Stand-alone CAN-controller

82C200

1 FEATURES

- Multi-master architecture
- Interfaces with a large variety of microcontrollers
- Bus access priority (determined by the message identifier)
- 2032 message identifiers
- Guaranteed latency time for high priority messages
- Powerful error handling capability
- Data length from 0 to 8 bytes
- Configurable bus interface
- Programmable clock output
- Multicast and Broadcast message facility
- Non destructive bit-wise arbitration
- Non-return-to-zero (NRZ) coding/decoding with bit-stuffing
- Programmable transfer rate (up to 1 Mbit/s)
- Programmable output driver configuration
- Suitable for use in a wide range of networks including the SAE networks Class A, B and C
- 16 MHz clock frequency
- –40 to +85/125 °C operating temperature.

2 GENERAL DESCRIPTION

The PCA82C200; PCF82C200 (hereafter generically referred to as PCX82C200) is a highly integrated stand-alone controller for the controller area network (CAN) used within automotive and general industrial environments. The temperature range includes an automotive temperature range version (PCA82C200) of –40 to +125 °C and a –40 to +85 °C version (PCF82C200) for general applications.

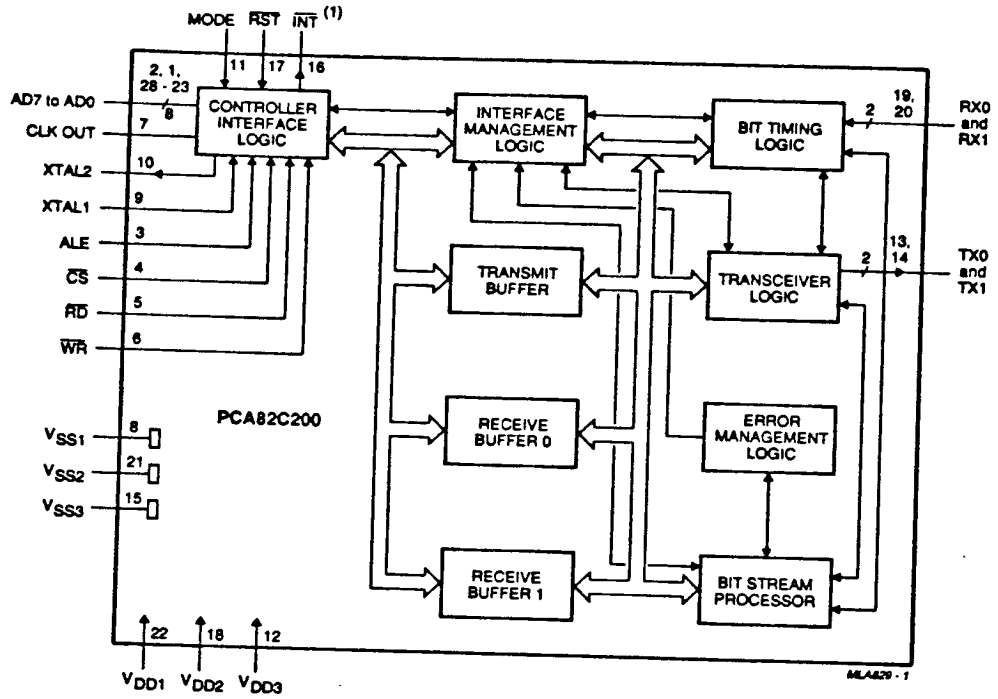
The PCX82C200 contains all necessary features required to implement a high performance communication protocol. The PCX82C200 with a simple bus line connection performs all the functions of the physical and data-link layers. The application layer of an Electronic Control Unit (ECU) is provided by a microcontroller, to which the PCX82C200 provides a versatile interface. The use of the PCX82C200 in an automotive or general industrial environment, results in a reduced wiring harness and an enhanced diagnostic and supervisory capability.

3 ORDERING AND PACKAGE INFORMATION

PHILIPS PART ORDER NUMBER PART MARKING	PHILIPS NORTH AMERICA PART ORDER NUMBER[1]	PACKAGE				TEMPERATURE RANGE (°C)
		PINS	PIN POSITION	MATERIAL	CODE	
PCA82C200P	PCA82C200PN	28	DIL	plastic	SOT117	–40 to +125
PCA82C200T	PCA82C200TD	28	SO28	plastic	SOT136A	–40 to +125
PCF82C200P	PCF82C200PN	28	DIL	plastic	SOT117	–40 to +85
PCF82C200T	PCF82C200TD	28	SO28	plastic	SOT136A	–40 to +85

NOTE:

1. Parts ordered by the Philips North America part number will be marked with the Philips part marking.



(1) During 'sleep mode' this pin functions as an input.

Fig.1 Block diagram.

4 PINNING

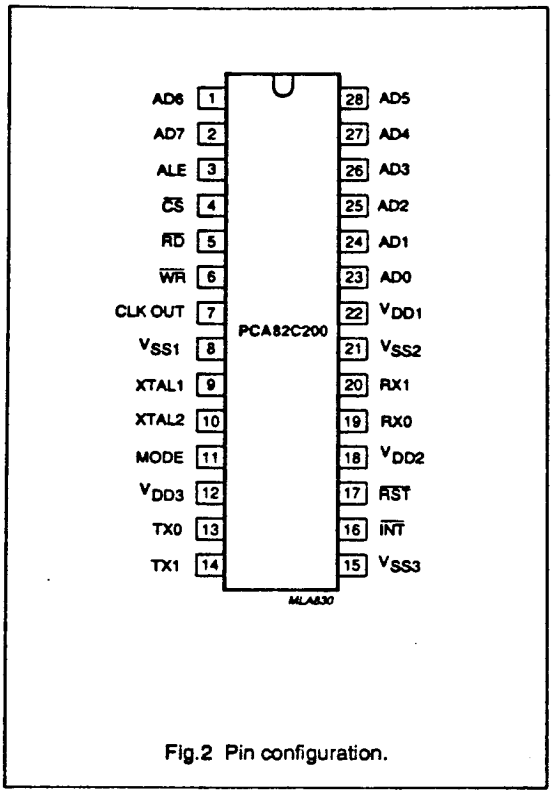


Fig.2 Pin configuration.

Pinning description

SYMBOL	PIN	DESCRIPTION
AD7–AD0	2, 1, 28–23	Multiplexed address/data bus.
ALE	3	ALE signal (Intel mode) or AS input signal (Motorola mode).
$\overline{CS}$	4	Chip select input, LOW level allows access to the PCX82C200.
$\overline{RD}$	5	$\overline{RD}$ signal (Intel mode) or E enable signal (Motorola mode) from the microcontroller.
$\overline{WR}$	6	$\overline{WR}$ signal (Intel mode) or RD/$\overline{WR}$ signal (Motorola mode) from the microcontroller.
CLK OUT	7	Clock output signal produced by the PCX82C200 for the microcontroller. The clock signal is derived from the built-in oscillator, via the programmable divider (see section 6.5). This output is capable of driving one CMOS or NMOS load.
V_{SS1}	8	Ground potential for the logic circuits.
XTAL1 (note 1)	9	Input to the oscillator's amplifier. External oscillator signal is input via this pin.
XTAL2 (note 1)	10	Output from the oscillator's amplifier. Output must be left open when an external oscillator signal is used.
MODE	11	Mode select input: connected to V_{DD} selects Intel mode; connected to V_{SS} selects Motorola mode.
V_{DD3}	12	5 V power supply for the output driver.
TX0	13	Output from the output-driver 0 to the physical bus-line.
TX1	14	Output from the output-driver 1 to the physical bus-line.
V_{SS3}	15	Ground potential for the output driver.
$\overline{INT}$	16	Interrupt output, used to interrupt the microcontroller (see section 6.2.4). $\overline{INT}$ is active if the Interrupt Register contains a logic HIGH bit (present). $\overline{INT}$ is an open drain output and is designed to be a wired-OR with other $\overline{INT}$ outputs within the system. A LOW level on this pin will reactivate the IC from the sleep mode (see section 6.2.2).
$\overline{RST}$	17	Reset input, used to reset the CAN interface (LOW level). Automatic power-ON reset can be obtained by connecting $\overline{RST}$ via a capacitor to V_{SS} and via a resistor to V_{DD} (e.g. C = 1 µF; R = 50 kΩ).
V_{DD2}	18	5 V power supply for the input comparator.
RX0–RX1	19, 20	Input from the physical bus-line to the input comparator of the PCX82C200. A dominant level will wake-up the PCX82C200. A recessive level is read if RX0 is higher than RX1 and vice versa for the dominant level.
V_{SS2}	21	Ground potential for the input comparator.
V_{DD1}	22	5 V power supply for the logic circuits.

Note

1. XTAL1 and XTAL2 pins should be connected to V_{SS} via 15 pF capacitors.

5 FUNCTIONAL DESCRIPTION

The PCX82C200 contains all necessary hardware for a high performance serial network communication (see Fig.1). The PCX82C200 controls the communication flow through the area network using the CAN-protocol. The PCX82C200 meets the following automotive requirements:

- short message length
- guaranteed latency time for urgent messages
- bus access priority, determined by the message identifier
- powerful error handling capability
- configuration flexibility to allow area network expansion.

The latency time defines the period between the initiation (Transmission Request) and the start of the transmission on the bus. Latency time is dependent on a variety of bus related conditions. In the case of a message being transmitted on the bus and one distortion the latency time can be up to 149 bit times (worst case). For more information see section 7.

5.1 Interface Management Logic (IML)

The IML interprets commands from the microcontroller, allocates the message buffers (TBF, RBF0 and RBF1) and provides interrupts and status information to the microcontroller.

5.2 Transmit Buffer (TBF)

The TBF is a 10 byte memory into which the microcontroller writes messages which are to be transmitted over the CAN network.

5.3 Receive Buffers (RBF0 AND RBF1)

The RBF0 and RBF1 are each 10 byte memories which are alternatively used to store messages received from the CAN network. The CPU can process one message while another is being received.

5.4 Bit Stream Processor (BSP)

The BSP is a sequencer, controlling the data stream between the Transmit Buffer, the Receive Buffer (parallel data) and the CAN-bus (serial data).

5.5 Bit Timing Logic (BTL)

The BTL synchronizes the PCX82C200 to the bitstream on the CAN-bus.

5.6 Transceiver Control Logic (TCL)

The TCL controls the output driver.

5.7 Error Management Logic (EML)

The EML performs the error confinement according to the CAN-protocol.

5.8 Controller Interface Logic (CIL)

The CIL is the interface to the external microcontroller. The PCX82C200 can directly interface with a variety of microcontrollers.

6 CONTROL SEGMENT AND MESSAGE BUFFER DESCRIPTION

The PCX82C200 appears to a microcontroller as a memory-mapped I/O device due to the on-chip RAM, guaranteeing the independent operation of both devices.

6.1 Address allocation

The address area of the PCX82C200 consists of the Control Segment and the message buffers. The Control Segment is programmed during an initialization download in order to configure communication parameters (e.g. bit timing). Communication over the CAN-bus is also controlled via this segment by the CPU. During initialization the CLOCK OUT signal may be programmed to a value determined by the microcontroller (see Fig.1). A message which is to be transmitted, must be written to the Transmit Buffer. After a successful reception the microcontroller may read the message from the Receive Buffer and then release it for further use.

