Single-chip 8-bit microcontroller with CAN controller

80C592/83C592/87C592

DESCRIPTION

The 80C592/83C592/87C592 (hereafter referred to generically as the 8XC592) is a stand-alone high-performance microcontroller designed for use in automotive and general industrial applications. In addition to the 80C51 standard features, this device provides a number of dedicated hardware functions for these applications. Three versions of this derivative will be offered:

- 83C592 (ROM version)
- 80C592 (ROMless version)
- 87C592 (EPROM/OTP version)

It combines the functions of the existing 8XC552 and the Philips CAN-Controller PCA82C200 (CAN: Controller Area Network) with the following enhanced features:

- 16K byte Program Memory
- 2 x 256 byte Data Memory
- DMA between CAN Transmit/Receive buffer and internal RAM

The temperature range includes –40°C to +85°C as well as automotive temperature range –40°C to +125°C for the ROM and ROMless version with a maximum clock frequency of 16MHz. The 87C592 has a temperature range of –40°C to +85°C.

The main differences to the 8XC552 microcontroller are:

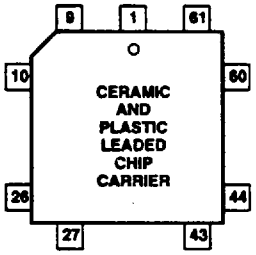
- a CAN-controller substitutes the I^2C-serial interface
- 16K byte programmable ROM resp. EPROM instead of 8K byte
- additional 256 byte RAM.

The 8XC592 contains a 16k x 8 EPROM (87C592), ROM (83C592) program memory, a volatile 512 x 8 read/write data memory, a Controller Area Network (CAN) controller, six 8-bit I/O ports, one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, a 10-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and CAN), a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC592 memory can be expanded externally using standard TTL compatible memories and logic.

FEATURES

- 80C51 core architecture
- 16k X 8 EPROM (87C592)
- 16k X 8 ROM (83C592)
- ROMless (80C592)
- 512 X 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- A 10-bit ADC with eight multiplexed analog inputs
- Two 8-bit resolution, pulse width modulation outputs
- 15 interrupt sources with 2 priority levels
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- CAN controller with DMA transfer between internal data RAM and CAN registers
- Up to 1 Mbit/s CAN-Controller with bus failure management facility
- $V_{DD}/2$ reference voltage
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Extended temperature ranges (–40 to +125°C)
- OTP package available
- ROM code protection

PIN CONFIGURATION



Pin	Function	Pin	Function
1	P5.0/ADC0	35	V_{SS}
2	V_{DD}	36	P2.0/A08
3	STADC	37	P2.1/A09
4	$\overline{PWM0}$	38	P2.2/A10
5	$\overline{PWM1}$	39	P2.3/A11
6	$\overline{EW}$	40	P2.4/A12
7	P4.0/CMSR0	41	P2.5/A13
8	P4.1/CMRS1	42	P2.6/A14
9	P4.2CMSR2	43	P2.7/A15
10	P4.3/CMSR3	44	$\overline{PSEN}$
11	P4.4/CMSR4	45	ALE/$\overline{PROG}$
12	P4.5/CMSR5	46	$\overline{EA}/V_{PP}$
13	P4.6/CMT0	47	P0.7/AD7
14	P4.7/CMT1	48	P0.6/AD6
15	RST	49	P0.5/AD5
16	P1.0/CT0I/INT2	50	P0.4/AD4
17	P1.1/CT1I/INT3	51	P0.3/AD3
18	P1.2/CT2I/INT4	52	P0.2/AD2
19	P1.3/CT3I/INT5	53	P0.1/AD1
20	P1.4/T2	54	P0.0/AD0
21	P1.5/RT2	55	REF
22	CV_{SS}	56	CRX1
23	P1.6/CTX0	57	CRX0
24	P1.7/CTX1	58	AVref–
25	P3.0/RxD	59	AVref+
26	P3.1TxD	60	AV_{SS}
27	P3.2/$\overline{INT0}$	61	AV_{DD}
28	P3.3/$\overline{INT1}$	62	P5.7/ADC7
29	P3.4/T0	63	P5.6/ADC6
30	P3.5/T1	64	P5.5/ADC5
31	P3.6/$\overline{WR}$	65	P5.4/ADC4
32	P3.7/$\overline{RD}$	66	P5.3/ADC3
33	XTAL2	67	P5.2/ADC2
34	XTAL1	68	P5.1/ADC1

