitting a string of 16 "0xFF" bytes. When the SPI receives an address of "0xFF", it will assume that the next byte is also an address. An asserted edge on $\overline{CS}$ also resynchronizes the SPI.

When the Slave Select pin is in the inactive state, the SCK is allowed to toggle. This will have no effect on the serial transmission, because the device is not selected.

The states of the pins AD0, AD1, AD2, AD3 are sampled on the rising edge of $\overline{RESET}$. They have the following functions:

Table 2 : AD[3:0] function in SPI mode

Pin	Function	Comment
AD0	CPOL	Idle Clock Polarity CPOL=0 SCK is idle low, CPOL=1 SCK is idle high
AD1	CPHA	Clock Phase. CPHA=0 data is sampled on the rising edge on SCK (CPOL=0), or on the falling of SCK (CPOL=1) CPHA=1 data is sampled on the falling edge of SCK (CPOL=0), or on the rising of SCK (CPOL=1)
AD2	CSAS	Chip Select Active State CSAS=0 : Asserted state if $\overline{CS}$ is logic low. CSAS=1 : Asserted state if $\overline{CS}$ is logic high.
AD3	STE	Enables the transmission of the synchronisation bytes, while the address and control bytes are transferred. STE=0 : The first two bytes which will be send to the CPU after $\overline{CS}$ is asserted are "0x00" and "0x00" STE=1 : The first two bytes which will be send to the CPU after $\overline{CS}$ is asserted are "0xAA"and "0x55".



5.2 — Serial Control Byte

The Serial Control Byte is transmitted by the CPU to the 29C462 as follows:

7	6	5	4	3	2	1	0
DIR	0	0	0	Serial Data Length			

DIR : Serial transmission direction.

zero : The data bytes or 29C462 configuration registers will be read, so the 29C462 will transfer informations to the CPU.

one : The 29C462 configuration registers or the data bytes will be sent from the CPU to the 29C462.

SDL : Serial Data Length

The first data byte (third byte of the SPI protocol) will be written to or read from the 29C462 address (first byte of the SPI protocol). After this, the address is incremented by the SPI logic and the next data is written or read from this address. In one data stream, a maximum of 15 data bytes can be transferred. A SDL of zero is not allowed. After a SDL of zero is received, the SPI must be resynchronized.

The serial interface is configured from the states of AD[3:0] on the rising edge of $\overline{\text{RESET}}$.

When the CPU conducts a read, the CPU sends an address byte and a serial control byte.
When the 29C462 responds back with data, the 29C462 ignores the MOSI pin (transmission from the CPU). The CPU may transmit an address and serial control byte after $\overline{\text{CS}}$ is de-actived and then re-actived. This means the chip select should be actived and de-actived for each read or write transmission.

Synchronisation bytes must be monitored carefully. For example, if the 29C462 does not transmit the "0xAA" and "0x55" synchronization bytes correctly, then the previous transmission may be incorrect too.

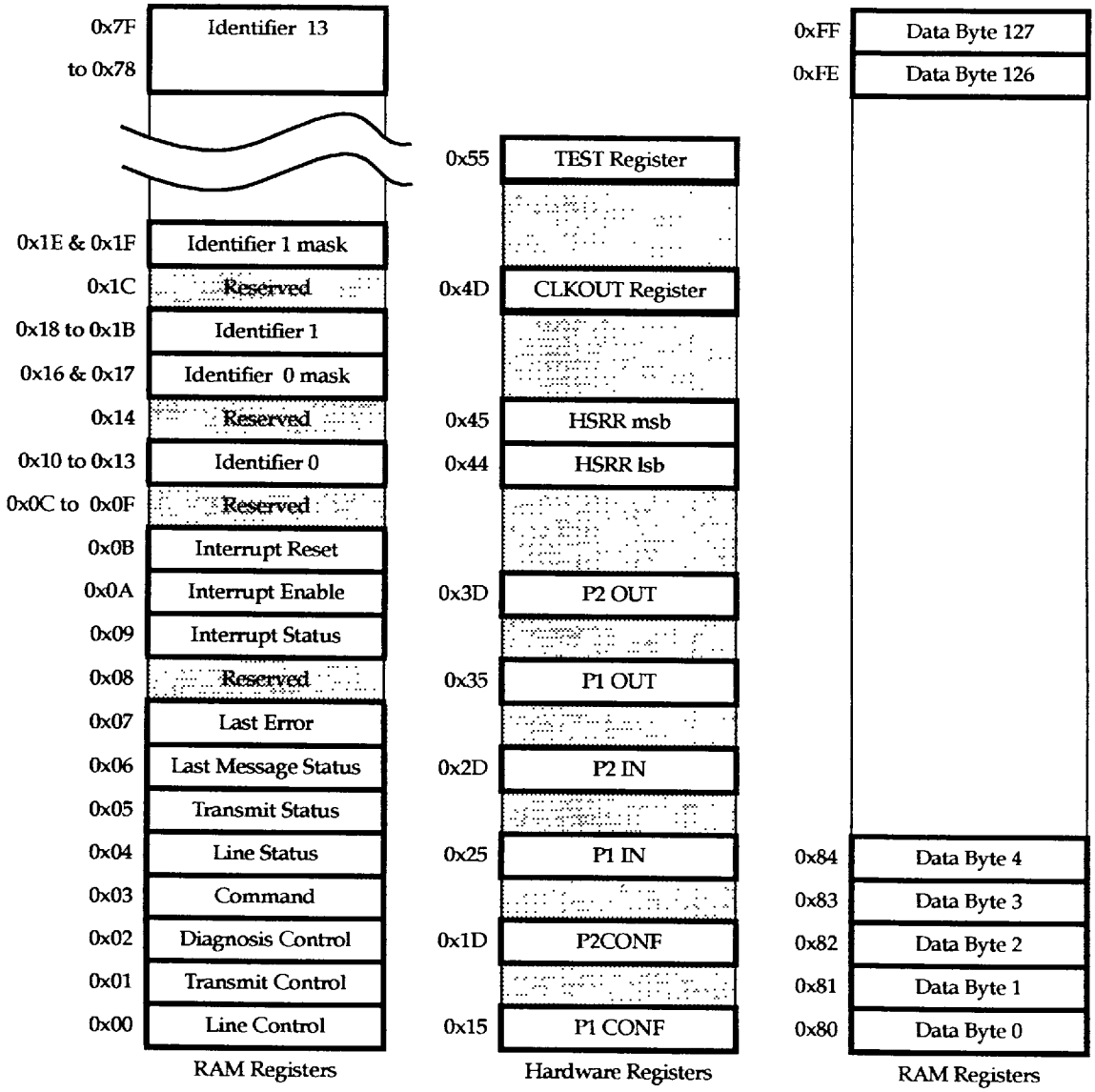
The MISO pin is tri-stated if $\overline{\text{CS}}$ is inactive.