6.2 Control Segment layout

The exchange of status, control and command signals between the microcontroller and the PCX82C200 is performed in the control segment. The layout of this segment is shown in Fig.3. After an initial down-load, the contents of the registers Acceptance Code, Acceptance Mask, Bus Timing Registers 0 and 1, and Output Control should not be changed. These registers may only be accessed when the Reset Request bit in the Control Register, is set HIGH (see section 6.2.1).

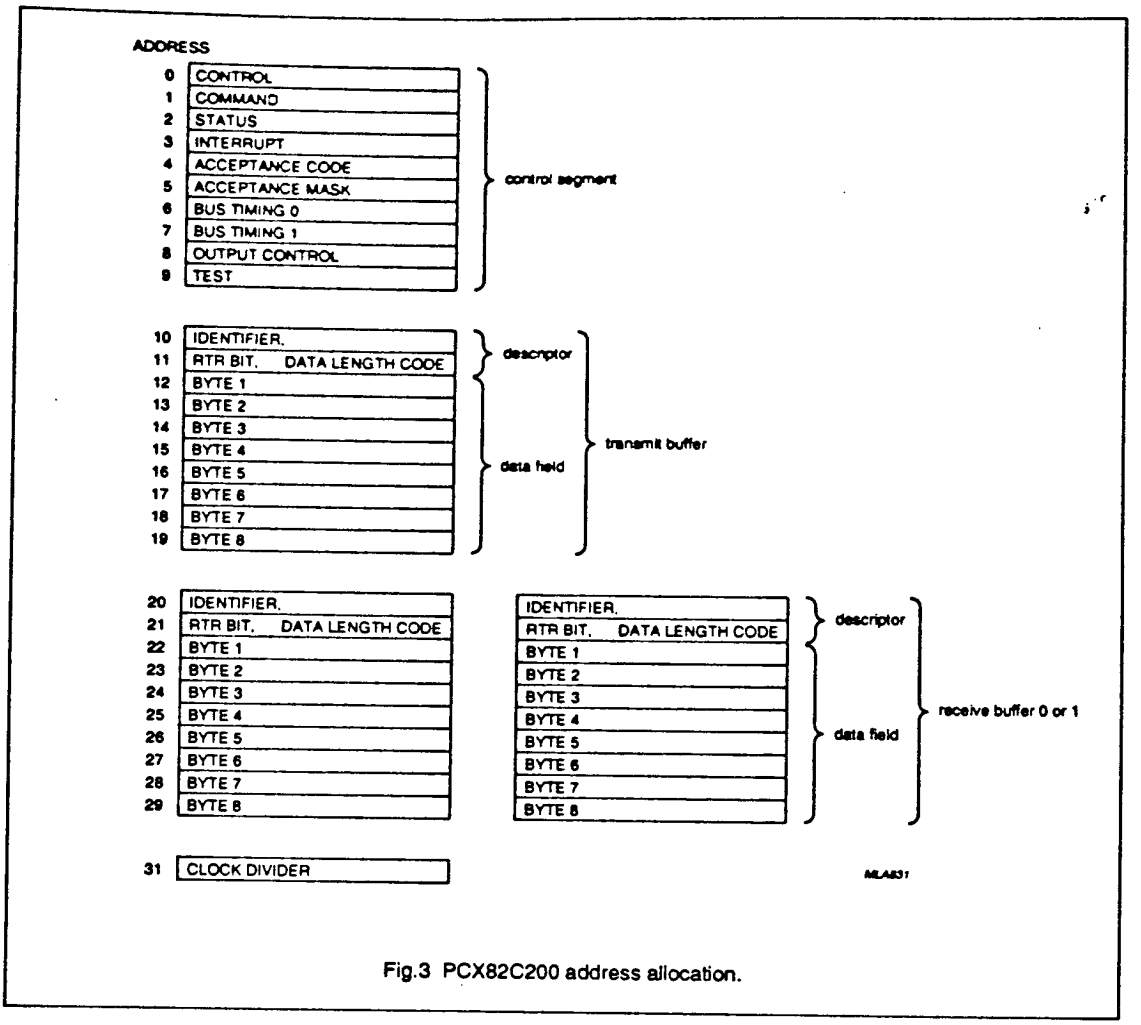


Fig.3 PCX82C200 address allocation.

Table 1 Register map

TITLE	ADDR	7	6	5	4	3	2	1	0
Control Segment									
Control Register	0	Test Mode	Sync	reserved	Overrun Interrupt Enable	Error Interrupt Enable	Transmit Interrupt Enable	Receive Interrupt Enable	Reset Request
Command Register	1	reserved	reserved	reserved	Goto Sleep	Clear Overrun Status	Release Receive Buffer	Abort Trans-mission	Trans-mission Request
Status Register	2	Bus Status	Error Status	Transmit Status	Receive Status	Trans-mission Complete Status	Transmit Buffer Access	Data Overrun	Receive Buffer Stat
Interrupt Register	3	reserved	reserved	reserved	Wake-Up Interrupt	Overrun Interrupt	Error Interrupt	Transmit Interrupt	Receive Interrupt
Acceptance Code Register	4	AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0
Acceptance Mask Register	5	AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0
Bus Timing Register 0	6	SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0
Bus Timing Register 1	7	SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0
Output Control Register	8	OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0
Test Register (note 1)	9	reserved	reserved	Map Internal Register	Connect RX Buffer 0 CPU	Connect TX Buffer CPU	Access Internal Bus	Normal RAM Connect	Float Output Driver
Transmit Buffer									
Identifier	10	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
RTR, Data Length Code	11	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
bytes 1–8	12–19	Data	Data	Data	Data	Data	Data	Data	Data
Receive Buffer 0/1									
Identifier	20	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
RTR, Data Length Code	21	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
bytes 1–8	22–29	Data	Data	Data	Data	Data	Data	Data	Data
Clock Divider	31	reserved	reserved	reserved	reserved	reserved	CD.2	CD.1	CD.0

Notes

1. The Test Register is used for production testing only.
2. Register 30 is not implemented.

6.2.1 Control Register (CR)

The contents of the Control Register are used to change the behaviour of the PCX82C200. Control bits may be set or reset by the attached microcontroller which uses the Control Register as a read/write memory.

Table 2 Description of the Control Register bits

CR	ADDRESS 0			
BIT	SYMBOL	NAME	VALUE	FUNCTION
CR.7	TM	Test Mode (note 1)	HIGH (enabled)	PCX82C200 enters Test Mode (normal operation impossible).
			LOW (disabled)	Normal operating mode.
CR.6	S	Sync (note 2)	HIGH (2 edges)	Bus-line transitions from recessive-to-dominant and vice versa are used for resynchronization (see sections 7.2 and 8).
			LOW (1 edge)	Only transitions from recessive-to-dominant are used for resynchronization.
CR.5	–	–	–	Reserved.
CR.4	OIE	Overrun Interrupt Enable	HIGH (enabled)	If the Data Overrun bit is set (see section 6.2.3), the microcontroller receives an Overrun Interrupt signal.
			LOW (disabled)	Microcontroller receives no Overrun Interrupt signal from the PCX82C200.
CR.3	EIE	Error Interrupt Enable	HIGH (enabled)	If the Error or Bus Status change (see section 6.2.3), the microcontroller receives an Error Interrupt signal.
			LOW (disabled)	Microcontroller receives no Error Interrupt signal.
CR.2	TIE	Transmit Interrupt Enable	HIGH (enabled)	When a message has been successfully transmitted or the transmit buffer is accessible again, (e.g. after an Abort Transmission command) the PCX82C200 transmits a Transmit Interrupt signal to the microcontroller.
			LOW (disabled)	No transmission of the Transmit Interrupt signal by the PCX82C200 to the microcontroller.
CR.1	RIE	Receive Interrupt Enable	HIGH (enabled)	When a message has been received without errors, the PCX82C200 transmits a Receive Interrupt signal to the microcontroller.
			LOW (disabled)	No transmission of the Receive Interrupt signal by the PCX82C200 to the microcontroller.
CR.0	RR	Reset Request (note 3)	HIGH (present)	Detection of a Reset Request results in the PCX82C200 aborting the current transmission or reception of a message and entering the reset state.
			LOW (absent)	On the HIGH-to-LOW transition of the Reset Request bit, the PCX82C200 returns to its normal operating state.

Notes

1. The Test Mode is intended for factory testing and not for customer use.
2. The Sync bit should only be modified if the Reset Request bit is set HIGH (present), otherwise it is ignored. It is possible to set the Sync bit while the Reset Request bit is changed from HIGH to LOW.
3. During an external reset ($\overline{RST}$ = LOW) or when the Bus Status bit is set HIGH (Bus-Off), the IML forces the Reset Request HIGH (present). During an external reset the microcontroller cannot set the Reset Request bit LOW (absent). Therefore, after having set the Reset Request bit LOW (absent), the microcontroller must check this bit to ensure that the external reset pin is not being held HIGH (present). After the Reset Request bit is set LOW (absent) the PCX82C200 will wait for:
   - one occurrence of the Bus-Free signal (11 recessive bits, see section 8.9.6), if the preceding reset (Reset Request = HIGH) was due to an external reset or a microcontroller initiated reset
   - 128 occurrences of Bus-Free, if the preceding reset (Reset Request = HIGH) was due to a PCX82C200 initiated Bus-Off, before re-entering the Bus-On mode (see section 8.9).

   When Reset Request is set HIGH (present), for whatever reason, the control, command, status and interrupt bits are affected, see Table 3. When Reset Request is set HIGH (present) the registers at addresses 4 to 8 are accessible but the TBF is not.

Table 3 Effects of setting the Reset Request bit HIGH (present)

TYPE	BIT	FUNCTION	EFFECT
Control	CR.7	Test Mode	LOW (disabled)
Command	CMR.4	Goto Sleep	LOW (wake-up)
	CMR.3	Clear Overrun Status	HIGH (clear)
	CMR.2	Release Receive Buffer	HIGH (released)
	CMR.1	Abort Transmission	LOW (absent)
	CMR.0	Transmission Request	LOW (absent)
Status	SR.7	Bus Status	LOW (Bus-On) (note 1)
	SR.6	Error Status	LOW (no error) (note 1)
	SR.5	Transmit Status	LOW (idle)
	SR.4	Receive Status	LOW (idle)
	SR.3	Transmission Complete Status	HIGH (complete)
	SR.2	Transmit Buffer Access	HIGH (released)
	SR.1	Data Overrun	LOW (absent)
	SR.0	Receive Buffer Status	LOW (empty)
Interrupt	IR.3	Overrun Interrupt	LOW (reset)
	IR.1	Transmit Interrupt	LOW (reset)
	IR.0	Receive Interrupt	LOW (reset)

Note

1. Only after an external reset; see note 1 to Table 5 "Description of the Status Register bits".

6.2.2 Command Register (CMR)

A command bit initiates an action within the transfer layer of the PCX82C200. The Command Register appears to the microcontroller as a write only memory. If a read access is performed to this address the byte 11111111 (binary) is returned.