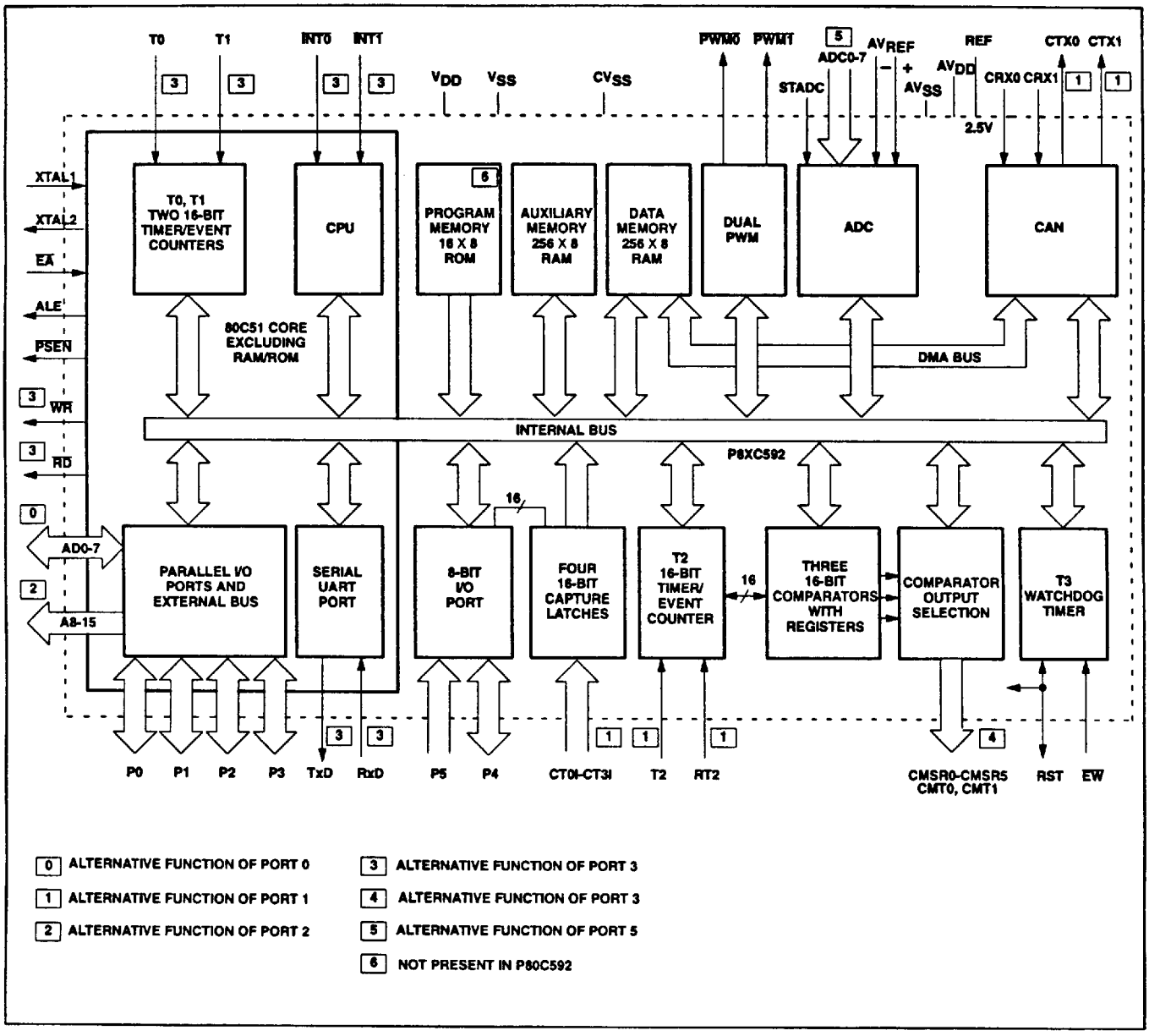
ORDERING INFORMATION

ROMless	ROM	EPROM	TEMPERATURE RANGE °C AND PACKAGE	FREQUENCY MHZ	DRAWING NUMBER
P80C592FFA	P83C592FFA	–	–40 to +85, 68-Pin Plastic Leaded Chip Carrier	1.2 to 16	0398
P80C592FHA	P83C592FHA	–	–40 to +125, 68-Pin Plastic Leaded Chip Carrier	1.2 to 16	0398
–	–	P87C592EFL	–40 to +85, 68-Pin Ceramic Leaded Chip Carrier w/Window	3.5 to 16	1240
–	–	P87C592EFA	–40 to +85, 68-Pin Plastic Leaded Chip Carrier	3.5 to 16	0398

Single-chip 8-bit microcontroller with CAN controller

80C592/83C592/87C592

BLOCK DIAGRAM



Single-chip 8-bit microcontroller with CAN controller

80C592/83C592/87C592

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{DD}	2	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software). This pin must not float.
$\overline{PWM0}$	4	O	**Pulse Width Modulation:** Output 0.
$\overline{PWM1}$	5	O	**Pulse Width Modulation:** Output 1.
$\overline{EW}$	6	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode. This pin must not float.
P0.0-P0.7	54-47	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s. Port 0 is also used to input the code byte during programming and to output the code byte during verification.
P1.0-P1.7	16-21, 23–24	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	16-19	I	**CT0I-CT3I (P1.0-P1.3):** Capture timer input signals for timer T2.
	16-19	I	**INT2-INT5 (P1.0-P1.3):** External interrupts 2-5.
	20	I	**T2 (P1.4):** T2 event input. Rising edge triggered.
	21	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
	23	O	**CTX0 (P1.6):** CAN transmitter output 0.
	24	O	**CTX1 (P1.7):** CAN transmitter output 1. Port 1 is also used to input the lower order address byte during EPROM programming and verification. A0 is on P1.0, etc.
CV_{SS}	22	I	**CV_{SS}:** CAN transmitter driver ground.
P2.0-P2.7	36-43	I/O	**Port 2:** 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08-A15). Port 2 is also used to input the upper order address during EPROM programming and verification. A8 is on P2.0, A9 on P2.1, through A12 on P2.4.
P3.0-P3.7	25-32	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	25		**RxD (P3.0):** Serial input port.
	26		**TxD (P3.1):** Serial output port.
	27		**$\overline{INT0}$ (P3.2):** External interrupt.
	28		**$\overline{INT1}$ (P3.3):** External interrupt.
	29		**T0 (P3.4):** Timer 0 external input.
	30		**T1 (P3.5):** Timer 1 external input.
	31		**$\overline{WR}$ (P3.6):** External data memory write strobe.
	32		**$\overline{RD}$ (P3.7):** External data memory read strobe.
P4.0-P4.7	7-14	I/O	**Port 4:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	7-12	O	**CMSR0-CMSR5 (P4.0-P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0-P5.7	68-62, 1	I	**Port 5:** 8-bit input port. **ADC0-ADC7 (P5.0-P5.7):** Alternate function: Eight input channels to ADC.
RST	15	I/O	**Reset:** Input to reset the 8XC592. It also provides a reset pulse as output when the watchdog timer overflows or after a CAN wakeup from power-down.
XTAL1	34	I	**Crystal Pin 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	33	O	**Crystal Pin 2:** Output of the inverting amplifier that forms the oscillator. Left open-circuit when an external clock is used.
V_{SS}	35	I	**Digital ground.**
$\overline{PSEN}$	44	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE/$\overline{PROG}$	45	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
$\overline{EA}/V_{PP}$	46	I	**External Access:** When $\overline{EA}$ is held at TTL level high, during reset the CPU executes out of the internal program ROM provided the program counter is less than 16384. When $\overline{EA}$ is held at TTL low level, during reset the CPU executes out of external program memory. $\overline{EA}$ is not allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
REF	55	O	**REF:** $AV_{DD}/2$ reference voltage output or input, depending on CAN control register bits. If the internal reference is used, then REF should be connected to AV_{SS} through a 10nf (or greater) capacitor.
CRX1	56	I	**CRX1:** CAN receiver input line 1.
CRX0	57	I	**CRX0:** CAN receiver input line 0.
AV_{REF-}	58	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	I	**Analog Ground** (for ADC and CAN receiver)
AV_{DD}	61	I	**Analog Power Supply** (for ADC and CAN receiver)

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} + 0.5V or V_{SS} – 0.5V, respectively.

The 8XC592 has the same operation as the 8XC552 for all features except the CAN interface and the AuxRAM. Please refer to the 8XC552 section in this data handbook for information on the PWM outputs, A/D converter, Timers 0, 1, or 2, the Watchdog Timer and the UART (SIO0).