6 — Registers

The 29C462 address memory map is shown below

6.1 — Memory Map

Table 3 : Memory Map



NOTE: All the non specified address between 0x00 and 0x7F are considered as Reserved

6.2 — Line Control Register (0x00)

7	6	5	4	3	2	1	0
CD3	CD2	CD1	CD0	PC	0	IVTX	IVRX

This register is a read/write register.
default at reset : 0x00

reserved : Bit 2, this bit must not be set by the user; a zero must always be written to this bit.

CD[3:0] : Clock Divider , they control the VAN bus rate through a Baud rate generator according to the formula below:

$$f(TSCLK) = \frac{f(XTAL1)}{n \times 16}$$

XTAL1 = 8 MHz (for example)

Table 4 : Baud Rate Divider

CLOCK DIVIDER CD[3:0]	Divide by	KTime-Slot/s	KBits/s
0000	1	500	400
0001	2	250	200
0010	4	125	100
0011	8	62.5	50
0100	16	31.25	25
0101	32	15.625	12.5
0110	64	7.813	6.25
0111	128	3.906	3.125
1000	1.5	333.333	266.666
1001	3	166.666	133.333
1010	6	83.333	66.666
1011	12	41.666	33.333
1100	24	20.833	16.666
1101	48	10.416	8.333
1110	96	5.208	4.166
1111	192	2.604	2.083

PC: Pulsed Code

One :The 29C462 will transmit and receive data using the pulsed coding mode (i.e optical or radio link mode). The use of this mode implies communication via the RXD0 input and the non-functionality of the diagnosis system.
Zero : (default at reset) The 29C462 will transmit and receive data using the Enhanced Manchester code. (RXD0, RXD1, RXD2 used).

IVTX: Invert TXD output

IVRX: Invert RXD inputs

The user can invert the logical levels used on either the TXD output or the RXD inputs in order to adept to different line drivers and receivers.

One : A one on either of these bits will invert the respective signals.
Zero: (default at reset) The 29C462 will set TXD to recessive state in Idle mode and consider the bus free (recessive states on RXD inputs).

6.3 — Transmit Control Register (0x01)

7	6	5	4	3	2	1	0
MR3	MR2	MR1	MR0	0	0	0	MT

This register is a read/write register.
default at reset : 0x00

reserved : Bit 1, 2, 3 these bit must not be set by the user; a zero must always be written to these bit.

MR[3:0] : Maximum Retries, these bits allow the user to control the amount of retries the circuit will perform if any errors occurred during transmission.

Table 5 : Retries

MR[3:0]	Nb of retry	Nb of transmis
0000	0	1
0001	1	2
0010	2	3
0011	3	4
0100	4	5
0101	5	6
0110	6	7
0111	7	8
1000	8	9
1001	9	10
1010	10	11
1011	11	12
1100	12	13
1101	13	14
1110	14	15
1111	15	16

Note : Bus contention is not regarded as an error and that an infinite number of transmission attempts will be performed if bus contention occurs continuously .

MT: Module type

The VAN bus supports three different module types:
- The autonomous module which is a bus master and can transmit Start Of Frame (SOF) sequences, initiate data transfers and receive messages,

- The synchronous access module, cannot transmit SOF sequences, but initiate data transfers and receive messages,

- The slave module, which can only transmit using an in-frame mechanism, and receive messages.

One : The 29C462 will generate SOF and is an autonomous module.
Zero : The 29C462 will not generate SOF sequences and is hence a synchronous access or a slave module.

6.4 — Diagnosis Control Register (0x02)

7	6	5	4	3	2	1	0
SDC3	SDC2	SDC1	SDC0	M1	M0	ETIP	ESDC

This regsiter is a read/write register
default at reset : 0x00.

The diagnosis is discussed in greater detail in paragraph 4 of this chapter.

In its four high order bits the user can program the SDC rate SDC[3:0].

In its two medium order bits the diagnosis system mode is controlled : M1, M0.

In the two low order bits, the user controls if the SDC and TIP are to be generated automatically ETIP, ESDC.

SDC[3:0] : SDC divider

Table 6 : System Diagnosis Clock Divider

SDC DIVIDER SDC[3:0]	Divide by	SDC DIVIDER SDC[3:0]	Divide by
0000	64	1000	16384
0001	128	1001	32768
0010	256	1010	65536
0011	512	1011	131072
0100	1024	1100	262144
0101	2048	1101	524288
0110	4096	1110	1048576
0111	8192	1111	2097152

M1, M0 : Operating mode command bits

Table 7 : Diganosis System Command Bits

M1	M0	
0	0	Communication on RXD0
1	0	Communication on RXD1
0	1	Communication on RXD2
1	1	Automatic selection

ETIP: Enable Transmission In Progress

One : Enable TIP genration,
Zero : Disable TIP generation.

The Transmission In Progress (TIP), tells the diagnosis system to enable transmission diagnosis.

ESDC : Enable System Diagnosis Clock

One : Enable SDC divider.
Zero : Disable SDC divider.

The Synchronous Diagnosis Clock (SDC), controls the cycle time of the synchronous diagnosis.

6.5 — Command Register (0x03)

7	6	5	4	3	2	1	0
GRES	SLEEP	IDLE	ACTI	REAR	0	0	MSDC

This register is a Write only register.

reserved : Bit 1, 2 these bit must not be set by the user; a zero must always be written to these bit.

If the circuit is operating at low bitrates there might be a considerable delay between the writing of this register and the performing of the actual command (worst case 6 timeslots). The user is therefore recommanded to verify, by reading the Line Status Register that the commands have been performed.

GRES : General Reset. The Reset circuit command bit performs, if set, exactly as if the external reset pin was asserted.

One : Reset active
Zero : Reset inactive

SLEEP : Sleep command. If the user sets the Sleep bit, the circuit will enter sleep mode. When the circuit is in sleep mode, all non-user registers are setup to minimize power consumption and the oscillator is stopped. To exit from this mode the user must set either the idle or activate commands.

One : Sleep active
Zero : Sleep inactive

IDLE : Idle command. If the user sets the Idle bit, the circuit will enter idle mode. In idle mode the oscillator will operate, but the 29C462 will not transmit or receive anything on the bus, and the TXD output will be in three state

One : Idle active
Zero : Idle inactive

ACTI : Activate command. The Activate command will put the circuit in the active mode, i.e it will transmit and receive normally on the bus. When the circuit is in activate mode the TXD three-state output is enabled.

One : Activate active
Zero : Activate inactive

REAR : Re-Arbitrate command. This command will, after the current attempt, reset the retry counter and re-arbitrate the messages to be transmitted in order to find the highest priority message to transmit.

One : Re-arbitrate active
Zero : Re-arbitrate inactive

MSDC : Manual System Diagnosis Clock.Rather than using the SDC divider described in paragraph 5.4, the user can use the manual SDC command to generate a SDC pulse for the diagnosis system.

This MSDC pulse should be high at least two timeslot clock.