Table 4 Description of the Command Register bits

CMR	ADDRESS 1			
BIT	**SYMBOL**	**NAME**	**VALUE**	**FUNCTION**
CMR.7	–	–	–	Reserved.
CMR.6	–	–	–	Reserved.
CMR.5	–	–	–	Reserved.
CMR.4	GTS	GoTo Sleep (note 1)	HIGH (sleep) LOW (wake up)	The PCX82C200 enters sleep mode, if the $\overline{INT}$ = HIGH (no interrupt signal from the PCX82C200 to the microcontroller pending or external source pending) and there is no bus activity. The PCX82C200 functions normally.
CMR.3	COS	Clear Overrun Status (note 2)	HIGH (clear) LOW (no action)	The Data Overrun status bit is set to LOW (see section 6.2.3). No action.
CMR.2	RRB	Release Receive Buffer (note 3)	HIGH (released) LOW (no action)	The receive buffer attached to the microcontroller is released. No action.
CMR.1	AT	Abort Transmission (note 4)	HIGH (present) LOW (absent)	If not already in progress, a pending Transmission Request is cancelled. No action.
CMR.0	TR	transmission Request (note 5)	HIGH (present) LOW (absent)	A message shall be transmitted. No action.

Notes

1. The PCX82C200 will enter sleep mode, if Goto Sleep is set HIGH (sleep), there is no bus activity and $\overline{INT}$ = HIGH (inactive). After sleep mode is set, the CLK OUT signal continues until at least 15 bit times have passed. The PCX82C200 will wake up when one of the three previously mentioned conditions is negated: after Goto Sleep is set LOW (wake up), there is bus activity or $\overline{INT}$ is driven LOW (active). On wake up, the oscillator is started and a Wake-Up Interrupt (see section 6.2.4) is generated. A PCX82C200 which is sleeping and then awakened by bus activity will not be able to receive this message until it detects a Bus-Free signal (see section 8.9.6).
2. This command bit is used to acknowledge the Data Overrun condition signalled by the Data Overrun status bit. It may be given or set at the same time as a Release Receive Buffer command bit.
3. After reading the contents of the Receive Buffer (RBF0 or RBF1) the microcontroller must release this buffer by setting the Release Receive Buffer bit HIGH (released). This may result in another message becoming immediately available.
4. The Abort Transmission bit is used when the microcontroller requires the suspension of the previously requested transmission, for example to transmit an urgent message. A transmission already in progress is not stopped. In order to determine if the original message had been transmitted successfully, or aborted, the Transmission Complete status bit should be checked after the Transmit Buffer Access bit has been set HIGH (released) or a Transmit Interrupt has been generated (see section 6.2.4).
5. If the Transmission Request bit was set HIGH in a previous command, it cannot be cancelled by setting the Transmission Request bit LOW (absent). Cancellation of the requested transmission may be performed by setting the Abort Transmission bit HIGH (present).

6.2.3 Status Register (SR)

The contents of the Status Register reflect the status of the PCX82C200 bus controller. The Status Register appears to the microcontroller as a read only memory.

Table 5 Description of the Status Register bits

SR	ADDRESS 2			
BIT	**SYMBOL**	**NAME**	**VALUE**	**FUNCTION**
SR.7	BS	Bus Status (note 1)	HIGH (Bus-Off)	The PCX82C200 is not involved in bus activities.
			LOW (Bus-On)	The PCX82C200 is involved in bus activities.
SR.6	ES	Error Status	HIGH (error)	At least one of the Error Counters (see section 8.10.3) has reached the microcontroller Warning Limit.
			LOW (ok)	Both Error Counters have not reached the Warning Limit.
SR.5	TS	Transmit Status (note 2)	HIGH (transmit)	The PCX82C200 is transmitting a message.
			LOW (idle)	No message is transmitted.
SR.4	RS	Receive Status (note 2)	HIGH (receive)	The PCX82C200 is receiving a message.
			LOW (idle)	No message is received.
SR.3	TCS	Transmission Complete Status (note 3)	HIGH (complete)	Last requested transmission has been successfully completed.
			LOW (incomplete)	Previously requested transmission is not yet completed.
SR.2	TBS	Transmit Buffer Access (note 3)	HIGH (released)	The microcontroller may write a message into the TBF.
			LOW (locked)	The microcontroller cannot access the Transmit Buffer. A message is either waiting for transmission or is in the process of being transmitted.
SR.1	DO	Data Overrun (note 4)	HIGH (overrun)	This bit is set HIGH (Overrun), when both Receive Buffers are full and the first byte of another message should be stored.
			LOW (absent)	No data overrun has occurred since the Clear Overrun command was given.
SR.0	RBS	Receive Buffer Status (note 5)	HIGH (full)	This bit is set when a new message is available.
			LOW (empty)	No message has become available since the last Release Receive Buffer command bit was set.

Notes

1. When the Bus Status bit is set HIGH (Bus-Off), the PCX82C200 will set the Reset Request bit HIGH (present). It will stay in this state until the microcontroller sets the Reset Request bit LOW (absent). Once this is completed the PCX82C200 will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) before setting the Bus Status bit LOW (Bus-On), the Error Status bit LOW (ok) and resetting the Error Counters.
2. If both the Receive Status and Transmit Status bits are LOW (idle) the CAN-bus is idle.
3. If the microcontroller tries to write to the Transmit Buffer when the Transmit Buffer Access bit is LOW (locked), the written bytes will not be accepted and will be lost without this being signalled. The Transmission Complete Status bit is set LOW (incomplete) whenever the Transmission Request bit is set HIGH (present). If an Abort Transmission command is issued, the Transmit Buffer will be released. If the message, which was requested and then aborted, was not transmitted, the Transmission Complete Status bit will remain LOW.
4. If Data Overrun = HIGH (Overrun) is detected, the currently received message is dropped. A transmitted message, granted acceptance, is also stored in a Receive Buffer. This occurs because it is not known if the PCX82C200 will lose arbitration and so become a receiver of the message. If no Receive Buffer is available, Data Overrun is signalled.
5. If the command bit Release Receive Buffer is set HIGH (released) by the microcontroller, the Receive Buffer Status bit is set LOW (empty) by IML. When a new message is stored in any of the receive buffers, the Receive Buffer Status bit is set HIGH (full) again.

6.2.4 Interrupt Register (IR)

The Interrupt Register allows the identification of an interrupt source. When one or more bits of this register are set, the $\overline{INT}$ pin is activated. All bits are reset by the PCX82C200 after this register is read by the microcontroller. This register appears to the microcontroller as a read only memory.

Table 6 Description of the Interrupt Register bits

IR	ADDRESS 3			
BIT	**SYMBOL**	**NAME**	**VALUE**	**FUNCTION**
IR.7	–	–	–	Reserved.
IR.6	–	–	–	Reserved.
IR.5	–	–	–	Reserved.
IR.4	WUI	Wake-Up Interrupt	HIGH (set)	The Wake-Up Interrupt bit is set HIGH, when the sleep mode is left (see section 6.2.2).
			LOW (reset)	Wake-Up Interrupt bit is reset by a read access of Interrupt Register by the microcontroller.
IR.3	OI	Overrun Interrupt (note 1)	HIGH (set)	This bit is set HIGH, if both Receive Buffers contain a message and the first byte of another message should be stored (passed acceptance), and the Overrun Interrupt Enable is HIGH (enabled).
			LOW (reset)	Overrun Interrupt bit is reset by a read access of Interrupt Register by the microcontroller.
IR.2	EI	Error Interrupt	HIGH (set)	This bit is set on a change of either the Error Status or Bus Status bits (see section 6.2.3) if the Error Interrupt Enable is HIGH (enabled).
			LOW (reset)	The Error Interrupt bit is reset by a read access of the Interrupt Register by the microcontroller.
IR.1	TI	Transmit Interrupt	HIGH (set)	This bit is set on a change of the Transmit Buffer Access bit from LOW to HIGH (released) and Transmit Interrupt Enable is HIGH (enabled).
			LOW (reset)	Transmit Interrupt bit will be reset after a read access of the Interrupt Register by the microcontroller.
IR.0	RI	Receive Interrupt (note 2)	HIGH (set)	This bit is set when a new message is available in the Receive Buffer and the Receive Interrupt Enable bit is HIGH (enabled).
			LOW (reset)	Receive Interrupt bit is automatically reset by a read access of Interrupt Register by the microcontroller.

Notes

1. Overrun Interrupt bit (if enabled) and Data Overrun bit (see section 6.2.3) are set at the same time.
2. Receive Interrupt bit (if enabled) and Receive Buffer Status bit (see section 6.2.3) are set at the same time.

6.2.5 Acceptance Code Register (ACR)

The Acceptance Code Register is part of the acceptance filter of the PCX82C200. This register can be accessed (read/write), if the Reset Request bit is set HIGH (present). When a message is received which passes the acceptance test and if there is an empty Receive Buffer, then the respective Descriptor and Data Field (see Fig.4) are sequentially stored in this empty buffer. In the case that there is no empty Receive Buffer, the Data Overrun bit is set HIGH (overrun), see sections 6.2.3 and 6.2.4. When the complete message has been correctly received the following occurs:

- the Receive Buffer Status bit is set HIGH (full)
- If the Receive Interrupt Enable bit is set HIGH (enabled), the Receive Interrupt is set HIGH (set).

The Acceptance Code bits (AC.7-AC.0) and the eight most significant bits of the message's Identifier (ID.10-ID.3) must be equal to those bit positions which are marked relevant by the Acceptance Mask bits (AM.7-AM.0). If the following equation is satisfied, acceptance is given:

$$[\,(ID.10\,..\,ID.3) = (AC.7\,..\,AC.0)\,]\ \textit{or}\ (AM.7\,..\,AM.0) = 1111\ 1111\ \textit{binary}$$

During transmission of a message which passes the acceptance test, the message is also written to its own Receive Buffer. If no Receiver Buffer is available, Data Overrun is signalled because it is not known at the start of a message whether the PCX82C200 will lose arbitration and so become a receiver of the message.

Table 7 Acceptance Code Register bits

ACR	ADDRESS 4						
7	6	5	4	3	2	1	0
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0

6.2.6 Acceptance Mask Register (AMR)

The Acceptance Mask Register is part of the acceptance filter of the PCX82C200. This register can be accessed (read/write) if the Reset Request bit is set HIGH (present). The Acceptance Mask Register qualifies which of the corresponding bits of the acceptance code are "relevant" or "don't care" for acceptance filtering.

Table 8 Acceptance Mask Register bits

AMR	ADDRESS 5						
7	6	5	4	3	2	1	0
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0

Table 9 Description of the Acceptance Mask Register bits

ACCEPTANCE MASK BIT	VALUE	COMMENTS
AM.7 to AM.0	HIGH (don't care)	This bit position is "don't care" for the acceptance of a message.
	LOW (relevant)	This bit position is "relevant" for acceptance filtering.

6.2.7 Bus Timing Register 0 (BTR0)

The contents of Bus Timing Register 0 defines the values of Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). This register can be accessed (read/write) if the Reset Request bit is set HIGH (present).

Table 10 Bus Timing Register 0 bits

BTR0	ADDRESS 6						
7	6	5	4	3	2	1	0
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0

Baud Rate Prescaler (BRP)

The period of the system clock t_{SCL} is programmable and determines the individual bit timing. The system clock is calculated using the following equation:

$$t_{SCL} = 2t_{CLK} (32BRP.5 + 16BRP.4 + 8BRP.3 + 4BRP.2 + 2BRP.1 + BRP.0 + 1)$$

t_{CLK} = time period of the PCX82C200 oscillator.