INTERNAL DATA MEMORY

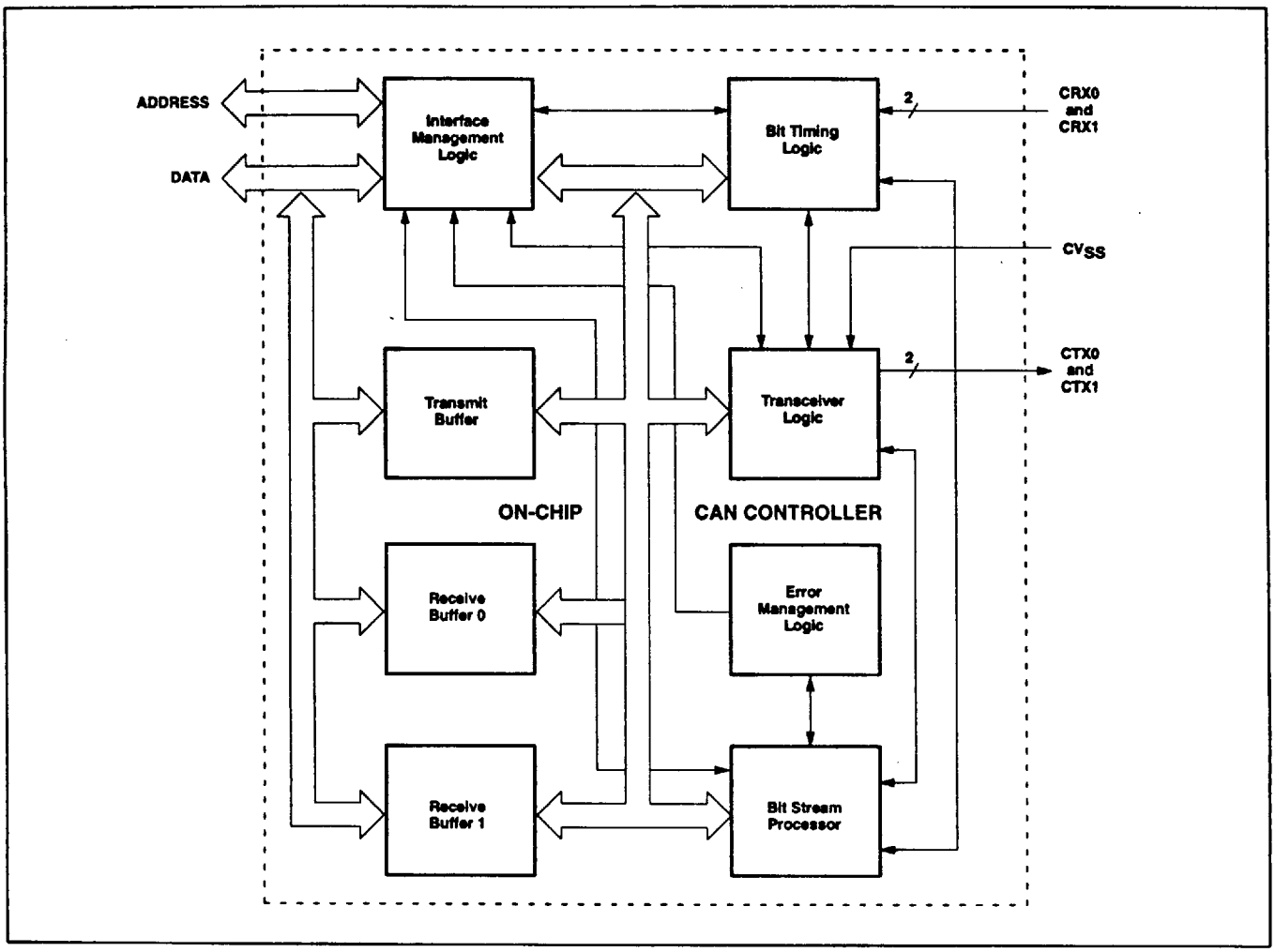
The internal Data Memory is divided into three physically separated parts: the 256 byte of Main RAM, the 256 byte of AuxRAM, and the 128 byte special function area. these can be addressed each in a different way.

- Main RAM 0 to 127 can be addressed directly and indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- Main RAM 128 to 255 can only be addressed indirectly. Address pointers are R0 and R1 of the selected register bank.
- AuxRAM 0 to 255 is indirectly addressable in the same way as the external Data Memory with MOVX instructions. Address pointers are R0, R1 of the selected register bank and the DPTR. An access to AuxRAM 0 to 255 will not affect the ports P0, P2, P3.6 and P3.7 during internal program execution.

An access to external Data Memory locations higher than 255 will be performed with the MOVX @DPTR instructions in the same way as in the 80C51 structure, so with P0 and P2 as data/address bus and P3.6 and P.37 as write and read strobe signals. Note that these external Data Memory locations cannot be accessed with R0 or R1 as address pointer.

Single-chip 8-bit microcontroller with CAN controller

BLOCK DIAGRAM OF 8XC592 CAN CONTROLLER CIRCUITRY



CAN FUNCTIONAL DESCRIPTION

SIO1, CAN (Controller Area Network)

CAN is the definition of a high performance communication protocol for serial data communication. The 8XC592 on-chip CAN Controller is a full implementation of the CAN-protocol. With the 8XC592, powerful local networks can be built, both for automotive and general industrial environments. This results in a strongly reduced wiring harness and enhanced diagnostic and supervisory capabilities.

Features

- Multi-master architecture
- Bus access priority determined by the message identifier
- 2032 message identifier
- Guaranteed latency time for high priority messages
- Powerful error handling capability
- Data length from 0 up to 8 bytes
- 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's network classes A, B and C
- DMA providing high-speed on-chip data exchange
- Bus failure management facility
- $AV_{DD/2}$ reference voltage

The CAN Controller 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.

NOTE:
* The latency time defines the period between the initiation (Transmission Request) and the start of the transmission on the bus. The latency time strongly depends on a large 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 the application note on bit timing.

CAN Functional Overview

The 8XC592 includes all hardware modules necessary to implement the transfer layer which represents the kernel of the CAN protocol. Refer to the block diagram (previous page) of the CAN controller portion of the 8XC592.

Interface Special Function Registers

The data transfer between the CPU and the CAN part of the 8XC592 is done via four SFRs:

CANADR (DBH):	to point to a register of the CAN–controller
CANDAT (DAH):	to read or write data
CANCON (D9H):	to read interrupt flags and to write commands
CANSTA (D8H):	to read status information and to write DMA pointer to the MAIN RAM

Additionally, the DMA–logic allows a high–speed data exchange between the CAN–controller and the MAIN RAM (see section "Handling of the CPU–CAN interface").

Interface Management Logic (IML)

The IML interprets the commands from the CPU, controls the allocation of the message buffers Transmit Buffer (TBF), Receive Buffer 0 (RBF0), and Receive Buffer 1 (RBF1), and provides interrupts and status information to the CPU.

Transmit Buffer (TBF)

The TBF is an interface between the CPU and the Bit Stream Processor (BSP) and is able to store a complete message. The buffer is written by the CPU and read by the BSP. The TBF is 10 bytes long to hold the Descriptor (2 bytes) and the Data-Field (up to 8 bytes) of the message.

Receive Buffer (RBF0, RBF1)

The RBF is an interface between the BSP and the CPU and stores a message received from the busline. Once filled by BSP and allocated to the CPU by IML, the buffer cannot be used to store subsequently received messages until the CPU has (read and) released the buffer.

To reduce the requirements on the CPU, two receive buffers (RBF0, RBF1) are implemented. While one RBF is allocated to the CPU, the BSP may write to the other one. Both RBF0 and RBF1 are 10 bytes long to hold the Descriptor (2 bytes) and the Data-Field (up to 8 bytes) of the message.