6.6 — Line Status Register (0x04)

7	6	5	4	3	2	1	0
X	SPG	IDG	S2	S1	S0	TXG	RXG

This register is a Read only register.
default at reset : 0bX01XXX00.

This register reports the operation mode of the 29C462 in the Sleep an Idle bits (Command Register located at

address 0x03) as well as the diagnosis system status bits S2 to S0 discussed in paragraph 4.

SPG : Sleeping

IDG : Idling. Default mode at reset

S2 to S0 : Diagnosis system status bits

S2 : As soon as one of the three inputs (RXD2, RXD1, RXD0) differs from the others in the input comparison analysis performs by the diagnosis system, S2 is set.

The only ways to reset this status bit are through the RI signal or a general reset.

Table 8 : Diagnosis System Status Bits

S1	S0	COMMUNICATION INDICATION
0	0	Nominal mode, differential communi.
0	1	Degraded over DATA, fault on $\overline{DATA}$
1	0	Degraded over $\overline{DATA}$, fault on DATA
1	1	Major error, fault on DATA and $\overline{DATA}$

TXG : Transmitting. If this status bit is active, it indicates that the 29C462 has chosen an identifier to transmit, and it will continue to make transmission attempt for this message until it succeeds or the retry count is exceeded.

RXG : Receiving. The receiving indicates that there is activity on the bus.

NOTE : For safe modification of identifier registers both bits should be inactive.

6.7 — Transmission Status Register (0x05)

7	6	5	4	3	2	1	0
NRT3	NRT2	NRT1	NRT0	IDT3	IDT2	IDT1	IDT0

This register is a Read only register.
default at reset : 0x00.

The transmission Status register contains the number of retries made up-to-date, according to the table 4 in paragraph 6.3, and the identifier currently in transmission.

NRT[3:0] : Number of retries done in transmission.

IDT[3:0] : Identifier currently in transmission.

6.8 — Last Message Status Register (0x06)

7	6	5	4	3	2	1	0
NRT3	NRT2	NRT1	NRT0	IDTR3	IDTR2	IDTR1	IDTR0

This register is a Read only register.
default at reset : 0x00.

This register is basically the same as the transmission status register. It contains the last identifier number that was successfully transmitted, received or exceeded its retry count.

If it was a successful transmission, the number of retries performed can be seen in this register as well.

NRTR[3:0] : Number of retries done successfully in transmission.

IDTR[3:0] : Identifier number that was successfully transmitted, received or exceeded its retry count.

6.9 — Last Error Status Register (0x07)

7	6	5	4	3	2	1	0
X	BOC	BOV	TYM	FCSE	ACKE	CV	FV

This register is a Read only register.
default at reset : 0x00.

The Last Error Status Register contains the error code for the last transmission or reception attempt. It is updated after each attempt, i.e. several error codes can be reported during one single transmission (with several retries).

BOC : Buffer occupied. BOC indicates that the received bit of the identifier status was set and a reception attempt was made.

One : BOC active
Zero : BOC inactive

BOV : Buffer overflow. BOV indicates that the buffer length setup in the identifier status register was shorter than the number of bytes received plus 1, and thus, some data was lost.

One : BOV active
Zero : BOV inactive

TYM : Type mismatch. TYM means that an identifier received is specified in an identifier register as either a reply or a received message, but that the message received on the bus indicates the opposite type.

One : TYM active
Zero : TYM inactive

FCSE : Framing Check Sequence Error. FCSE indicates a mismatch between the FCS received and the FCS calculated

One : FCSE active
Zero : FCSE inactive

ACKE : Acknowledge Error. ACKE is set when a transmission attempt requested an acknowledge from the receiver but none was received, or vice-versa.

One : ACKE active
Zero : ACKE inactive

CV : Code Violation. CV means that a data bit on the bus was expected to be a Manchester bit, but it was not or than a transmitted dominant bit was received recessive (physical violation due to the medium) or bad SOF sequence.

One : CV active
Zero : CV inactive

FV : Frame Violation. FV indicates a bad framing sequence on the VAN bus.

One : FV active
Zero : FV inactive

6.10 — Interrupt Status Register (0x09)

7	6	5	4	3	2	1	0
REI	X	X	ER	TE	RE	TOK	ROK

This register is a Read only register.
default at reset : 0x80

REI : Reset interrupt. REI indicates that the circuit has detected a valid reset command via the RESET pin or the reset command bit.

NOTE: This interrupt cannot be disabled, since its enable bit is set when a reset is detected.

One : REI active.
Zero : REI inactive.

ER : Exceeded retry. ER is set when a transmission has been retried too many times.

One : IT active.
Zero : IT inactive.

The last four interrupts indicate successful and unsuccessful transmissions and receptions.
active.

TE : Transmit error interrupt.

RE : Receive error interrupt.

TOK : Transmit OK interrupt.

ROK : Receive OK interrupt.

One : TE, RE, TOK, ROK actives
Zero : TE, RE, TOK, ROK inactives

6.11 — Interrupt Enable Register (0x0A)

7	6	5	4	3	2	1	0
1	0	0	ERE	TEE	REE	TOKE	ROKE

This register is a read/write register.
default at reset : 0x80

NOTE: On reset the Reset interrupt enable bit is set to 1 instead of 0, as is the general rule.

ERE : Exceeded retry count enable.

TEE : Transmit error enable

REE : Reception error enable.

TOKE : Transmission OK enable.

ROKE : Reception OK enable.

One : IT enabled.
Zero : IT disabled.

6.12 — Interrupt Reset Register (0x0B)

7	6	5	4	3	2	1	0
RER	0	0	ERS	TES	RES	TOKS	ROKS

This register is a write only register

ERS : Exceeded retry count reset.

TES : Transmit error reset.

RES : Reception error reset.

TOKS : Transmission OK reset.

ROKS : Reception OK reset.

One : IT reset.
Zero : IT unchanged.

6.13 — Identifier Registers

There is a total of 14 identifier registers, each occupying 8 bytes for addressing simplicity, integrated into the circuit. Each register contains two 16-bit registers for the indentifier, mask and command fields plus two 8-bit registers for DMA pointers and message status.

The base_address of each identifier register is (0x10 + (0x08 * identifier_number)).

When the 29C462 is reset either via the external reset pin or the general reset command, the identifier registers are not affected. That is, on power-up of the circuit, all the identifier registers start with random values.

Due to this fact, the user should take care to initialize all the identifier registers before exiting from idle mode. The easiest way to disable an identifier register is to set the received and transmitted bits to 1 in the Length of Status Register.

6.13.1 — Identifier and Command Register

15 to 4	3	2	1	0
12 bits identification tag	EXT	RAK	RNW	RTR

This register is a Read/Write register.

The identifier and command word register is a 16-bit register located at the base_address. It allows the user to specify the full 12-bit identifier field of the ISO recommendation and the 4-bit command.