Synchronization Jump Width (SJW)

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must resynchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum number of clock cycles a bit period may be shortened or lengthened by one resynchronization:

$$t_{SJW} = t_{SCL} (2SJW.1 + SJW.0 + 1)$$

For further information on bus timing see sections 6.2.8 and 7.

6.2.8 Bus Timing Register 1 (BTR1)

The contents of Bus Timing Register 1 defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. This register may be accessed (read/write) if the Reset Request bit is set HIGH (present).

Table 11 Bus Timing Register 1 bits

BTR1	ADDRESS 7						
7	6	5	4	3	2	1	0
SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

Sampling (SAM)

Table 12 Selection of sampling

BIT	VALUE	COMMENTS
SAM	HIGH (3 samples)	Three samples are taken.
	LOW (1 sample)	The bus is sampled once.

SAM = LOW (logic 0) is recommended for high speed buses (SAE class C), while SAM = HIGH (logic 1) is recommended for slow/medium speed buses (class A and B) where filtering of spikes on the bus-line is beneficial (see section 7.1.6).

Time Segment 1 (TSEG1) and Time Segment 2 (TSEG2)

TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

$t_{TSEG1} = t_{SCL}$ (8TSEG1.3 + 4TSEG1.2 + 2TSEG1.1 + TSEG1.0 + 1)

$t_{TSEG2} = t_{SCL}$ (4TSEG2.2 + 2TSEG2.1 + TSEG2.0 + 1)

For further information on bus timing see sections 6.2.7 and 7.

6.2.9 Output Control Register (OCR)

The Output Control Register allows, under software control, the set-up of different output driver configurations. This register may be accessed (read/write) if the Reset Request bit is set HIGH (present).

Table 13 Output Control Register bits

OCR	ADDRESS 8						
7	6	5	4	3	2	1	0
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0

If the PCX82C200 is in the sleep mode (Goto Sleep = HIGH) a recessive level is output on the TX0 and TX1 pins. If the PCX82C200 is in the reset state (Reset Request = HIGH) the output drivers are floating.

Normal Output Mode

In Normal Output Mode the bit sequence (TXD) is sent via TX0 and TX1. The voltage levels on the output driver pins TX1 and TX0 depend on both the driver characteristic programmed by OCTPx, OCTNx (float, pull-up, pull-down, push-pull) and the output polarity programmed by OCPOLx (see Fig.4).

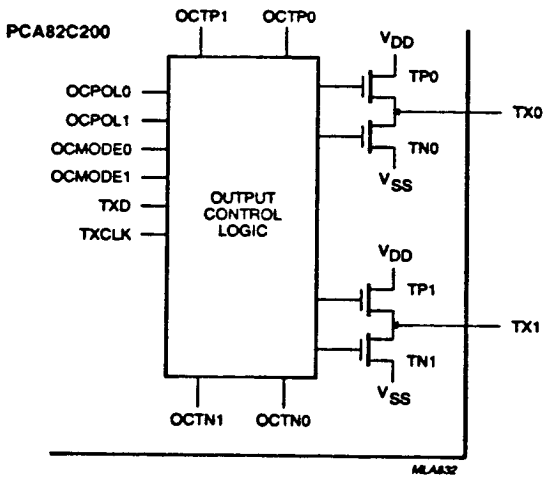


Fig.4 Configurable CAN transmitter.

Clock Output Mode

For the TX0 pin this is the same as in Normal Output Mode. However, the data stream to TX1 is replaced by the transmit clock (TXCLK). The rising edge of the transmit clock (non inverted) marks the beginning of a bit period. The clock pulse width is t_{sc}.

Bi-phase Output Mode

In contrast to Normal Output Mode the bit representation is time variant and toggled. If the bus controllers are galvanically decoupled from the bus-line by a transformer, the bit stream is not allowed to contain a DC component. This is achieved by the following scheme. During recessive bits all outputs are deactivated (3-state). Dominant bits are sent alternatingly on TX0 and TX1, i.e. the first dominant bit is sent on TX0, the second is sent on TX1, and the third one is sent on TX0 again, etc.

Test Output Mode

For the TX0 pin this is the same as in Normal Output Mode. To measure the delay time of the transmitter and receiver this mode connects the output of the input comparator (COMP OUT) with the input of the output driver TX1. This mode is used for production testing only.

The following two tables, Table 14 and Table 15, show the relationship between the bits of the Output Control Register and the two serial output pins TX0 and TX1 of the PCX82C200, connected to the serial bus (see Fig.1).

Table 14 Description of the Output Mode bits

OCMODE1	OCMODE0	DESCRIPTION
1	0	Normal Output Mode; TX0, TX1: bit sequence (TXD; note 1).
1	1	Clock Output Mode; TX0: bit sequence, TX1: bus clock (TXCLK).
0	0	Bi-phase Output Mode.
0	1	Test Output Mode; TX0: bit sequence, TX1: COMP OUT.

Note

1. TXD is the data bit to be transmitted.

Table 15 Output pin set-up

DRIVE	OCTPx	OCTNx	OCPOLx	TXD	TPx (note 1)	TNx (note 2)	TXx (note 3)
Float	0	0	0	0	OFF	OFF	float
	0	0	0	1	OFF	OFF	float
	0	0	1	0	OFF	OFF	float
	0	0	1	1	OFF	OFF	float
Pull-down	0	1	0	0	OFF	ON	LOW
	0	1	0	1	OFF	OFF	float
	0	1	1	0	OFF	OFF	float
	0	1	1	1	OFF	ON	LOW
Pull-up	1	0	0	0	OFF	OFF	float
	1	0	0	1	ON	OFF	HIGH
	1	0	1	0	ON	OFF	HIGH
	1	0	1	1	OFF	OFF	float
Push/Pull	1	1	0	0	OFF	ON	LOW
	1	1	0	1	ON	OFF	HIGH
	1	1	1	0	ON	OFF	HIGH
	1	1	1	1	OFF	ON	LOW

Notes

1. TPx is the on-chip output transistor x, connected to V_{DD}; x = 0 or 1.
2. TNx is the on-chip output transistor x, connected to V_{SS}; x = 0 or 1.
3. TXx is the serial output level on pin TX0 or TX1. It is required that the output level on the CAN-bus is dominant with TXD = 0 and recessive with TXD = 1 (see section 8.1.1).

6.2.10 Test Register (TR)

The Test Register is used for production testing only.

6.3 Transmit Buffer layout

The global layout of the Transmit Buffer is shown in Fig.3. This buffer serves to store a message from the microcontroller to be transmitted by the PCX82C200. It is subdivided into Descriptor and Data Field. The Transmit Buffer can be written to and read from by the microcontroller (see note 3 to Table 2).

6.3.1 Descriptor

Table 16 Descriptor Byte 1 (DSCR1)

DSCR1	ADDRESS 10						
7	6	5	4	3	2	1	0
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3

Table 17 Descriptor Byte 2 (DSCR2)

DSCR2	ADDRESS 11						
7	6	5	4	3	2	1	0
ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0

Identifier (ID)

The Identifier consists of 11 bits (ID.10 to ID.0). ID.10 is the most significant bit, which is transmitted first on the bus during the arbitration process. The Identifier acts as the message's name, used in a receiver for acceptance filtering and also determines the bus access priority during the arbitration process. The lower the binary value of the Identifier the higher the priority. This is due to the larger number of leading dominant bits during arbitration (see section 8.7 "Bus organization").

Remote Transmission Request bit (RTR)

Table 18 Description of the RTR bit

BIT	VALUE	COMMENTS
RTR	HIGH (remote)	Remote Frame will be transmitted by the PCX82C200.
	LOW (data)	Data Frame will be transmitted by the PCX82C200.

Data Length Code (DLC)

The number of bytes (Data Byte Count) in the Data Field of a message is coded by the Data Length Code. At the start of a Remote Frame transmission the Data Length Code is not considered due to the RTR bit being HIGH (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the Data Length Code must be specified correctly to avoid bus errors, if two CAN-controllers start a Remote Frame transmission simultaneously.

The range of the Data Byte Count is 0 to 8 bytes and coded as follows:

Data Byte Count = 8DLC.3 + 4DLC.2 + 2DLC.1 + DLC.0

For reasons of compatibility no Data Byte Counts other than 0 to 8 should be used.

6.3.2 Data Field

The number of transferred data bytes is determined by the Data Length Code. The first bit transmitted is the most significant bit of data byte 1 at address 12.

6.4 Receive Buffer layout

The layout of the Receive Buffer and the individual bytes correspond to the definitions given for the Transmit Buffer layout, except that the addresses start at 20 instead of 10 (see Fig.3).

6.5 Clock Divider Register (CDR)

The Clock Divider Register controls the CLK OUT frequency for the microcontroller (see Fig.1). It can be written to or read by the microcontroller. The default state of the register is divide by 12 for Motorola mode and divide by 2 for Intel mode. Values from 0 to 7 may be written into this register and will result in the CLK OUT frequencies shown in Table 20.

Table 19 Clock Divider Register bits

CDR	ADDRESS 31						
7	6	5	4	3	2	1	0
-	-	-	-	-	CD.2	CD.1	CD.0

Note

Bits CDR.7 to CDR.3 are reserved

Table 20 CLK OUT frequency selection

CD.2	CD.1	CD.0	CLK OUT FREQUENCY
0	0	0	$f_{osc}/2$
0	0	1	$f_{osc}/4$
1	1	0	$f_{osc}/6$
0	1	1	$f_{osc}/8$
1	0	0	$f_{osc}/10$
1	0	1	$f_{osc}/12$
1	1	0	$f_{osc}/14$
1	1	1	f_{osc}

Note

1. f_{OSC} is the frequency of the oscillator

7 BUS TIMING/SYNCHRONIZATION

The Bus Timing Logic (BTL) monitors the serial bus-line via the on-chip input comparator and performs the following functions (see section 5):

- monitors the serial bus-line level
- adjusts the sample point, within a bit period (programmable)
- samples the bus-line level using majority logic (programmable, 1 or 3 samples)
- synchronization to the bit stream:
  - hard synchronization at the start of a message
  - resynchronization during transfer of a message.

The configuration of the BTL is performed during the initialization of the PCX82C200. The BTL uses the following three registers:

- Control register (Sync)
- Bus Timing Register 0
- Bus Timing Register 1.

7.1 Bit timing

A bit period is built up from a number of system clock cycles (t_{SCL}), see section 6.2.7. One bit period is the result of the addition of the programmable segments TSEG1 and TSEG2 and the general segment SYNCSEG (see sections 6.2.7 to 6.2.8).

7.1.1 Synchronization Segment (SYNCSEG)

The incoming edge of a bit is expected during this state; this state corresponds to one system clock cycle (1 x t_{SCL}).

7.1.2 Time Segment 1 (TSEG1)

This segment determines the location of the sampling point within a bit period, which is at the end of TSEG1. TSEG1 is programmable from 1 to 16 system clock cycles (see section 6.2.8).

The correct location of the sample point is essential for the correct functioning of a transmission. The following points must be taken into consideration:

- a Start-Of-Frame (see section 8.2.1) causes all PCX82C200's to perform a 'hard synchronization' (see section 7.2.1) on the first recessive-to-dominant edge. During arbitration, however, several PCX82C200's may simultaneously transmit. Therefore it may require twice the sum of bus-line, input comparator and the output driver delay times until the bus is stable. This is the propagation delay time.
- to avoid sampling at an incorrect position, it is necessary to include an additional synchronization buffer on both sides of the sample point. The main reasons for incorrect sampling are:
  - incorrect synchronization due to spikes on the bus-line
  - slight variations in the oscillator frequency of each PCX82C200 in the network, which results in a phase error.