Bit Stream Processor (BSP)

This is a sequencer controlling the data stream between transmit and receive buffers (parallel data) and the busline (serial data). The BSP contains the Acceptance Filter and also controls the TCL and the EML such that the processes of reception, arbitration, transmission, and error signaling are performed according to the protocol. The BSP provides signals to the IML indicating when a message has got acceptance, when a receive buffer contains a valid message, and also when the transmit butter is no longer required after a successful transmission.

Bit Timing Logic (BTL)

This block monitors the busline using the (built-in) Input Comparator and handles the busline-related bit timing.

The BTL synchronizes on a "recessive" to "dominant" busline transition at the beginning of a message (hard synchronization) and resynchronizes on further transitions during the reception of a message (soft synchronization).

The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g., due to oscillator drifts) and to define the sampling time and the number of samples (one or three) to be taken within a bit time.

Transceiver Logic (TCL)

The TCL controls the transmit output driver.

Error Management Logic (EML)

The EML is responsible for the error confinement of the transfer-layer modules. The EML gets error announcements from BSP and then informs the BSP, TCL, and IML about error statistics.

CONTROL SEGMENT AND MESSAGE BUFFER DESCRIPTION

The CAN Controller appears to the CPU as an on chip memory mapped peripheral, guaranteeing the independent operation of both parts.

Address Allocation

The address area of the CAN Controller consists of the Control Segment and the message buffers. The Control Segment is programmed during an initialization down-load in order to configure communication parameters (e.g., bit timing). The communication over the CAN-bus is also controlled via this segment by the CPU. A message which is to be transmitted, must be written to the Transmit Buffer. After a successful reception the CPU may read the message from the Receive Buffer and then release it for further use.

Control Segment Layout

The exchange of status, control and command signals between the CPU and the CAN Controller is performed in the control segment. The layout of this segment is shown in Figure 1. After an initial down-load, the contents of the registers Acceptance Code, Acceptance Mask, bus Timing 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 "Control Register").

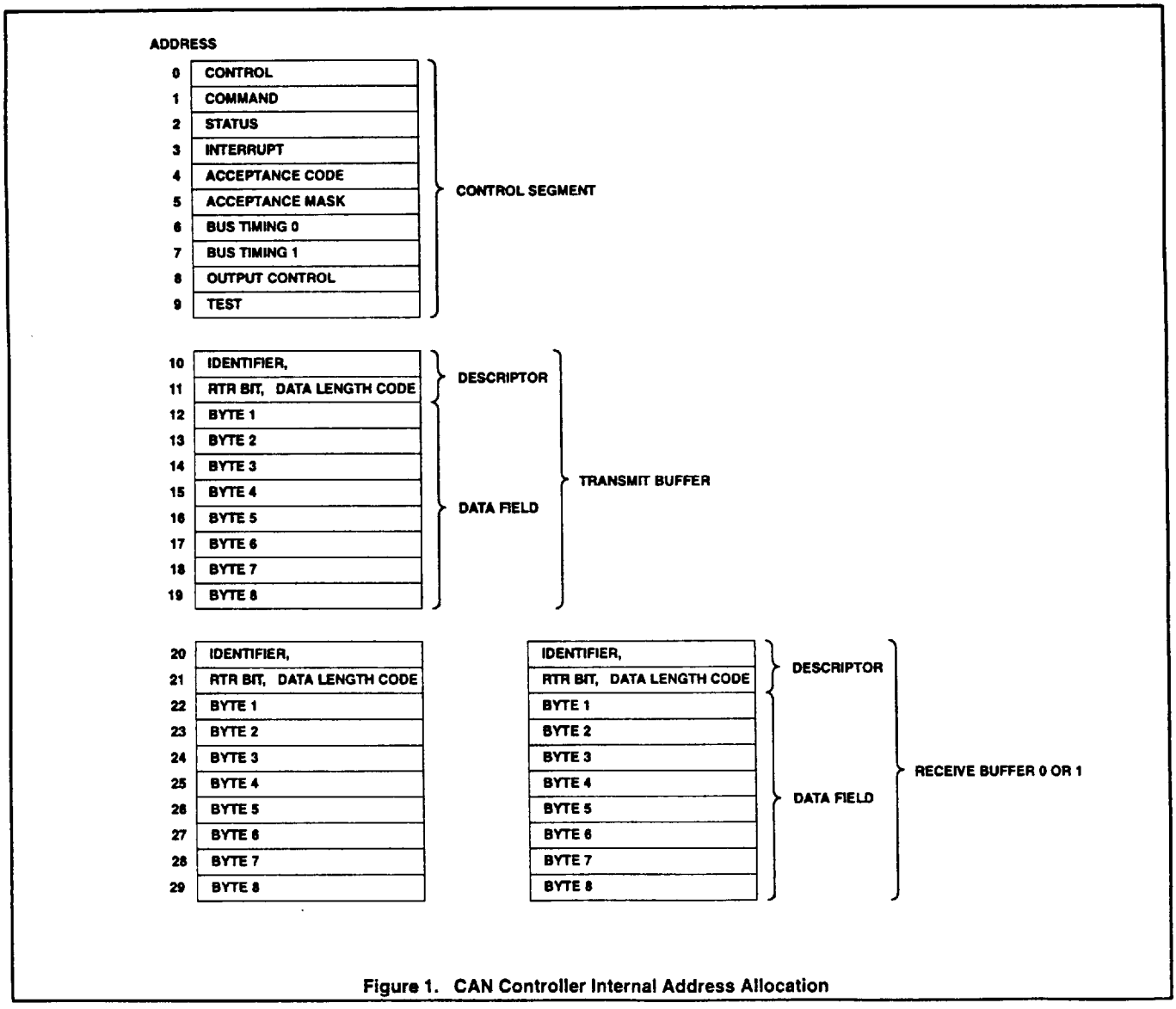


Figure 1. CAN Controller Internal Address Allocation

Table 1. CAN Registers

DESCRIPTION	ADDRESS	7 (MSB)	6	5	4	3	2	1	0 (LSB)
Control Segment									
Control Register	0	Test Mode	Synch	Reference Active	Overrun Interrupt Enable	Error Interrupt Enable	Transmit Interrupt Enable	Receive Interrupt Enable	Reset Request
Command Register	1	RX0 Active	RX1 Active	Wake-Up Mode	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 Status
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	COTN0	OCPOL0	OCMODE1	OCMODE0
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

Control Register (CR)