Note, that no identifier comparison will be done on the command bits, but that the EXT bit will be written into the register upon a reception hit.

Also note, that the RNW and RTR bits, as well as the status bits in the length and status register, must be in a valid position for reception or transmission. If not, the message corresponding to this identifier is considered as unactive or invalid.

Upon a reception hit (i.e. a good comparison between the identifier received and an identifier specified, taking the comparison mask into account, as well as a status and command indicating a message to be received), the 13 high bits of this register will be rewritten with the identifier and EXT bits actually received. There is no way of knowing if an acknowledge sequence was requested or not.

6.13.2 — Message Start Address Register

7	6	5	4	3	2	1	0
0	7 bits Buffer Start Address						

The message start address register at address (base_address + 0x02) is 8 bits wide. It indicates where in the RAM area the message buffer is located.

Since the RAM area base address is 0x80, the value in this register is the offset from that address. If the message buffer length value is illegal (i.e. zero), this register is redefined as beeing a link pointer, thus containing the identifier number of the identifier that contains the actual buffer address, length and received status. However, the identifier, mask, error and transmitted status used will be that of the originally matched identifier. In any case, if a link is intented, the four high bits of this register should be set to 0.

This allows several identifier registers to use the same actual reception buffer in RAM, thus diminishing the memory usage.

Note that only 1 level of link is supported.

6.13.3 — Message Length & Status Register

7	6	5	4	3	2	1	0
5 bits Buffer Length					IDE	IDT	IDR

The message buffer length and status register at address (base_address + 0x03) is also 8 bits wide.

The 5 high bits of this register allows the user to specify either the length of the message to be transmitted, or the maximum length of a message receivable in the pointed reception buffer.

Note, that the first byte in the message buffer does not contain data, but the length of the message received. This implies that the length value has to be equal to or greater than the maximum length of a message to be received in this buffer (or the length of a message to be tansmitted) plus 1, thus allowing a maximum length of 30 bytes.

If the value of this field is "illegal" (i.e 0x00) then this message pointer is defined as being a link (see Message start address register).

The three low order bits of this register contains the message status, and together with the RNW and RTR bits of the command, they define the message type of this identifier. The status bits are only set by the 29C462, so it's the user that must reset them. The received and

transmitted bits are only set if the corresponding frame is without errors or the retry count has been exceeded.

IDE : Identifier error

IDT : Identifier transmitted

IDR : Identifier received

6.13.4 — Mask Register

15 to 4	3	2	1	0
12 bits identification mask	0	0	0	0

The 16-bit Mask register at address (base_address + 0x06) allows bitwise masking of the comparison between the identifier received and the identifier specified.

A value of 1 indicates comparison enabled, while 0 indicates comparison disabled.

6.14 — PORT REGISTER

P1CONF (0x15)

7	6	5	4	3	2	1	0
P1CONF[7:0]							

Port 1 Input/Output configuration bits

One : Port pin configured as an output
Zero : Port pin configured as an input

The default value of the P1CONF register after a reset is 0x00.

P2CONF (0x1D)

7	6	5	4	3	2	1	0
P2CONF[7:0]							

Port 2 Input/Output configuration bits

One : Port pin configured as an output
Zero : Port pin configured as an input

The default value of the P2CONF register after a reset is 0x00.

P1IN (0x25)

7	6	5	4	3	2	1	0
P1IN[7:0]							

Port 1 Data In

One : A one (high voltage) is read from the pin.
Zero : A zero (low voltage) is read from the pin.

P2IN (0x2D)

7	6	5	4	3	2	1	0
P2IN[7:0]							

Port 2 Data In

One : A one (high voltage) is read from the pin.
Zero : A zero (low voltage) is read from the pin.

P1OUT (0x35)

7	6	5	4	3	2	1	0
P1OUT[7:0]							

Port 1 Data Out

One : A one (high voltage) is written to the pin.
Zero : A zero (low voltage) is written to the pin.

The default value of the P1OUT register after a reset is 0x00.

P2OUT (0x3D)

7	6	5	4	3	2	1	0
P2OUT[7:0]							

Port 2 Data Out

One : A one (high voltage) is written to the pin.
Zero : A zero (low voltage) is written to the pin.

The default value of the P2OUT register after a reset is 0x00.

6.15 — High Speed Registers (0x44 & 0x45)

15 to 8 (0x45)	7 to 0 (0x44)
High Byte	Low Byte

The High Speed Read Register (HSRR) is a read only register and is the ouput buffer for the CPU Interface Logic.This register is part of the CPU Interface Logic and is not located in the RAM. During a read to the RAM (low speed registers) this register is loaded with the value of the low speed register being accessed.

The HSRR is available to provide a method to read the 29C462 when the CPU (host microcontroller) is unable to satisfy read cycle timings for low speed 29C462 regsiters. In other words, if the read access time of the 29C462 is too slow for the CPU and the CPU cannot extend the read bus cycle, the following method should be used.

The default value of the HSRR after a reset is unknown.

DOUBLE READ OPERATION

The CPU can execute double reads where the first read addresses the low speed register and the second read addresses the HSRR. The first read is a dummy read for the CPU, however the low speed register value is stored in the HSRR. The second read to the HSRR will produce the data from the desired low speed register.

Therefore, if the access time of a low speed register is too long for the CPU then a second read to the HSRR will produce the correct data. Please note low speed register and HSRR have different access timing specifications in the 29C462 data-sheet.

During a 16-bit read access the low and high byte will contain the 16-bit value from the read access. For an 8-bit read access, the low byte will contain the value from the last even address read access while the high byte will contain the value from the last odd address read access.

6.16 — CLKOUT Register (0x4D)

7	6	5	4	3	2	1	0
0	0	0	0	CDV3	CDV2	CDV1	CDV0

The CLKOUT register controls the frequency of the CLKOUT signal according to the table below.

Table 9 : Programming CLKOUT

CDV[3:0]	CLKOUT Frequency
0000	XTAL1
0001	XTAL1/2
0010	XTAL1/3
0011	XTAL1/4
0100	XTAL1/5
0101	XTAL1/6
0110	XTAL1/7
0111	XTAL1/8
1000	XTAL1/9
1001	XTAL1/10
1010	XTAL1/11
1011	XTAL1/12
1100	XTAL1/13
1101	XTAL1/14
1110	XTAL1/15
1111	Reserved

6.17 — TESTREG (0x55)

This register is only intented for factory use, and the functionality of this one is not specified in any way. Furthermore,it is subject to change without notice, the only exception beeing for incoming inspection tests using the MHS test program.

7 — Message types

There are 5 basic message types defined in the 29C462. Two of them (transmit and receive message types) corespond to the normal frame, and the rest correspond to the different versions of reply frames.

Transmit Message				
	RNW	RTR	Transmitted	Received
Initial setup	0	0	0	Dont care
After transmis-sion	0	0	1	Unchanged

To transmit a normal data frame on the VAN bus, the user must program an identifier as a Transmit Message. The 29C462 will then transmit this message on the bus until it has succeeded or the retry count is exceeded.