Time Segment 1 consists of the segment for compensation of propagation delays and the synchronization buffer directly before the sample point (see Fig.5).

7.1.3 Time Segment 2 (TSEG2)

This time segment provides:

- additional time at the sample point for calculation of the subsequent bit levels (e.g. arbitration)
- synchronization buffer segment directly after the sample point (see section 7.1.2).

TSEG2 is programmable from 1 to 8 system clock cycles (see section 6.2.8).

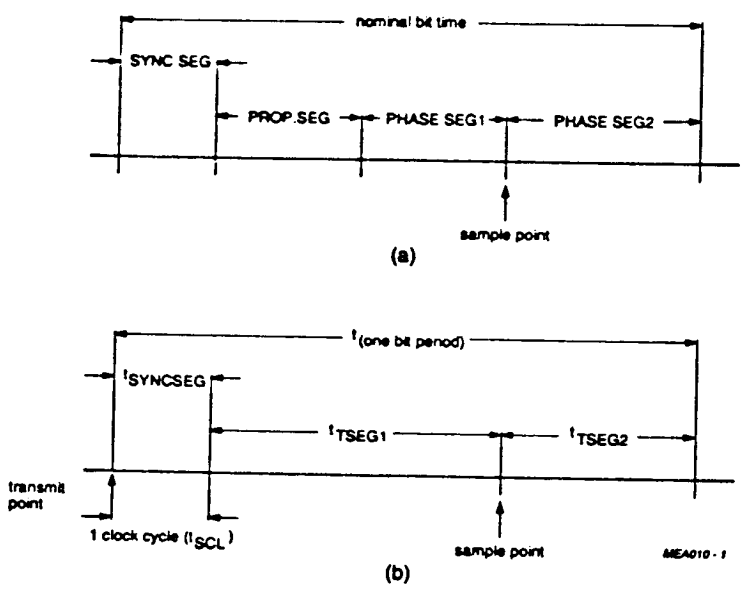


Fig.5 (a) Bit period as defined by the CAN-protocol. (b) Bit period as implemented in the PCX82C200.

7.1.4 Synchronization Jump Width (SJW)

SJW defines the maximum number of clock cycles (t_{SCL}) a bit period may be reduced or increased by one resynchronization. SJW is programmable from 1 to 4 system clock cycles (see section 6.2.7).

7.1.5 Propagation delay time

The propagation delay time (t_{prop}) is calculated by summing the maximum propagation delay times of the physical bus, the input comparator and the output driver. The resulting sum is multiplied by 2 and then rounded up to the nearest multiple of t_{SCL}.

t_{prop} = 2 x (physical bus delay + input comparator delay + output driver delay)

7.1.6 Bit timing restrictions

Restrictions on the configuration of the bit timing are based on internal processing. The restrictions are:

- $t_{TSEG2} \geq 2t_{SCL}$
- $t_{TSEG2} \geq t_{SJW}$
- $t_{TSEG1} \geq t_{TSEG2}$
- $t_{TSEG1} \geq t_{SJW} + t_{prop}$

The three sample mode (SAM = 1) has the effect of introducing a delay of one system clock cycle on the bus-line. This must be taken into account for the correct calculation of TSEG1 and TSEG2:

- $t_{TSEG1} \geq t_{SJW} + t_{prop} + 2t_{SCL}$
- $t_{TSEG2} \geq 3t_{SCL}$

7.2 Synchronization

Synchronization is performed by a state machine which compares the incoming edge with its actual bit timing and adapts the bit timing by hard synchronization or resynchronization.

7.2.1 Hard synchronization

This type of synchronization occurs only at the beginning of a message. The PCX82C200 synchronizes on the first incoming recessive-to-dominant edge of a message (being the leading edge of a message's Start-Of-Frame bit; see section 7.1).

7.2.2 Resynchronization

Resynchronization occurs during the transmission of a message's bit stream to compensate for:

- variations in individual PCX82C200 oscillator frequencies
- changes introduced by switching from one transmitter to another (e.g. during arbitration).

As a result of resynchronization either t_{TSEG1} may be increased by up to a maximum of t_{SJW} or t_{TSEG2} may be decreased by up to a maximum of t_{SJW}:

- $t_{TSEG1} \leq t_{SCL}$ ((TSEG1 + 1) + (SJW + 1))
- $t_{TSEG2} \geq t_{SCL}$ ((TSEG2 + 1) − (SJW + 1))

Note: TSEG1, TSEG2 and SJW are the programmed numerical values.

The phase error (e) of an edge is given by the position of the edge relative to SYNCSEG, measured in system clock cycles (t_{SCL}). The value of the phase error is defined as:

- e = 0, if the edge occurs within SYNCSEG
- e > 0, if the edge occurs within TSEG1
- e < 0, if the edge occurs within TSEG2.

The effect of resynchronization is:

- the same as that of a hard synchronization, if the magnitude of the phase error (e) is less or equal to the programmed value of t_{SJW} (see section 6.2.7)
- to increase a bit period by the amount of t_{SJW}, if the phase error is positive and the magnitude of the phase error is larger than t_{SJW}
- to decrease a bit period by the amount of t_{SJW}, if the phase error is negative and the magnitude of the phase error is larger than t_{SJW}.

7.2.3 Synchronization Rules

The synchronization rules are as follows:

- only one synchronization within one bit time is used
- an edge is used for synchronization only if the value detected at the previous sample point differs from the bus value immediately after the edge
- hard synchronization is performed whenever there is a recessive-to-dominant edge during Bus-Idle (see section 7)

- all other edges (recessive-to-dominant and optionally dominant-to-recessive edges if the Sync bit is set HIGH, see section 6.2.1) which are candidates for resynchronization will be used with the following exception:
  - a transmitting PCX82C200 will not perform a resynchronization as a result of a recessive-to-dominant edge with positive phase error, if only these edges are used for resynchronization. This ensures that the delay times of the output driver and input comparator do not cause a permanent increase in the bit time.

8 COMMUNICATION PROTOCOL

8.1 Frame types

The PCX82C200 bus controller supports the four different CAN-protocol frame types for communication:

- Data Frame, to transfer data
- Remote Frame, request for data
- Error Frame, globally signal a (locally) detected error condition
- Overload Frame, to extend delay time of subsequent frames (an Overload Frame is not initiated by the PCX82C200).

8.1.1 Bit representation

There are two logical bit representations used in the CAN-protocol:

- a recessive bit on the bus-line appears only if all connected PCX82C200's send a recessive bit at that moment
- dominant bits always overwrite recessive bits i.e. the resulting bit level on the bus-line is dominant.

8.2 Data Frame

A Data Frame carries data from a transmitting PCX82C200 to one or more receiving PCX82C200's. A Data Frame is composed of seven different bit-fields:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field (may have a length of zero)
- CRC Field
- Acknowledge Field
- End-Of-Frame.

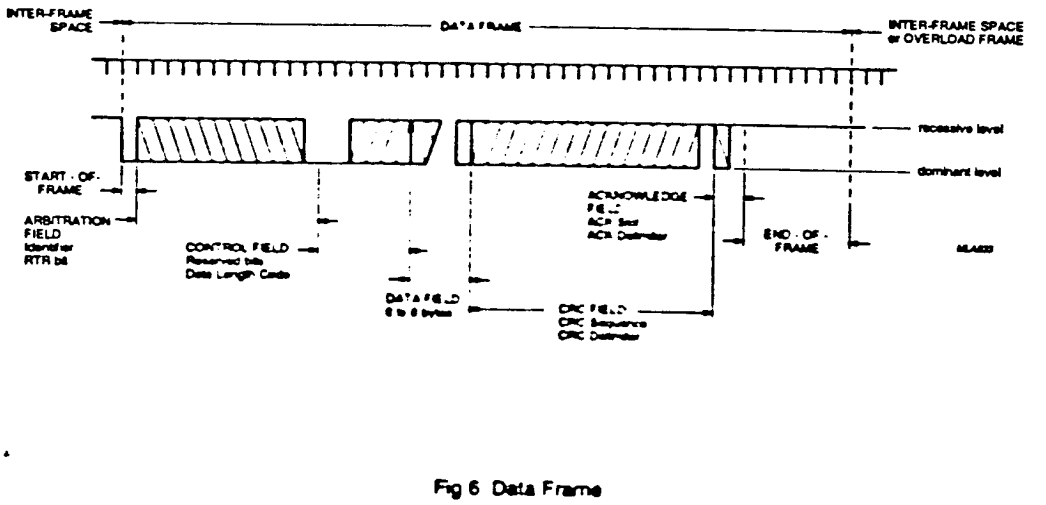


Fig 6 Data Frame

8.2.1 Start-Of-Frame bit

Signals the start of a Data Frame or Remote Frame. It consists of a single dominant bit used for hard synchronization of a PCX82C200 in receive mode.

8.2.2 Arbitration Field

Consists of the message identifier and the RTR bit (see section 6.3.1). In the event of simultaneous message transmissions by two or more PCX82C200's the bus access conflict is solved by bit-wise arbitration, which is active during the transmission of the Arbitration Field.

Identifier

This 11-bit field is used to provide information about the message, as well as the bus access priority. It is transmitted in the order ID.10 to ID.0 (LSB). The situation that the seven most significant bits (ID.10 to ID.4) are all recessive must not occur.

An Identifier does not define which particular PCX82C200 will receive the frame, because a CAN based communication network does not discriminate between a point-to-point, multicast or broadcast communication.

Remote Transmission Request bit (RTR)

A PCX82C200, acting as a receiver for certain information may initiate the transmission of the respective data by transmitting a Remote Frame to the network, addressing the data source via the Identifier and setting the RTR bit HIGH (remote; recessive bus level). If the data source simultaneously transmits a Data Frame containing the requested data, it uses the same Identifier. No bus access conflict occurs due to the RTR bit being set LOW (data; dominant bus level) in the Data Frame.

8.2.3 Control Field

This field consists of six bits. It includes two reserved bits (for future expansions of the CAN-protocol), transmitted with a dominant bus level, and is followed by the Data Length Code (4 bits). The number of bytes in the (destuffed; number of data bytes to be transmitted/received) Data Field is indicated by the Data Length Code. Admissible values of the Data Length Code and hence the number of bytes in the (destuffed) Data Field, are 0 to 8. A logic 0 (logic 1) in the Data Length Code is transmitted as a dominant (recessive) bus level, respectively.

8.2.4 Data Field

The data, stored within the Data Field of the Transmit Buffer, are transmitted according to the Data Length Code. Conversely, data of a received Data Frame will be stored in the Data Field of a Receive Buffer. Data is stored byte-wise both for transmission by the microcontroller and on reception by the PCX82C200. The most significant bit of the first data byte (lowest address) is transmitted/received first.

8.2.5 Cyclic Redundancy Code Field (CRC)

The CRC Field consists of the CRC Sequence (15 bits) and the CRC Delimiter (1 recessive bit). The Cyclic Redundancy Code (CRC) encloses the destuffed bit stream of the Start-Of-Frame, Arbitration Field, Control Field, Data Field and CRC Sequence. The most significant bit of the CRC Sequence is transmitted/received first. This frame check sequence, implemented in the PCX82C200, is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits, see section 8.8.3 With Start-Of-Frame (dominant bit) included in the code word, any rotation of the code word can be detected by the absence of the CRC Delimiter (recessive bit).