The contents of the Control Register are used to change the behavior of the CAN Controller. Control bits may be set or reset by the CPU 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.0	RR	Reset Request[1]	HIGH (present)	Detection of a Reset Request results in the CAN Controller aborting the current transmission/reception of a message entering the reset state.
			LOW (absent)	On the HIGH-to-LOW transition of the Reset Request bit, the CAN Controller returns to its normal operating state.
CR.1	RIE	Receive Interrupt Enable	HIGH (enabled)	When a message has been received without errors, the CAN Controller transmits a Receive Interrupt signal to the CPU.
			LOW (disabled)	No transmission of the Receive Interrupt signal by the CAN Controller to the CPU.
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 CAN Controller transmits a Transmit Interrupt signal to the CPU.
			LOW (disabled)	No transmission of the Transmit Interrupt signal by the CAN Controller to the CPU.
CR.3	EIE	Error Interrupt Enable	HIGH (enabled)	If the Error or Bus Status change (see status Register), the CPU receives an Error Interrupt signal.
			LOW (disabled)	The CPU receives no Error Interrupt signal.
CR.4	OIE	Overrun Interrupt Enable	HIGH (enabled)	If the Data Overrun bit is set (see Status Register), the CPU receives an Overrun Interrupt signal.
			LOW (disabled)	The CPU receives no Overrun Interrupt signal from the CAN Controller.
CR.5	RA	Reference Active[2]	HIGH (output)	The pin REF is an $AV_{DD/2}$ reference output.
			LOW (input)	A reference voltage may be input.
CR.6	S	Synch[2]	HIGH (2 edges)	Bus-line transitions from recessive-to-dominant and vice versa are used for resynchronization.
			LOW (1 edge)	Only transitions from recessive-to-dominant are used for resynchronization.
CR.7	–	RESERVED		

NOTES:

1. During an external reset (RST = HIGH) or when the Bus Status bit is set HIGH (Bus-Off), the IML forces the Reset Request HIGH (present). During an external reset the CPU cannot set the Reset Request bit LOW (absent). Therefore, after having set the Reset Request bit LOW (absent), the CPU 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 CAN controller will wait for:
   - one occurrence of the Bus-Free signal (11 recessive bits), if the preceding reset (Reset Request = HIGH) has been caused by an external reset or a CPU initiated reset.
   - 128 occurrences of Bus-Free, if the preceding reset (Reset Request = HIGH) has been caused by a CAN Controller initiated Bus-Off, before re-entering the Bus-On mode.
   - When Reset Request is set HIGH (present), for whatever reason, the control, command, status and interrupt bits are affected, see Table 3. Only, when Reset Request is set HIGH (present) the registers at addresses 4 to 8 are accessible.
2. A modification of the bits Reference Active and Synch is only possible with Reset Request = HIGH (present). It is allowed to set these bits while Reset Request is changed from HIGH to LOW. After an external reset (pin RST = HIGH) the Reference Active bit is set HIGH (output), the Synch bit is undefined.

Table 3. Effects of Setting the Reset Request Bit HIGH (present)

TYPE	BIT	EFFECT
Control	Reference Active	HIGH (output), only after an external reset
Command	RX0 Active / RX1 Active	HIGH (RX0=CRX0, RX1=CRX1), only after an external reset
	Sleep	LOW (wake-up)
	Clear Overrun Status	HIGH (clear)
	Release Receive Buffer	HIGH (released)
	Abort Transmission	LOW (absent)
	Transmission Request	LOW (absent)
Status	Bus Status	LOW (Bus-On), only after an external reset
	Error Status	LOW (no error), only after an external reset
	Transmit Status	LOW (idle)
	Receive Status	LOW (idle)
	Transmission Complete Status	HIGH (complete)
	Transmit Buffer Access	HIGH (released)
	Data Overrun	LOW (absent)
	Receive Buffer Status	LOW (empty)
Interrupt	Overrun Interrupt	LOW (reset)
	Transmit Interrupt	LOW (reset)
	Receive Interrupt	LOW (reset)

Command Register (CMR)

A command bit initiates an action within the transfer layer of the CAN controller. the Command Register appears to the CPU as a read/write memory, except of the bits CMR.0 to CMR.3, which return HIGH if being read.