Receive Message				
	RNW	RTR	Transmitted	Received
Initial setup	0	1	Dont care	0
After reception	0	1	Unchanged	1

The opposite of the transmit message type is the Receive Message type. This message type will not generate any frames on the bus. Instead it will listen to the bus until a frame passes that matches its identifier, with the mask taken into account, and then receive the data in that frame.

The data received will be stored in the message buffer and the length of the message received is stored in the first byte of the message buffer.

The actual identifier received is stored in the identifier register itself. This identfier may differ from the identifier specified in the register due to the effect of the mask register.

Normally this should not interfere with the next identifier comparison since the bits that may differ are masked via the mask register.

Reply Request Message				
	RNW	RTR	Transmitted	Received
Initial setup	1	1	0	0
After transmis-sion (Waiting for reply)	1	1	1	0
After reception (of reply)	1	1	1	1

The Reply Request Message type is a demand to transmit on the VAN bus a reply request. When this message type is programmed, three things can happen.

In the first case no other modules on the bus responded with an in-frame reply, and in this case the 29C462 will set the message type to the after transmission state. When this message type is programmed, the 29C462 will listen on the bus for a deffered reply frame matching this identifier, without transmitting the reply request.

The second case is that another module on the bus replies with an in-frame reply.In this case the message type will pass immediatly into the after reception state, without passing the after transmission state.

Reply Request Message without transmission				
	RNW	RTR	Transmitted	Received
Initial setup	1	1	Dont care	0
After reception	1	1	Unchanged	1

In the third case the 29C462 has not yet started to transmit the reply request, when another module either requests a reply, andgets it, or transmits a deffered reply. Warning! This should avoided as it may result in an illegal message type (Illegal Reply Request).

Immediate Reply Message				
	RNW	RTR	Transmitted	Received
Initial setup	1	0	0	0
After transmis-sion	1	0	1	1

The Immediate Reply Message will attempt to transmit an in-frame reply, using the data in the message buffer.

Deffered Reply Message				
	RNW	RTR	Transmitted	Received
Initial setup	1	0	0	1
After reception (of reply request)	1	0	1	1

Above a Deffered Reply Message is shown. This message type will immediately transmit a deffered reply frame.

Reply Request Detection Message				
	RNW	RTR	Transmitted	Received
Initial setup	1	0	1	0
After reception	1	0	1	1

Finally there is the Reply Request Detector Message type. Its purpose is to receive a reply request frame and notify the processor, without transmitting an in-frame reply.

Inactive Message				
	RNW	RTR	Transmitted	Received
Recom-manded	Dont care	Dont care	1	1
After transmis-sion	0	0	1	Dont care
After reception	0	1	Dont care	1
Illegal reply request	1	1	0	1

The table above shows all inactive message types. The last combination will transmit a reply request, but will not receive the reply since its buffer is tagged as occupied.

8 — Priority

The priority handling on the VAN bus itself is already explained in the Line interface section. The priorities for the messages in the 29C462 is however slightly different.

For instance it's possible that an identifier matches two or more of the identifiers programmed into the registers. In this case, it is the lowest identifier number that has priority, i.e. if both identifier 5 and 10 match the identifier received, it is the identifier 5 that will receive the message.

However, since the identifier 5 will become an inactive message when it has received the frame, the next time the same identifier is seen on the bus, the corresponding data will be received by identifier 10.

The same is valid for messages to be transmitted, i.e. if two or more messages are ready to be transmitted, it is the one with the lowest identifier number that will get priority.

If for instance identifier 10 is beeing transmitted, and identifier 5 becomes ready for transmission, the 29C462 will continue to transmit identifier 10 until it succeeds or exceeds maximum amount of retries allowed.

Should it be urgent that the message in identifier 5 be transmitted, the microprocessor can give the rearbitrate command to the 29C462, and will abort the transmission of identifier 10 after the current attempt, and check what messages are available for transmission.

Since identifier 5 has a higher priority than identifier 10, the 29C462 will start to transmit the former until it is successful or exceeds the retry count, and then continue with identifier 10. The retry count for identifier 10 will be lost though, and is reset to zero.

9 — Electrical Characteristics

9.1 — Absolute Maximum Ratings *

Ambient temperature . . . . . -40°C to +125°C
Storage temperature . . . . . -65°C to +150°C
Power supply voltage . . . . . -0.5V to +7V
Input voltage . . . . . -0.5V to VCC + 0.5V

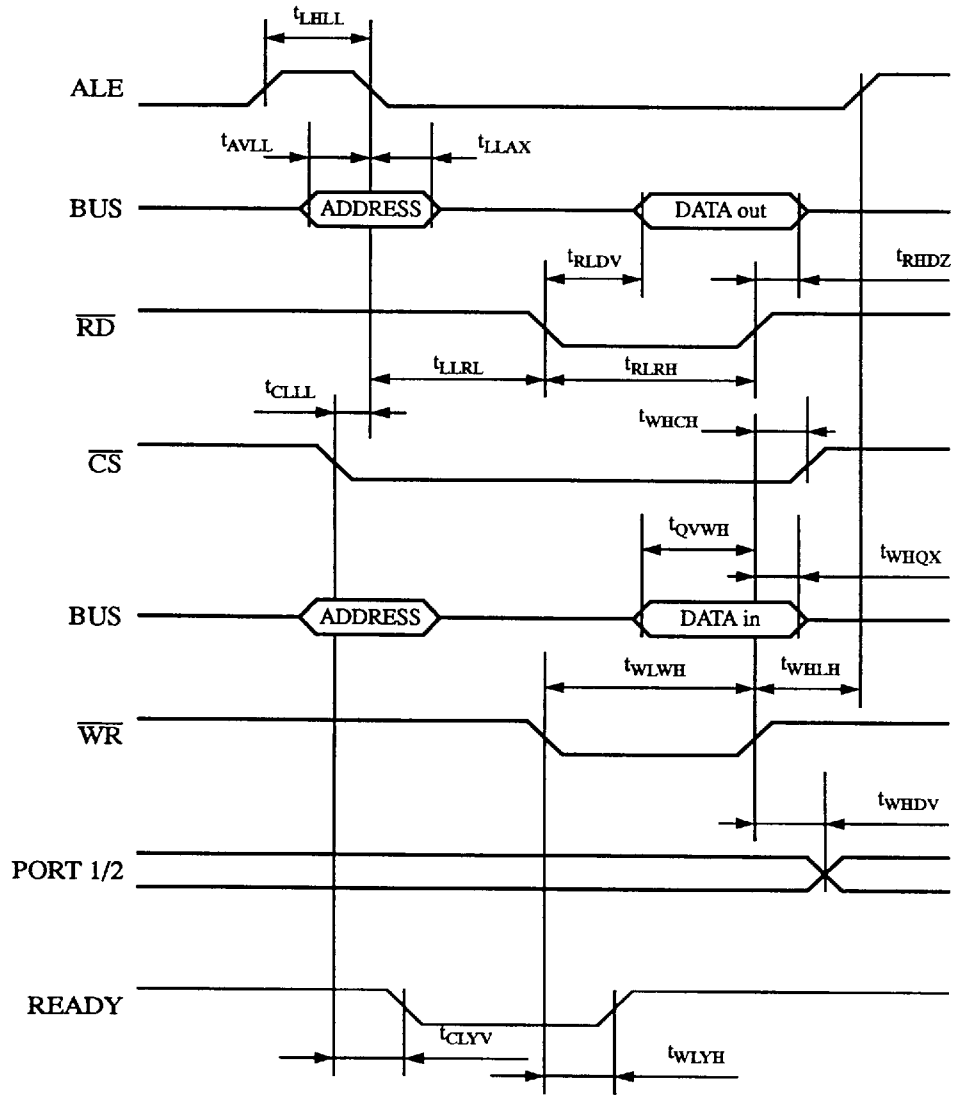
*NOTICE :Stresses at or above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions exceeding those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions may affect device reliability.