8.2.6 Acknowledge Field (ACK)

The Acknowledge Field consists of two bits, the Acknowledge Slot and the Acknowledge Delimiter, which are transmitted with a recessive level by the transmitter of the Data Frame. All PCX82C200's having received the matching CRC Sequence, report this by overwriting the transmitter's recessive bit in the Acknowledge Slot with a dominant bit (see section 8.9.2). Thereby a transmitter, still monitoring the bus level recognizes that at least one receiver within the network has received a complete and correct message (i.e. no error was found). The Acknowledge Delimiter (recessive bit) is the second bit of the Acknowledge Field. As a result, the Acknowledge Slot is surrounded by two recessive bits: the CRC Delimiter and the Acknowledge Delimiter.

All nodes within a CAN network may use all the information coming to the network by the PCX82C200's (shared memory concept). Therefore, acknowledgement and error handling are defined to provide all information in a consistent way throughout this shared memory. Hence, there is no reason to discriminate different receivers of a message in the acknowledge field. If a

node is disconnected from the network due to bus failure, this particular node is no longer part of the shared memory. To identify a 'lost node' additional and application specific precautions are required.

8.2.7 End-Of-Frame

Each Data Frame or Remote Frame is delimited by the End-Of-Frame bit sequence which consists of seven recessive bits (exceeds the bit stuff width by two bits). Using this method a receiver detects the end of a frame independent of a previous transmission error because the receiver expects all bits up to the end of the CRC sequence to be coded by the method of bit-stuffing (see section 8.7.3). The bit-stuffing logic is deactivated during the End-Of-Frame sequence.

8.3 Remote Frame

A PCX82C200, acting as a receiver for certain information may initiate the transmission of the respective data by transmitting a Remote Frame to the network, addressing the data source via the Identifier and setting the RTR bit HIGH (remote; recessive bus level). The Remote Frame is similar to the Data Frame with the following exceptions:

- RTR bit is set HIGH
- Data Length Code is ignored
- no Data Field contained.

Note that the Data Length Code value should be the same as for the corresponding Data Frame (although this is ignored for a Remote Frame).

A Remote Frame is composed of six different bit fields:

- Start-Of-Frame
- Arbitration Field
- Control Field
- CRC-Field
- Acknowledge Field
- End-Of-Frame.

See section 8.2 for a more detailed explanation of the Remote Frame bit fields.

8.4 Error Frame

The Error Frame consists of two different fields. The first field is accomplished by the superimposing of Error Flags contributed from different PCX82C200s. The second field is the Error Delimiter (see Fig.7).

8.4.1 Error Flag

There are two forms of an Error Flag:

- Active Error Flag, consists of six consecutive dominant bits
- Passive Error Flag, consists of six consecutive recessive bits unless it is overwritten by dominant bits from other PCX82C200's.

An error-active PCX82C200 (see section 8.9) detecting an error condition signals this by transmission of an Active Error Flag. This Error Flag's form violates the bit-stuffing law (see section 8.7.3) applied to all fields, from Start-Of-Frame to CRC Delimiter, or destroys the fixed form of the fields Acknowledge Field or End-Of-Frame (see Fig.6). Consequently, all other PCX82C200's detect an error condition and start transmission of an Error Flag. Therefore the sequence of dominant bits, which can be monitored on the bus, results from a superposition of different Error Flags transmitted by individual PCX82C200's. The total length of this sequence varies between six (minimum) and twelve (maximum) bits.

An error-passive PCX82C200 (see section 8.9) detecting an error condition tries to signal this by transmission of a Passive Error Flag. The error-passive PCX82C200 waits for six consecutive bits with identical polarity, beginning at the start of the Passive Error Flag. The Passive Error Flag is complete when these six identical bits have been detected.

8.4.2 Error Delimiter

The Error Delimiter consists of eight recessive bits and has the same format as the Overload Delimiter. After transmission of an Error Flag, each PCX82C200 monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every PCX82C200 has finished sending its Error Flag and all PCX82C200's start transmission of seven recessive bits (plus the recessive bit at dominant-to-recessive transition, results in a total of eight recessive bits). After this event and an Intermission Field all error-active PCX82C200's within the network can start a transmission simultaneously.

If a detected error is signalled during transmission of a Data Frame or Remote Frame, the current message is spoiled and a retransmission of the message is initiated.

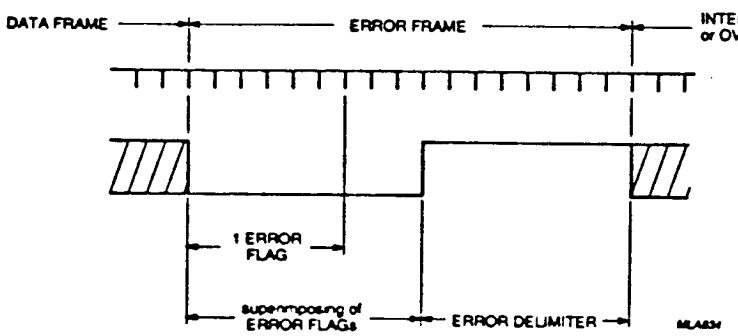


Fig.7 Error Frame.

If a PCX82C200 monitors any deviation of the Error Frame, a new Error Frame will be transmitted. Several consecutive Error Frame's may result in the PCX82C200 becoming error-passive and leaving the network unblocked.

In order to terminate an Error Flag correctly, an error-passive CAN-bus controller requires the bus to be Bus-Idle (see section 8.6.2) for at least three bit periods (if there is a local error at an error-passive receiver). Therefore a CAN-bus should not be 100% permanently loaded.

8.5 Overload Frame

The Overload Frame consists of two fields, the Overload Flag and the Overload Delimiter. There are two conditions in the CAN-protocol which lead to the transmission of an Overload Flag:

- condition 1; receiver circuitry requires more time to process the current data before receiving the next frame (receiver not ready)
- condition 2; detection of a dominant bit during Intermission Field (see section 8.6.1).

The transmission of an Overload Frame may only start:

- condition 1; during the first bit period of an expected Intermission Field
- condition 2; one bit period after detecting the dominant bit during Intermission Field.

The PCX82C200 will never initiate transmission of a condition 1 Overload Frame and will only react on a transmitted condition 2 Overload Frame, according to the CAN-protocol. No more than two Overload Frames are generated to delay a Data Frame or a Remote Frame. Although the overall form of the Overload Frame corresponds to that of the Error Frame, an Overload Frame does not initiate or require the retransmission of the preceding frame.

8.5.1 Overload Flag

The Overload Flag consists of six dominant bits and has a similar format to the Error Flag.

The Overload Flag's form corrupts the fixed form of the Intermission Field. All other PCX82C200's detecting the overload condition also transmit an Overload Flag (condition 2).

8.5.2 Overload Delimiter

The Overload Delimiter consists of eight recessive bits and takes the same form as the Error Delimiter. After transmission of an Overload Flag, each PCX82C200 monitors the bus-line until it detects a transition from a

dominant-to-recessive bit level. At this point in time, every PCX82C200 has finished sending its Overload Flag and all PCX82C200's start simultaneously transmitting seven more recessive bits.

8.6 Inter-Frame Space

Data Frames and Remote Frames are separated from preceding frames (all types) by an Inter-Frame Space, consisting of an Intermission Field and a Bus-Idle. Error-passive PCX82C200's also send a Suspend Transmission (see section 8.9.5) after transmission of a message. Overload Frames and Error Frames are not preceded by an Inter-Frame Space.

8.6.1 Intermission Field

The Intermission Field consists of three recessive bits. During an Intermission period, no frame transmissions will be started by any PCX82C200. An Intermission is required to have a fixed time period to allow a CAN-controller to execute internal processes prior to the next receive or transmit task.

8.6.2 Bus-Idle

The Bus-Idle time may be of arbitrary length (minimum 0 bit). The bus is recognized to be free and a CAN-controller having information to transmit may access the bus. The detection of a dominant bit level during Bus-Idle on the bus is interpreted as the Start-Of-Frame.

8.7 Bus organization

Bus organization is based on five basic rules described in the following paragraphs.

8.7.1 Bus access

PCX82C200's only start transmission during the Bus-Idle state. All PCX82C200's synchronize on the leading edge of the Start-Of-Frame (hard synchronization).

8.7.2 Arbitration

If two or more PCX82C200's simultaneously start transmitting, the bus access conflict is solved by a bit-wise arbitration process during transmission of the Arbitration Field.

During arbitration every transmitting PCX82C200 compares its transmitted bit level with the monitored bus level. Any PCX82C200 which transmits a recessive bit and monitors a dominant bus level immediately becomes the receiver of the higher priority message on the bus without corrupting any information on the bus. Each message contains an unique Identifier and a RTR bit describing the type of data within the message.

The Identifier together with the RTR bit implicitly define the message's bus access priority. During arbitration the most significant bit of the Identifier is transmitted first and the RTR bit last. The message with the lowest binary value of the Identifier and RTR bit has the highest priority. A Data Frame has higher priority than a Remote Frame due to its RTR bit having a dominant level.

For every Data Frame there is an unique transmitter. For reasons of compatibility with other CAN-bus controllers, use of the Identifier binary bit pattern ID = 1111111XXXX (X being bits of arbitrary level) is forbidden. The number of available different Identifiers is 2032 ($2^{11} - 2^4$).

8.7.3 Coding/decoding

The following bit fields are coded using the bit-stuffing technique:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence.

When a transmitting PCX82C200 detects five consecutive bits of identical polarity to be transmitted, a complementary (stuff) bit is inserted into the transmitted bit-stream.

When a receiving PCX82C200 has monitored five consecutive bits with identical polarity in the received bit streams of the above described bit fields, it automatically deletes the next received (stuff) bit. The level of the deleted stuff bit has to be the complement of the previous bits; otherwise a Stuff Error will be detected and signalled (see section 8.8.2).

The remaining bit fields or frames are of fixed form and are not coded or decoded by the method of bit-stuffing.

The bit-stream in a message is coded according to the Non-Return-to-Zero (NRZ) method, i.e. during a bit period, the bit level is held constant, either recessive or dominant.

8.7.4 Error signalling

A PCX82C200 which detects an error condition, transmits an Error Flag. Whenever a Bit Error, Stuff Error, Form Error or an Acknowledgement Error is detected, transmission of an Error Flag is started at the next bit. Whenever a CRC Error is detected, transmission of an Error Flag starts at the bit following the Acknowledge Delimiter, unless an Error Flag for another error condition has already started. An Error Flag violates the bit-stuffing law or corrupts the fixed form bit fields. A violation of the bit-stuffing law affects any PCX82C200 which detects the error condition. These devices will also transmit an Error Flag.

An error-passive PCX82C200 (see section 8.9) which detects an error condition, transmits a Passive Error Flag. A Passive Error Flag is not able to interrupt a current message at different PCX82C200's, but this type of Error Flag may be cancelled by other PCX82C200's. After having detected an error condition, an error-passive PCX82C200 will wait for six consecutive bits with identical polarity and when monitoring them, interpret them as an Error Flag.