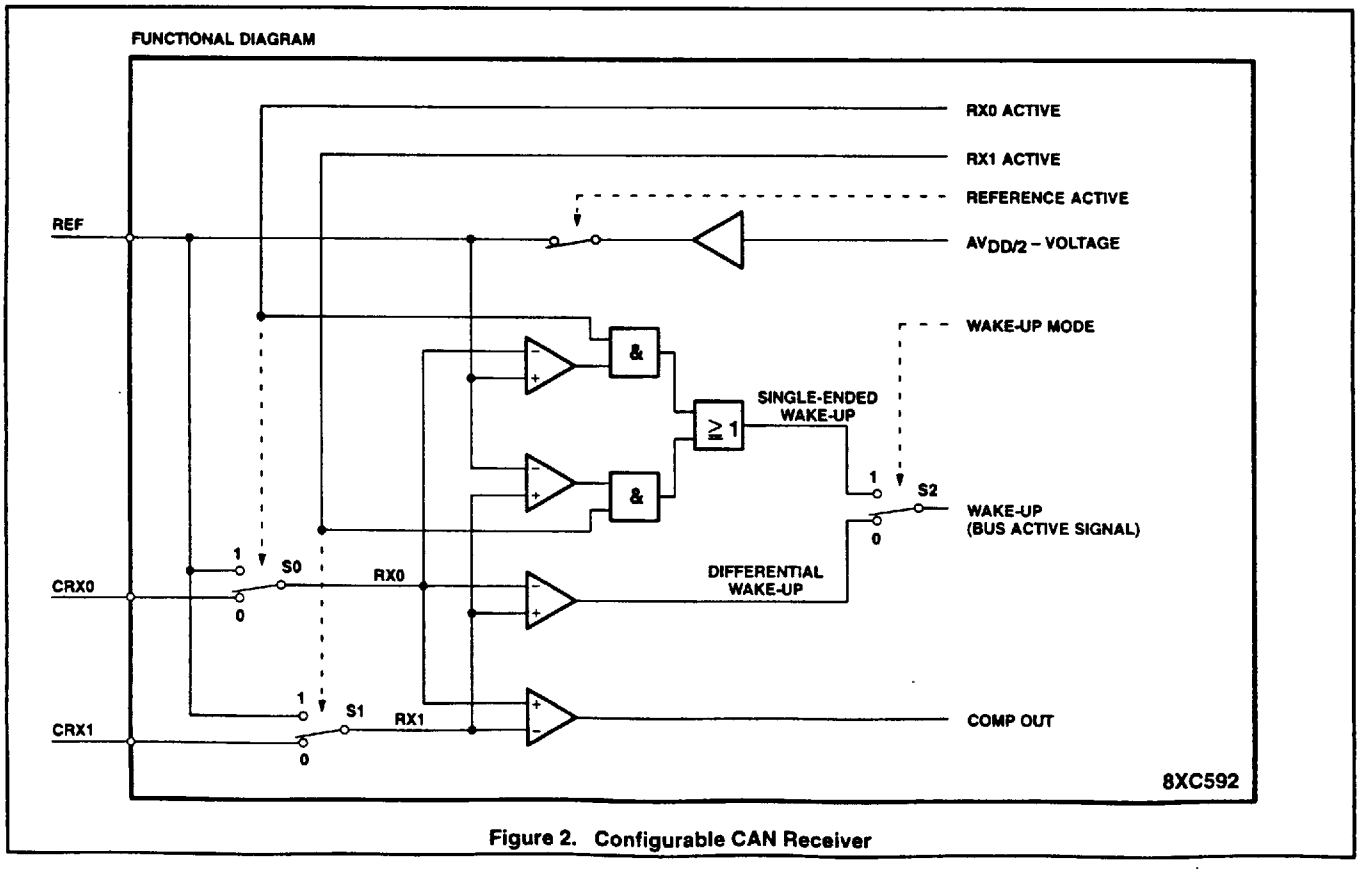


Figure 2. Configurable CAN Receiver

Table 4. Description of the Command Register Bits

CMR	ADDRESS 1				
BIT	SYMBOL	NAME	VALUE		FUNCTION
CMR.0	TR	Transmission Request[1]	HIGH (present)		A message shall be transmitted.
			LOW (absent)		No action.
CMR.1	AT	Abort Transmission[2]	HIGH (present)		If not already in progress, a pending Transmission Request is cancelled.
			LOW (absent)		No action.
CMR.2	RRB	Release Receive Buffer[3]	HIGH (released)		The Receive Buffer attached to the CPU is released.
			LOW (no action)		No action.
CMR.3	COS	Clear Overrun[4]	HIGH (clear)		The Data Overrun status bit is set to LOW (see Status Register).
			LOW (no action)		No action.
CMR.4	SLP	Sleep[5]	HIGH (sleep)		The CAN Controller enters sleep mode if no CAN interrupt is pending and there is no bus activity.
			LOW (wake up)		The CAN Controller functions normally.
CMR.5	WUM	Wake-Up Mode[6]	HIGH (single ended)		The difference of the RX signals to the internal reference voltage $AV_{DD/2}$ is used for wake up.
			LOW (differential)		The differential signal between RX0 and RX1 is used for wake up.
CMR.6	RX1A	RX1 Active[7]	RX0 Active	RX1 Active	See Figure 2.
CMR.7	RX0A	RX0 Active[7]	1	1	RX0 = CRX0, RX1 = CRX1
			1	0	RX0 = CRX0, RX1 = $AV_{DD/2}$
			0	1	RX0 = $AV_{DD/2}$, RX1 = CRX1
			0	0	No action.

NOTES:

1. 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).
2. The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g., to transmit an urgent message. A transmission already in progress is not stopped. In order to see if the original message had been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. this should be done after the Transmit Buffer Access bit has been set HIGH (released) or a Transmit Interrupt has been generated (see Interrupt Register).
3. After reading the contents of the Receive Buffer (RBF0 or RBF1) the CPU must release this buffer by setting Release Receive Buffer bit HIGH (released). This may result in another message becoming immediately available.
4. This command bit is used to acknowledge the Data Overrun condition signaled by the Data Overrun status bit. It may be given or set at the same time as a Release Receive Buffer command bit.
5. The CAN Controller will enter sleep mode, if the Sleep bit is set HIGH (sleep), there is no bus activity and no interrupt is pending. A CAN Controller will wake up after the Sleep bit is set LOW (wake up) or when there is bus activity. On wake up, a Wake-Up Interrupt (see Interrupt Register) is generated (see "Power Reduction Modes"). A CAN Controller which is sleeping and then awoken by bus activity will not be able to receive this message until it detects a Bus-Free signal. The Sleep bit, if being read, reflects the status of the CAN Controller.
6. The Wake-Up Mode bit should be set at the same time as the Sleep bit. The differential wake up mode is useful if both bus wires are fully functioning; it minimizes the probability of wake ups due to noise. The single ended wake up mode is recommended if a wake up must be possible even if one bus wire is already or may become disturbed (See Figure 2).
7. The RX0/RX1 Active bits, if being read, reflect the status of the respective switches (See Figure 2). It is recommended to change the switches only during the reset state (Reset Request is HIGH).

Status Register (SR)

The contents of the Status Register reflects the status of the CAN Controller. The Status Register appears to the CPU as a read only memory.

Table 5. Description of the Status Register Bits

SR	ADDRESS 2			
BIT	SYMBOL	NAME	VALUE	FUNCTION
SR.0	RBS	Receive Buffer Status[1]	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.
SR.1	DO	Data Overrun[2]	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.2	TBS	Transmit Buffer Access[3]	HIGH (released)	The CPU may write a message into the TBF.
			LOW (locked)	The CPU cannot access the Transmit Buffer. A message is either waiting for transmission or is in the process of being transmitted.
SR.3	TCS	Transmit Complete Status[3]	HIGH (complete)	Last requested transmission has been successfully completed.
			LOW (incomplete)	Previously requested transmission is not yet completed.
SR.4	RS	Receive Status[4]	HIGH (receive)	The CAN Controller is receiving a message.
			LOW (idle)	No message is received.
SR.5	TS	Transmit Status[4]	HIGH (transmit)	The CAN Controller is transmitting a message.
			LOW (idle)	No message is transmitted.
SR.6	ES	Error Status	HIGH (error)	At least one of the Error Counters has reached the CPU Warning limit.
			LOW (ok)	Both Error Counters have not reached the warning limit.
SR.7	BS	Bus Status[5]	HIGH (Bus-Off)	The CAN Controller is not involved in bus activities.
			LOW (Bus-On)	The CAN Controller is involved in bus activities.

NOTES:

1. If the command bit Release Receive Buffer is set HIGH (released) by the CPU, 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.
2. 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 CAN Controller will lose arbitration and so become a receiver of the message. If no Receive Buffer is available, Data Overrun is signaled. However, this transmitted and accepted message does neither cause a Receive Interrupt nor set the Receive Buffer Status bit to HIGH (full). Also, a Data Overrun does not cause the transmission of an Overload Frame.
3. If the CPU 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 signaled. 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 both the Receive Status and Transmit Status bits are LOW (idle) the CAN-bus is idle.
5. When the Bus Status bit is set HIGH (Bus-Off), the CAN Controller will set the Reset Request bit HIGH (present). It will stay in this state until the CPU sets the Reset Request bit LOW (absent). Once this is completed, the CAN Controller 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.

Interrupt Register (IR)

The Interrupt Register allows the identification of an interrupt source. When one or more bits of this register are set, a CAN interrupt (SIO1) will be indicated to the CPU. All bits are reset by the CAN Controller after this register is read by the CPU. This register appears to the CPU as a read only memory.

Table 6. Description of the Interrupt Register Bits

IR	ADDRESS 3			
BIT	**SYMBOL**	**NAME**	**VALUE**	**FUNCTION**
IR.0	RI	Receive Interrupt[1]	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 CPU.
IR.1	TI	Transmit Interrupt	HIGH (set)	This bit is set on a change of the Transmit Buffer Access 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 CPU.
IR.2	EI	Error Interrupt	HIGH (set)	This bit is set on a change of either the Error Status or Bus Status bits (see Status Register) 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 CPU.
IR.3	OI	Overrun Interrupt[2]	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 the Interrupt register by the CPU.
IR.4	WUI	Wake-Up Interrupt	HIGH (set)	The Wake-Up Interrupt bit is set HIGH when the sleep mode is left (see Command Register).
			LOW (reset)	Wake-Up Interrupt bit is reset by a read access of the Interrupt Register by the CPU.
IR.5	–	RESERVED		
IR.6	–	RESERVED		
IR.7	–	RESERVED		

NOTES:
1. Receive Interrupt bit (if enabled) and Receive Buffer Status bit (see Status Register) are set at the same time.
2. Overrun Interrupt bit (if enabled) and Data Overrun bit (see Command Register) are set at the same time.