9.2 — DC Characteristics TA = -40°C to +125°C; VCC = 5V ±10%; VSS = 0V

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	CMOS Input low voltage TTL Input low voltage		1.5 0.8	V	
V_{IH}	CMOS Input high voltage TTL Input high voltage	3.5 2.2		V	
V_{OL}	Output low voltage		0.4	V	I_{OL}=3.2mA, V_{CC} min.
V_{OH}	Output high voltage	2.4		V	I_{OH}=-3.2mA,V_{CC} min.
I_L	Input leakage current No Pullup/Pulldown With Pullup With Pulldown		 ±5 -50 +50	µA	 $0 < V_{IN} < V_{CC}$ $V_{IN} = 0$ $V_{IN} = V_{CC}$
I_{OZ}	Three-state output leakage current		±5	µA	V_{OUT} = 0 or V_{CC}
I_{CC}	Supply current				
C_{IO}	I/O Buffer capacitance		10	pF	Not tested

9.3 — AC Characteristics

8/16 Bit Multiplexed Intel Modes (Modes 0 & 1)



AC Characteristics (continued)

8/16 Bit Multiplexed Intel Modes (Modes 0 & 1)

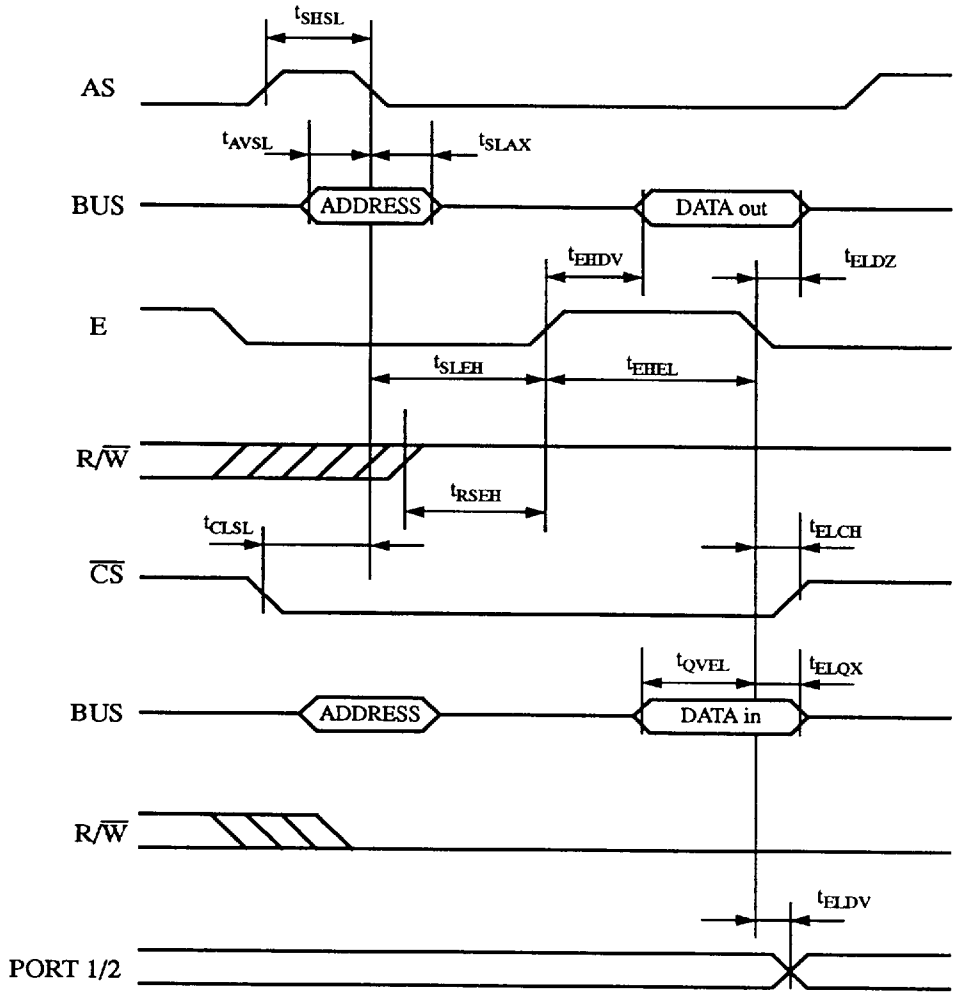
Conditions : TA = -40°C to +125°C; VCC = 5V ±10%; VSS = 0V; CL =50pF

Symbol	Parameter	Min	Max
$1/t_{XTAL}$	Oscillator frequency	4 MHz	32 MHz
t_{AVLL}	Address valid to ALE low	20 ns	
t_{LLAX}	Address hold to ALE low	20 ns	
t_{LHLL}	ALE high time	60 ns	
t_{LLRL}	ALE low to $\overline{RD}$ low	90ns	
t_{CLLL}	$\overline{CS}$ low to ALE low	20 ns	
t_{QVWH}	Data setup to $\overline{WR}$ high	30 ns	
t_{WHQX}	Input Data hold after $\overline{WR}$ high	20 ns	
t_{WLWH}	$\overline{WR}$ pulse width	40 ns	
t_{WHLH}	$\overline{WR}$ high to next ALE high	20 ns	
t_{WHCH}	$\overline{WR}$ high to $\overline{CS}$ high	0 ns	
t_{RLRH}	$\overline{RD}$ pulse width	180 ns	
t_{RLDV}	$\overline{RD}$ low to Data valid for Hardware Registers / mode 0		60 ns
	$\overline{RD}$ low to Data valid for RAM Registers /mode 0		180 ns
	$\overline{RD}$ low to Data valid for HSRR / mode 1		60 ns
	$\overline{RD}$ low to Data valid / mode 1		180 ns
t_{RHDZ}	Data float after $\overline{RD}$ high	0 ns	45 ns
t_{RLYH}	$\overline{RD}$ low to READY float		180 ns
t_{WHDV}	$\overline{WR}$ high to output Data valid on Port 1/2 for mode 0		50 ns
	$\overline{WR}$ high to output Data valid on Port 1/2 for mode 1		180 ns
t_{CLYV}	$\overline{CS}$ low to READY low		40 ns
t_{WLYH}	$\overline{WR}$ low to READY float		40 ns