After transmission of an Error Flag, each PCX82C200 monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every PCX82C200 has finished transmitting its Error Flag and all PCX82C200's start transmitting seven additional recessive bits (Error Delimiter, see section 8.4.2).

The message format of a Data Frame or Remote Frame is defined in such a way, that all detectable errors can be signalled within the message transmission time and therefore, it is very simple for a PCX82C200 to associate an Error Frame to the corresponding message and to initiate retransmission of the corrupted message.

If a PCX82C200 monitors any deviation of the fixed form of an Error Frame, it transmits a new Error Frame.

8.7.5 Overload signalling

Some CAN-controllers (but not the PCX82C200) require to delay the transmission of the next Data Frame or Remote Frame by transmitting one or more Overload Frames. The transmission of an Overload Frame must start during the first bit of an expected Intermission. Transmission of Overload Frames which are reactions on a dominant bit during an expected Intermission Field, start one bit after this event.

Though the format of Overload Frame and Error Frame are identical, they are treated differently. Transmission of an Overload Frame during Intermission Field does not initiate the retransmission of any previous Data Frame or Remote Frame.

If a CAN-controller which transmitted an Overload Frame monitors any deviation of its fixed form, it transmits an Error Frame.

8.8 Error detection

The processes described in the following paragraphs are implemented in the PCX82C200 for error detection.

8.8.1 Bit Error

A transmitting PCX82C200 monitors the bus on a bit-by-bit basis. If the bit level monitored is different from the transmitted one, a Bit Error is signalled. The exceptions being:

- during the Arbitration Field, a recessive bit can be overwritten by a dominant bit. In this case, the PCX82C200 interprets this as a loss of arbitration
- during the Acknowledge Slot, only the receiving PCX82C200's are able to recognize a Bit Error.

8.8.2 Stuff Error

The following bit fields are coded using the bit-stuffing technique:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence.

There are two possible ways of generating a Stuff Error:

- the disturbance generates more than the allowed five consecutive bits with identical polarity. These errors are detected by all PCX82C200's
- a disturbance falsifies one or more of the five bits preceding the stuff bit. This error situation is not recognized as a Stuff Error by the receivers. Therefore, other error detection processes may detect this error condition such as: CRC check, format violation at the receiving PCX82C200's or Bit Error detection by the transmitting PCX82C200.

8.8.3 CRC Error

To ensure the validity of a transmitted message all receivers perform a CRC check. Therefore, in addition to the (destuffed) information digits (Start-Of-Frame up to Data Field), every message includes some control digits (CRC Sequence; generated by the transmitting PCX82C200 of the respective message) used for error detection.

The code used for the PCX82C200 bus controller is a (shortened) BCH code, extended by a parity check and has the following attributes:

- 127 bits as maximum length of the code
- 112 bits as maximum number of information digits (maximum 83 bits are used by PCX82C200)
- length of the CRC Sequence amounts to 15 bits
- Hamming distance d = 6.

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

The CRC Sequence is calculated by the following procedure:

1. the destuffed bit stream consisting of Start-Of-Frame up to the Data Field (if present) is interpreted as a polynomial with coefficients of 0 or 1
2. this polynomial is divided (modulo-2) by the following generator polynomial:

$f(X) = (X^{14} + X^{9} + X^{8} + X^{6} + X^{5} + X^{4} + X^{2} + X + 1)(X + 1)$ = 1100010110011001 binary.

The remainder of this polynomial division is the CRC Sequence which includes a parity check. Burst errors are detected up to a length of 15 [degree of f(X)]. Multiple errors (number of disturbed bits at least d = 6) are not detected with a residual error probability of 2^{-15} (= 3×10^{-5}) by CRC check only.

8.8.4 Form Error

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

- End-Of-Frame
- Intermission
- Acknowledge Delimiter
- CRC Delimiter.

During the transmission of these bit fields an error condition is recognized if a dominant bit level instead of a recessive one is detected.

8.8.5 Acknowledgement Error

This is detected by a transmitter whenever it does not monitor a dominant bit during the Acknowledge Slot.

8.8.6 Error detection by an Error Flag of another PCX82C200

The detection of an error is signalled by transmitting an Error Flag. An Active Error Flag causes a Stuff Error, a Bit Error or a Form Error at all other PCX82C200's.

8.8.7 Error detection capabilities

Errors which occur at all PCX82C200's (global errors) are 100% detected. For local errors, i.e. for errors occurring at some PCX82C200's only, the shortened BCH code, extended by a parity check, has the following error detection capabilities:

- up to five single bit errors are 100% detected, even if they are distributed randomly within the code
- all single bit errors are detected if their total number (within the code) is odd
- the residual error probability of the CRC check amounts to 3×10^{-5}. As an error may be detected not only by CRC check but also by other detection processes described in sections 8.8.1 to 8.8.5, the residual error probability is several magnitudes less than 3×10^{-6} for undetected errors.

8.9 Error confinement (definitions)

8.9.1 Bus-Off

A PCX82C200 which has too many unsuccessful transmissions, relative to the number of successful transmissions, will enter the Bus-Off state. It remains in this state, neither receiving nor transmitting messages until the Reset Request bit is set LOW (absent) and both Error Counters are set to '0' (see note 1 to Table 5 and section 8.10.3).

8.9.2 Acknowledge (ACK)

A PCX82C200 which has received a valid message correctly, indicates this to the transmitter by transmitting a dominant bit level on the bus during the Acknowledge Slot, independent of accepting or rejecting the message.

8.9.3 Error-Active

An error-active PCX82C200 is in its normal operating state able to receive and to transmit normally and also to transmit an active Error Flag (see section 8.10.3).

8.9.4 Error-Passive

An error-passive PCX82C200 may transmit or receive messages normally. In the case of a detected error condition it transmits a Passive Error Flag, instead of an Active Error Flag. Hence the influence on bus activities by an error-passive PCX82C200 (e g. due to a malfunction) is reduced.

8.9.5 Suspend Transmission

After an error-passive PCX82C200 has transmitted a message, it sends eight recessive bits after the Intermission Field and then checks for Bus-Idle. If during Suspend Transmission another PCX82C200 starts transmitting a message the suspended PCX82C200 will become the receiver of this message; otherwise being in Bus-Idle it may start to transmit a further message.

8.9.6 Start-up

A PCX82C200 which was either switched off or is in the Bus-Off state, must run a start-up routine in order to:

- synchronize with other available PCX82C200's, before starting to transmit. Synchronizing is achieved, when 11 recessive bits, equivalent to Acknowledge Delimiter, End-Of-Frame and Intermission Field, have been detected (Bus-Free)
- wait for other PCX82C200s without passing into the Bus-Off state (due to a missing acknowledge), if there is no other PCX82C200 currently available.

8.10 Aims of error confinement

8.10.1 Distinction of short and long-lasting disturbances

The microcontroller must be informed when there are long-lasting disturbances and when bus activities have returned to normal operation. During long-lasting disturbances, a PCX82C200 enters the Bus-Off state and the microcontroller may use default values.

Minor disturbances of bus activities will not affect a PCX82C200. In particular, a PCX82C200 does not enter the Bus-Off state or inform the microcontroller of a short-lasting bus disturbance.

8.10.2 Detection and localization of hardware disturbances and defects

The rules for error confinement are defined by the CAN-protocol specification (and implemented in the PCX82C200), in that the PCX82C200, being nearest to the error-locus, reacts with a high probability, the quickest (i.e. becomes error-passive or Bus-Off), hence errors can be localized and their influence on normal bus activities is minimized.

8.10.3 Error Confinement

All PCX82C200's contain a Transmit Error Counter and a Receive Error Counter, which registers errors during the transmission and the reception of messages, respectively.

If a message is transmitted or received correctly, the count is decreased. In the event of an error, the count is increased. The Error Counters have an non-proportional method of counting: an error causes a larger counter increase than a correctly transmitted/received message causes the count to decrease. Over a period of time this may result in an increase in error counts, even if there are fewer corrupted messages than uncorrupted ones. The level of the Error Counters reflect the relative frequency of disturbances. The ratio of increase/decrease depends on the acceptable ratio of invalid/valid messages on the bus and is hardware implemented to eight.

If one of the Error Counters exceeds the Warning Limit of 96 error points, indicating a significant accumulation of error conditions, this is signalled by the PCX82C200 (Error Status, Error Interrupt).

A PCX82C200 operates in the error-active mode until it exceeds 127 error points on one of its Error Counters. At this point it will enter the error-passive state.

A transmit error which exceeds 255 error points results in the PCX82C200 entering the Bus-Off state.

9 LIMITING VALUES

Limiting values in accordance with the Absolute Maximum Rating System (IEC134)

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage range	4.5	5.5	V
I_I	input/output current on any pin except from TX0 and TX1	–	±10	mA
I_{OT}	sink current of TX0 and TX1 together (note 1)	–	28	mA
I_{OT}	source current of TX0 and TX1 together (note 1)	–	–20	mA
T_{amb}	operating ambient temperature range:			
	PCA82C200	–40	+125	°C
	PCF82C200	–40	+85	°C
T_{stg}	storage temperature range	–65	+150	°C
P_{tot}	total power dissipation (note 2)	–	1	W

Notes

1. I_{OT} is allowed in case of a bus failure condition because then the TX-outputs are switched off automatically after a short time (Bus-Off state). During normal operation I_{OT} is a peak current, permitted for $t < 100$ ms. The average output current must not exceed 10 mA for each TX-output.
2. The value is based on the maximum allowable die temperature and the thermal resistance of the package, not on device power consumption.

10 DC CHARACTERISTICS

$V_{DD} \leq 5$ V ±10%; $V_{SS} = 0$ V; $T_{amb} = -40$ to +125 °C for the PCA82C200 and $T_{amb} = -40$ to +85 °C for the PCF82C200. All voltages measured with respect to V_{SS} unless otherwise specified

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
Supply					
V_{DD}	supply voltage range		4.5	5.5	V
I_{DD}	supply current: operating	$\overline{RST} = V_{SS}$; f_{CLK} = 16 MHz (note 1)	–	15	mA
I_{sm}	sleep mode	oscillator inactive (note 2)	–	40	µA
Inputs					
V_{IL1}	LOW level input voltage (except XTAL1, RX0 and RX1)		–0.5	0.8	V
V_{IL2}	XTAL1 LOW level input voltage		–	$0.2V_{DD}$	V
V_{IH1}	HIGH level input voltage (except XTAL1, $\overline{RST}$, RX0 and RX1)		3.2	$V_{DD} + 0.5$	V
V_{IH2}	XTAL1 HIGH level input voltage		$0.7V_{DD}$	–	V
V_{IH3}	$\overline{RST}$ HIGH level input voltage		3.3	$V_{DD} + 0.5$	V
I_{LI}	input leakage current (except XTAL1, RX0 and RX1)	$0.45 V < V_I < V_{DD}$	–	±10	µA

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
Outputs					
V_{OL}	LOW level output voltage (except XTAL2, TX0 and TX1)	I_{OL} = 1.6 mA	–	0.45	V
V_{OH1}	HIGH level output voltage (except TX0, TX1, $\overline{INT}$ and CLK OUT)	I_{OH} = –80 µA	2.4	–	V
V_{OH2}	CLK OUT HIGH level output voltage	I_{OH} = –80 µA	0.8V_{DD}	–	V
CAN input comparator					
		V_{DD} = 5 V ±5%; 1.4 V < V_I < V_{DD} – 1.4 V			
V_{DIF}	differential input voltage	note 3	±42	–	mV
V_{HYST}	hysteresis voltage	note 3	12	45	mV
I_I	input current		–	±400	nA
CAN output driver					
		V_{DD} = 5 V ±5%			
V_{OLT}	TX0 and TX1 output voltage LOW	I_O = 1.2 mA (note 3)	–	0.1	V
		I_O = 10 mA	–	1.0	V
V_{OHT}	TX0 and TX1 output voltage HIGH	I_O = 1.2 mA (note 3)	V_{DD} – 0.1	–	V
		I_O = 10 mA	V_{DD} – 1.0	–	V

Notes

1. (AD0 – AD7) = ALE = $\overline{RD}$ = $\overline{WR}$ = $\overline{CS}$ = V_{DD}; MODE = V_{SS}; RX0 = 2.7 V; RX1 = 2.3 V; XTAL1 = 0.5V/V_{DD} – 0.5 V; all outputs unloaded.
2. (AD0 – AD7) = ALE = $\overline{RD}$ = $\overline{WR}$ = $\overline{INT}$ = $\overline{RST}$ = $\overline{CS}$ = MODE = RX0 = V_{DD}; RX1 = XTAL1 = V_{SS}; all outputs unloaded.
3. Not tested during production.