Acceptance Code Register (ACR)

The Acceptance Code Register is part of the acceptance filter of the CAN Controller. 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 Figure 1) 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 Status and Interrupt Register. 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) EQUAL (AC.7 . . AC.0)] or (AM.7 . . AM.0) = 1111 1111_b

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

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

Acceptance Mask Register (AMR)

The Acceptance Mask Register is part of the acceptance filter of the CAN Controller. 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.

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

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.

Bus Timing Register 0 (BTR 0)

The contents of Bus Timing Register 0 defines the values of the 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).

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 8XC592 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)

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 can be accessed (read/write) if the Reset Request bit is set HIGH (present).

Bus Timing Register 1 Bits

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

Sampling (SAM)

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

SAM = LOW is recommended for high speed buses (SAE class C), while SAM = HIGH is recommended for slow/medium speed buses (class A and B) where filtering of spikes on the bus-line is beneficial (see "Bit Timing Restrictions").

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 + 2TSEG.1 + TSEG1.0 + 1)

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

For further information on bus timing see Bus Timing Register 0 and "Bus Timing/ Synchronization".

Output Control Register (OCR)

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

Output Control Register Bits

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

If the CAN Controller is in the sleep mode (Sleep = HIGH) a recessive level is output on the CTX0 and CTX1 pins. If the CAN Controller 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 CTX0 and CTX1. The voltage levels on the output driver pins CTX0 and CTX1 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 Figure 3).

Clock Output Mode

For the CTX0 pin this is the same as in Normal Output Mode. However, the data stream to CTX1 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_{SCL}.

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 de-activated (floating). Dominant bits are sent alternatingly on CTX0 and CTX1, i.e., the first dominant bit is sent on CTX0, the second is sent on CTX1, and the third one is sent on CTX0 again, etc.

Test Output Mode

For the CTX0 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 CTX1. This mode is used for production testing only.

The following two tables, Table 7 and Table 8, show the relationship between the bits of the Output Control Register and the two serial output pins CTX0 and CTX1 of the 8XC592 – CAN Controller, connected to the serial bus (see Block Diagram, page 855).

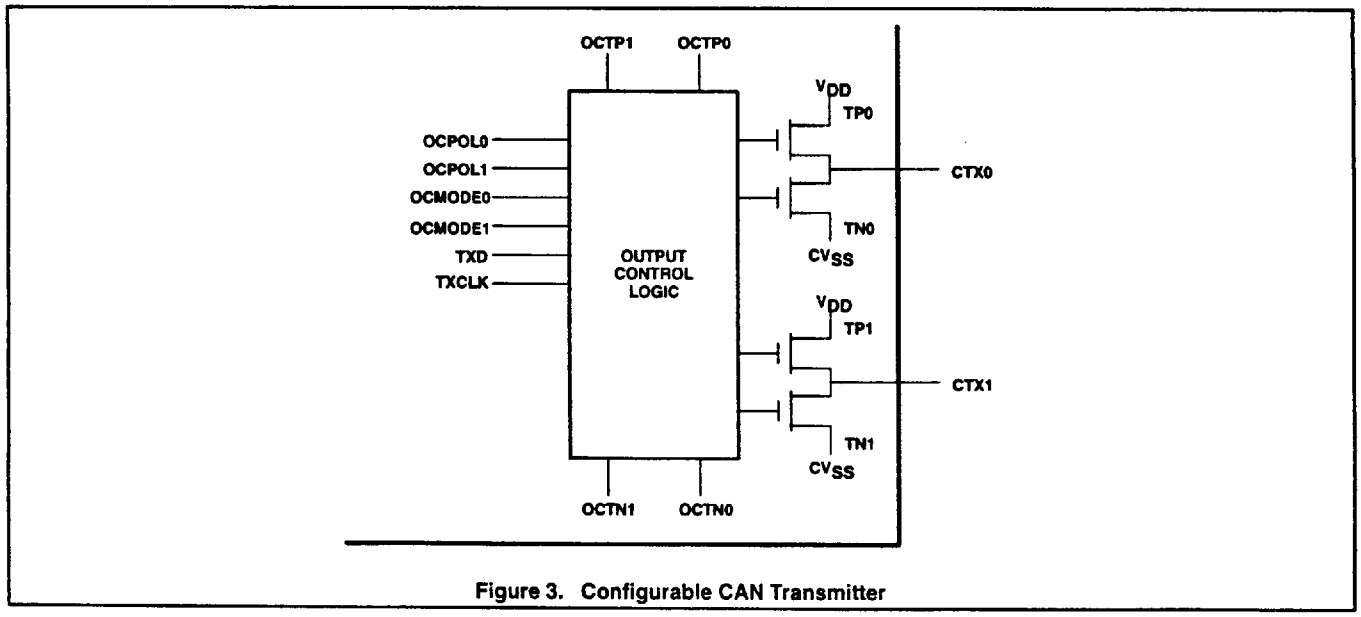


Figure 3. Configurable CAN Transmitter

Table 7. Description of the Output Mode Bits

OCMODE1	OCMODE0	DESCRIPTION
1	0	Normal output Mode; CTX0, CTX1: bit sequence (TXD; note 1)
1	1	Clock output Mode; CTX0: bit sequence, CTX1: bus clock (TXCLK)
0	0	Bi-phase output Mode
0	1	Test output Mode; CTX0: bit sequence, CTX1: COMP OUT

NOTE:
1. TXD is the data bit to be transmitted.

Table 8. Output Pin Set-Up

DRIVE	OCTPx	OCTNx	OCPOLx	TXD	TPx	TNx	CTXx
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 CV_{SS}; x = 0 or 1.
3. CTXx is the serial output level on CTX0 or CTX1. It is required that the output level on the CAN bus is dominant with TXD = 0 and recessive with TXD = 1.

TRANSMIT BUFFER LAYOUT

The global layout of the Transmit Buffer is shown in Figure 1. This buffer serves to store a message from the CPU to be transmitted by the CAN Controller. It is subdivided into Descriptor and Data Field. the Transmit Buffer can be written to and read from by the CPU.

DESCRIPTOR

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

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.

Remote Transmission Request Bit (RTR)

Description of the RTR Bit

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

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 0,1,2,... and 8 should be used.

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.

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 Figure 1).