NOTE: References to $\overline{WR}$ also pertain to $\overline{WRH}$

AC Characteristics (continued)

8 Bit Multiplexed Non-Intel Mode (Mode 2)



AC Characteristics (continued)

8 Bit Multiplexed Non-Intel Mode (Mode 2)

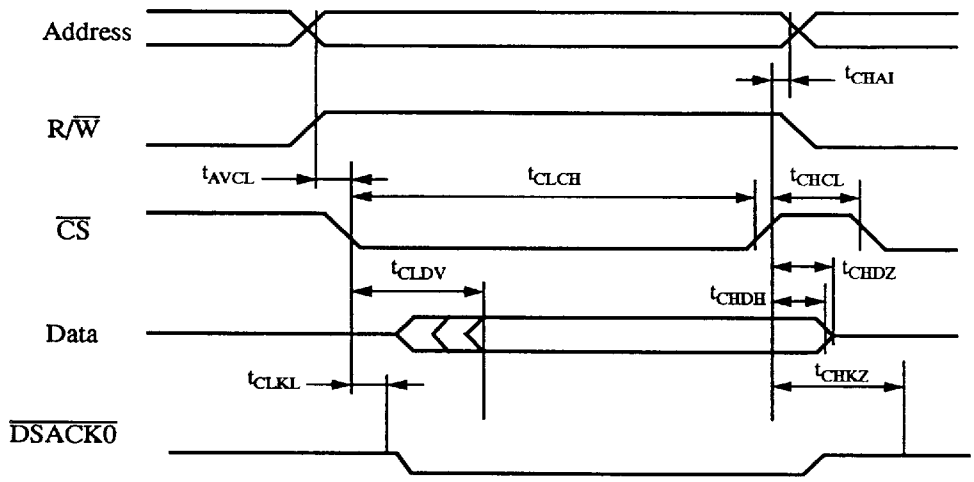
Conditions : TA = -40°C to +125°C; VCC = 5V ±10%; VSS = 0V; CL = 50pF

Symbol	Parameter	Min	Max
$1/t_{XTAL}$	Oscillator frequency	4 MHz	32 MHz
t_{AVSL}	Address valid to AS low	20 ns	
t_{SLAX}	Address hold after AS low	20 ns	
t_{ELDZ}	Data float after E low	0 ns	45 ns
t_{EHDV}	E high to Data valid for Hardware Registers		60 ns
	E high to Data valid for Ram Registers		180 ns
t_{QVEL}	Data setup to E low	30 ns	
t_{ELQX}	Input Data hold after E low	20 ns	
t_{ELDV}	E high to output Data valid on Port 1/2		60 ns
t_{EHEL}	E high time	180 ns	
t_{SHSL}	AS high time	60 ns	
t_{RSEH}	Setup time of R/$\overline{W}$ to E high	30 ns	
t_{SLEH}	AS low to E high	90 ns	
t_{CLSL}	$\overline{CS}$ low to AS low	20 ns	
t_{EHCH}	E low to $\overline{CS}$ high	0 ns	

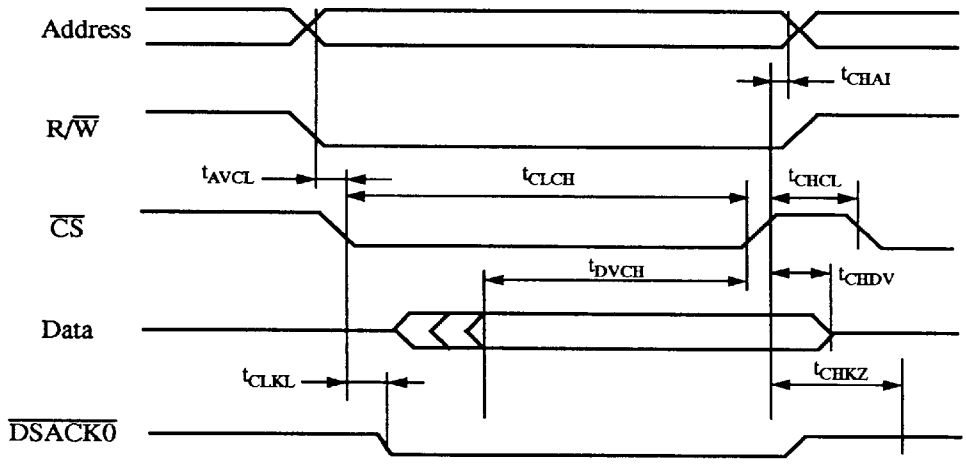
AC Characteristics (continued)

8 Bit Non-Multiplexed Asynchronous Mode (Mode 3)

READ CYCLE



WRITE CYCLE



AC Characteristics (continued)

8 Bit Non-Multiplexed Asynchronous Mode (Mode 3)

Conditions : TA = -40°C to +125°C; VCC = 5V ±10%; VSS = 0V; CL = 50pF

Symbol	Parameter	Min	Max
$1/t_{XTAL}$	Oscillator frequency	4 MHz	32 MHz
t_{AVCL}	Address or R/$\overline{W}$ valid to $\overline{CS}$ low setup	20 ns	
t_{CLDV}	$\overline{CS}$ low to Data valid for Hardware Registers		60 ns
	$\overline{CS}$ low to Data valid for RAM Registers		180 ns
t_{CHDV}	29C462 input Data hold after $\overline{CS}$ high	25 ns	
t_{CHDH}	29C462 output Data hold after $\overline{CS}$ high	0 ns	
t_{CHDZ}	$\overline{CS}$ high to output Data float		35 ns
t_{CHKZ}	$\overline{CS}$ high to $\overline{DSACK0}$ float		25 ns
t_{CHCL}	$\overline{CS}$ width between successive cycles	25 ns	
t_{CHAI}	$\overline{CS}$ high to address or R/$\overline{W}$ invalid	10 ns	
t_{CLCH}	$\overline{CS}$ width low	180 ns	
t_{DVCH}	CPU write Data valid to $\overline{CS}$ high	15 ns	
t_{CLKL}	$\overline{CS}$ low to $\overline{DSACK0}$ low		180 ns