11 AC CHARACTERISTICS

V_{DD} = 5 V ±10%; V_{SS} = 0 V; C_L = 50 pF (output pins); T_{amb} = −40 to +85/125 °C; unless otherwise specified (note 1)

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
f_{CLK}	oscillator frequency		3	16	MHz
t_{SU1}	address set-up to ALE/AS LOW		10	–	ns
t_{HD1}	address hold time		22	–	ns
t_{PW1}	ALE/AS pulse width		35	–	ns
t_{VD1}	$\overline{RD}$ LOW to valid data output	Intel mode	–	60	ns
t_{VD2}	E HIGH to valid data output	Motorola mode	–	60	ns
t_{DF1}	data float after $\overline{RD}$ HIGH	Intel mode	10	55	ns
t_{DF2}	data float after E LOW	Motorola mode	10	55	ns
t_{SU2}	input data set-up to $\overline{WR}$ HIGH	Intel mode	30	–	ns
t_{HD2}	input data hold after $\overline{WR}$ HIGH	Intel mode	13	–	ns
t_{HH1}	$\overline{WR}$ HIGH to next ALE HIGH		23	–	ns
t_{LH3}	E LOW to next AS HIGH	Motorola mode	23	–	ns
t_{SU3}	input data set-up to E LOW	Motorola mode	30	–	ns
t_{HD3}	input data hold after E LOW	Motorola mode	25	–	ns
t_{LL1}	ALE LOW to $\overline{WR}$ LOW	Intel mode	10	–	ns
t_{LL2}	ALE LOW to $\overline{RD}$ LOW	Intel mode	10	–	ns
t_{LH1}	AS LOW to E HIGH	Motorola mode	10	–	ns
t_{SU4}	set-up time of RD/$\overline{WR}$ to E HIGH	Motorola mode	20	–	ns
t_{PW2}	$\overline{WR}$ pulse width	Intel mode	170	–	ns
t_{PW3}	$\overline{RD}$ pulse width	Intel mode	170	–	ns
t_{PW4}	E pulse width	Motorola mode	170	–	ns
t_{LL3}	$\overline{CS}$ LOW to $\overline{WR}$ LOW	Intel mode	0	–	ns
t_{LL4}	$\overline{CS}$ LOW to $\overline{RD}$ LOW	Intel mode	0	–	ns
t_{LH2}	$\overline{CS}$ LOW to E HIGH	Motorola mode	0	–	ns
Input comparator/output driver					
t_{sd}	sum of the input and output delays	V_{DD} = 5 V ±5%; V_{DIF} = ±42 mV; 1.4 V < V_I < V_{DD} − 1.4 mV	–	62	ns

Note

1. AC characteristics are not tested.

11.1 AC timing diagrams

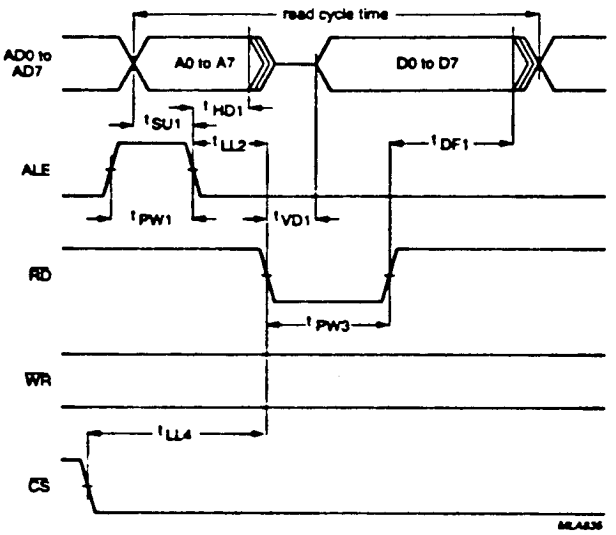


Fig.8 Read cycle timing diagram; Intel mode.

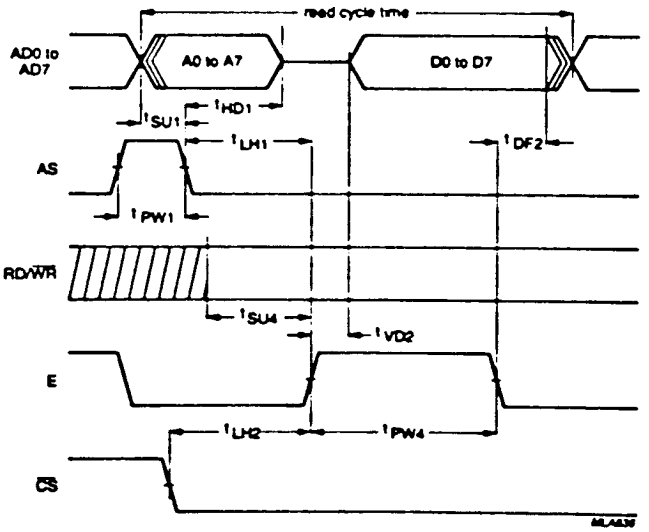


Fig.9 Read cycle timing diagram; Motorola mode.

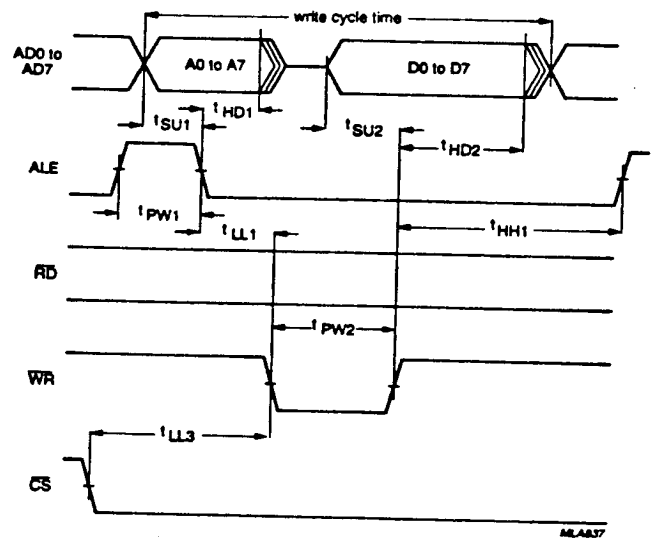


Fig.10 Write cycle timing diagram; Intel mode.

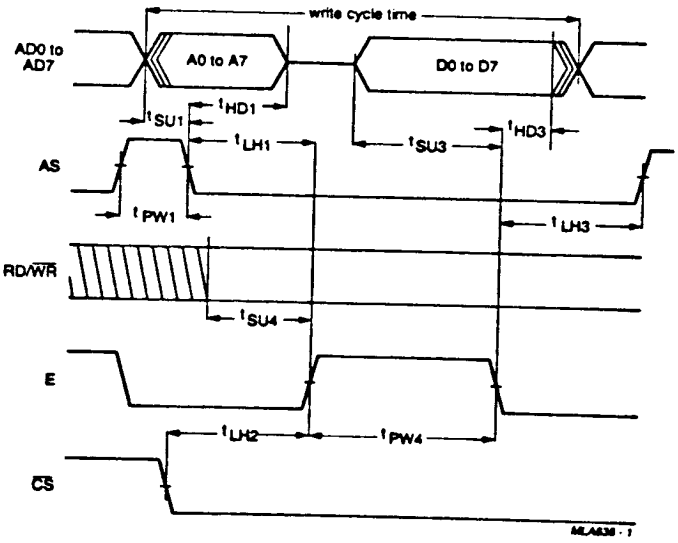


Fig.11 Write cycle timing diagram; Motorola mode.

11.2 Additional AC Information

To provide optimum noise immunity under worse case conditions, the chip is powered by three separate pins and grounded by three separate pins, see Fig.12.

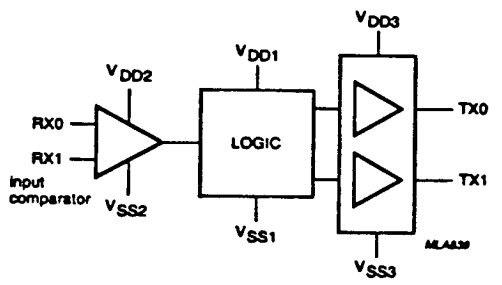


Fig.12 Optimized noise immunity block diagram.

12 DEVELOPMENT SUPPORT AND TOOLS

12.1 The PCX82C200 Evaluation Board

Philips offers powerful support during the design and test stages of CAN networks, working closely with customers to develop their systems. The 'Philips Stand-alone CAN-Controller (PSCC) Evaluation Board' is a versatile tool being a ready-to-use hardware and software module, very similar to a real CAN module. Since a 5 V power supply is provided, the board can be used in any vehicle without modification. An RS232 interface allows a terminal or a PC with terminal-emulation software to be connected to the board. The board comprises:

- a PCX82C200 CAN-bus controller
- a PCA80C552 microcontroller with up to 32K x 8 bits external RAM and EPROM
- a 5 V power supply with protection against car battery disturbances
- two different physical CAN-bus interfaces (selectable)
- an RS232 interface
- demonstration hardware
- a wrap field for customer-specific circuitry.

The software provided with the board supports "learning about CAN" and assists in prototype (e.g. in-vehicle) networks. It provides:

- demonstration software (automatically-initiated)
- the menu-driven software comprises:
  - a facility to alter the contents of the PCX82C200 registers
  - a bus monitor to receive messages from the CAN-bus and to display them on a terminal
  - a download facility for the user's application software.

With these facilities the board is a basis for prototype modules; when using entirely your own software, the board can be used as a custom, debugged and proven hardware module.

12.2 Advanced support

For further development support, Philips subcontractor I+ME offers a complete set of development tools including:

- a CAN simulator; CAN/Net Sim
- an emulator; CAN/Net Emu
- a network analyzer; CAN/Net Anal.

I+ME can be contacted through the following address:

I+ME GmbH
Ferdinandstrasse 15 A
D-3340 Wolfenbuettel
West Germany.

Phone: ++49-5331-72066
Fax: ++49-5331-32455