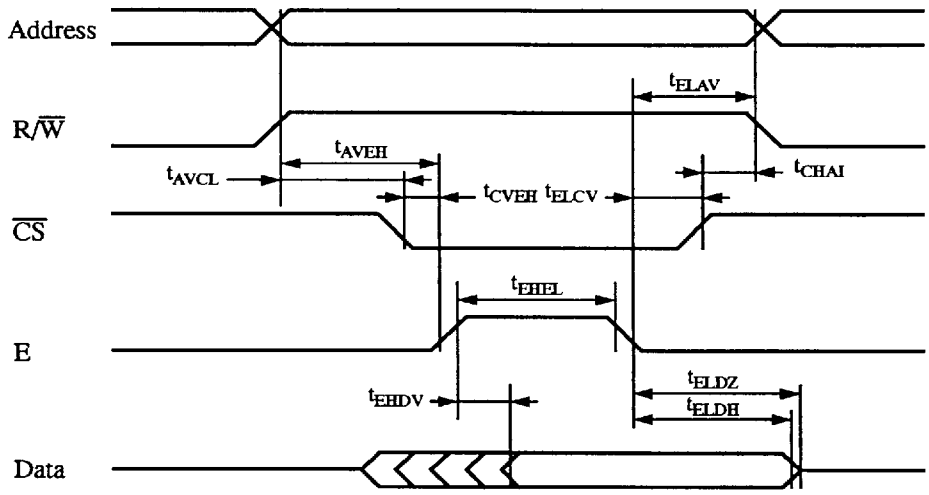
Note:

$\overline{RD}$ and ALE must be tied high in this mode.

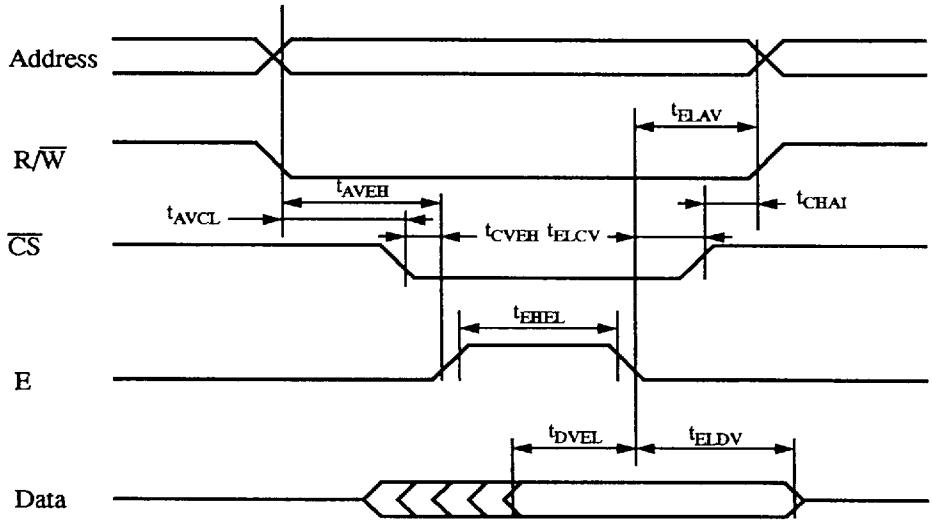
AC Characteristics (continued)

8 Bit Non-Multiplexed Synchronous Mode (Mode 3)

READ CYCLE



WRITE CYCLE



AC Characteristics (continued)

8 Bit Non-Multiplexed Synchronous Mode (Mode 3)

Conditions : TA = -40°C to +125°C; VCC = 5V ±10%; VSS = 0V; CL = 50pF

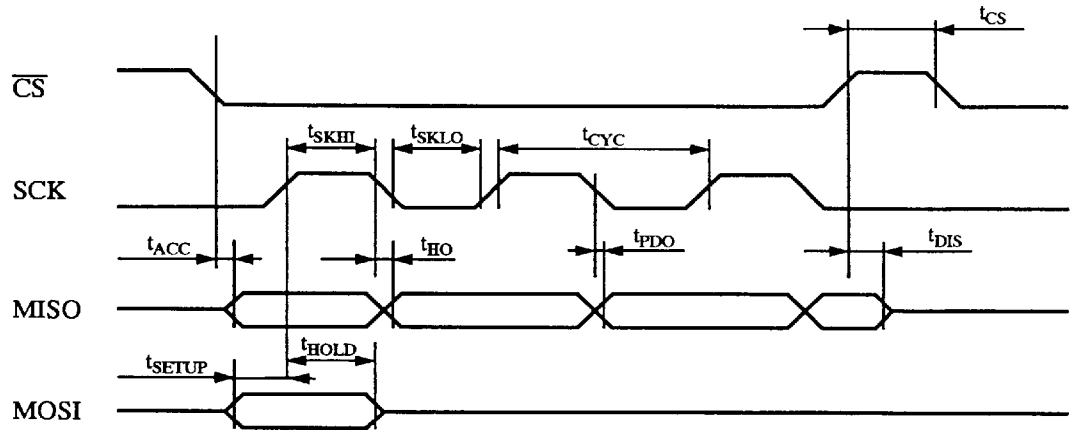
Symbol	Parameter	Min	Max
$1/t_{XTAL}$	Oscillator frequency	4 MHz	32 MHz
t_{EHDV}	E high to Data valid for Hardware Registers		60 ns
	E high to Data valid for RAM Registers		180 ns
t_{ELDH}	Data hold after E low for a read cycle	0 ns	
t_{ELDZ}	Data float after E low		35 ns
t_{ELDV}	Data hold after E low for a write cycle	25 ns	
t_{AVEH}	Address and R/$\overline{W}$ to E setup	25 ns	
t_{ELAV}	Address and R/$\overline{W}$ valid after E falls	15 ns	
t_{CVEH}	$\overline{CS}$ valid to E high	5 ns	
t_{ELCV}	$\overline{CS}$ valid after E low	5 ns	
t_{DVEL}	Data setup to E low	55 ns	
t_{EHEL}	E active high		180 ns
t_{AVCL}	Address or R/$\overline{W}$ to $\overline{CS}$ low setup	0 ns	
t_{CHAI}	$\overline{CS}$ high to Address invalid	10 ns	

Note:

ALE must be tied high in this mode.

AC Characteristics (continued)

Serial Interface Mode



AC Characteristics (continued)

Serial Interface Mode

Conditions : TA = -40°C to +125°C; VCC = 5V ±10%; VSS = 0V; CL = 50pF

Symbol	Parameter	Min	Max
t_{CYC}	1/SPI clock		8 MHz
t_{SKHI}	Minimum Clock High Time	100 ns	
t_{SKLO}	Minimum Clock Low Time	100 ns	
t_{ACC}	Access Time	50 ns	
t_{PDO}	Maximum Data Out Delay Time		60 ns
t_{HO}	Minimum Data Out Hold Time	0 ns	
t_{DIS}	Maximum Data Out Disable Time		50 ns
t_{SETUP}	Minimum Data Setup Time	35 ns	
t_{HOLD}	Minimum Data Hold Time	35 ns	
t_{CS}	Minimum Time between Consecutive $\overline{CS}$ Assertions	180ns	