Table 9. The Special Function Registers Between CPU and CAN

SFR	ADR	ACS	MSB 7	6	5	4	3	2	1	LSB 0
CANADR	DBh	RW	DMA	reserved[3]	Auto Inc	CANA4	CANA3	CANA2	CANA1	CANA0
CANDAT	DAh	RW	CAND7	CAND6	CAND5	CAND4	CAND3	CAND2	CAND1	CAND0
CANCON[1]	D9h	R	reserved[3]	reserved[3]	reserved[3]	Wake Up Interrupt	Overrun Interrupt	Error Interrupt	Transmit Interrupt	Receive Interrupt
		W	RX0 Active	RX1 Active	Wake Up Mode	Sleep	Clear Overrun	Release RxBuf	Abort Transm	Transm Request
CANSTA[1,2]	D8h	R	Bus Status	Error Status	Transmit Status	Receive Status	TxComp1 Status	TxBuf Access	Data Overrun	RxBuf Status
		W	RAMA7	RAMA6	RAMA5	RAMA4	RAMA3	RAMA2	RAMA1	RAMA0

NOTES:
1. Do not use a RMW instruction.
2. The bit addresses of CANSTA (7:0) are DFH . . D8H.
3. Reserved bits are read as HIGH.

HANDLING OF THE CPU-CAN INTERFACE

Via the four special registers CANADR, CANDAT, CANCON and CANSTA, the CPU has access to the CAN Controller and also to the DAM-logic. Note that CANCON and CANSTA have different meanings for a read and write access.

CANADR

The five least significant bits CANADR.4 down to CANADR.0 (CANA4 . . . CANA0) define the address of one of the CAN Controller internal registers to be accessed via CANDAT. For instance, after an external hardware (e.g., power-on) reset CANADR contains the value 64h, and hence the CPU accesses (read/write) the Acceptance Code register of the CAN Controller, via the Special Function Register CANDAT. CANADR also controls the auto address increment mode via bit CANADR.5 (AutoInc) and the DMA-logic via bit CANADR.7 (DMA). CANADR is implemented as a read/write register.

CANDAT

The Special Function Register CANDAT appears as a port to the CAN Controller's internal register (memory location) being selected by CANADR. Reading or writing CANDAT is effectively an access to that CAN Controller internal register, which is selected by CANADR. CANDAT is implemented as a read/write register.

CANCON

When reading CANCON the Interrupt Register of the CAN Controller is accessed, while writing to CANCON means an access to the Command Register. CANCON is implemented as a read/write register.

CANSTA

Reading CANSTA is an access to the Status Register of the CAN Controller. Writing to CANSTA sets the address of the on-chip Main RAM (internal data memory) for a subsequent DMA transfer. CANSTA is implemented as a bit-addressable read/write register.

Auto Address Increment

With the auto address increment mode a fast stack-like reading and writing of CAN Controller internal registers is provided. If the bit CANADR.5 (AutoInc) is high, the content of CANADR is incremented automatically after any read or write access to CANDAT. For instance, loading a message into the Transmit Buffer can be done by writing 2AH into CANADR and then moving byte by byte of the message to CANDAT.

Incrementing CANADR beyond $xx111111_b$ resets the bit CANADR.5 (AutoInc) automatically (CANADR = $xx000000_b$).

High Speed DMA

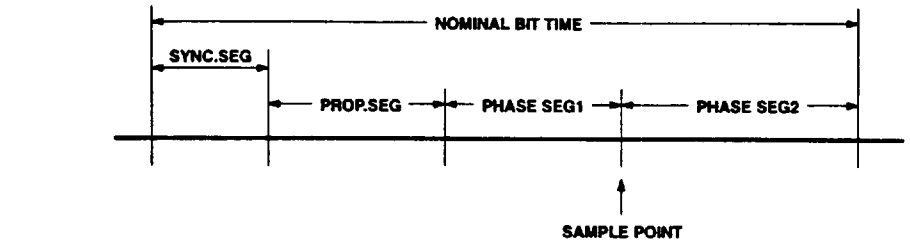
The DMA-logic allows to transfer a complete message (up to 10 bytes) between CAN Controller and Main RAM in 2 instruction cycles at maximum; up to 4 bytes are transferred in 1 instruction cycle. The performance of the CPU is strongly enhanced because this very fast transfer is don in the background. A DMA transfer is achieved by first writing the RAM address (0 . . FFH) into CANSTA and then setting the TX- or RX-Buffer address in CANADR and the bit CANADR.7 (DMA) simultaneously; the RAM address points to the location of the first byte to be transferred. Setting the DMA bit causes an automatic evaluation of the Data Length Code and then the transfer; for a TX-DMA transfer the Data Length Code is expected at the location "RAM address + 1".

In order to program a TX-DMA transfer, the value 8AH (address 10) has to be written into CANADR. Then a complete message, consisting of the 2-byte Descriptor and the Data Field (0 . . . 8 bytes), starting at location "RAM address" is transferred to the TX-Buffer.

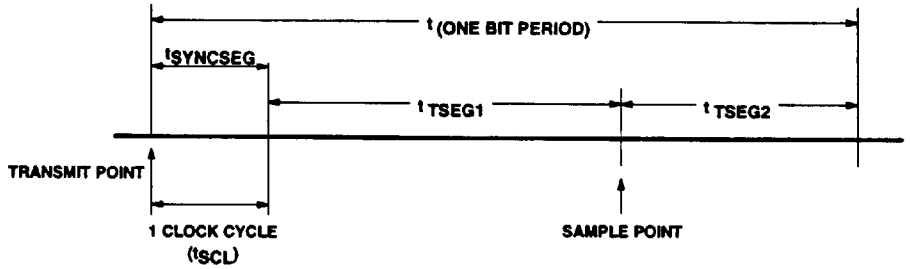
The RX-DMA transfer is very versatile. By writing a value in the range of 94H (address 20) up to 9DH (address 29) of CANADR the whole or a part of the received message, starting at the specified address, is transferred to the internal data memory. This allows e.g. to transfer the bytes of the Data Field only.

After a successful DMA transfer the DMA- (and Auto Inc-) bit is reset.

During a DMA transfer the CPU can process the next instruction. However, an access to the data memory, CANADR, CANDAT, CANCON or CANSTA is not allowed. After having set the DMA-bit, every interrupt is disabled until the end of the transfer. Note, that disadvantageous programming may lead to an interrupt response time of at most 10 instruction cycles. The shortest interrupt response time is achieved by using a 2-cycle instructions directly after setting the DMA-bit. During the reset state (bit Reset Request is HIGH) a DMA transfer is not possible.



a) As defined by the CAN protocol.



a) As implemented in the 8XC592's on-chip CAN Controller.

Figure 4. Bit Period

BUS TIMING / SYNCHRONIZATION

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

- 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 CAN Controller. The BTL uses the following three registers:

- Control register (Synch)
- Bus Timing Register 0
- Bus Timing Register 1

Bit Timing

A bit period is built up from a number of system clock cycles (t_{SCL}) (see "Bus Timing Register 0"). One bit period is the result of the addition of the programmable segments TSEG1 and TSEG2 and the general segment SYNCSEG.

Synchronization Segment (SYNCSEG)
The incoming edge of a bit is expected during this state; this state corresponds to one system clock cycle ($1 \times t_{SCL}$).

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.

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 causes all CAN Controllers to perform a "hard synchronization" on the first recessive-to-dominant edge. During arbitration, however, several CAN Controllers 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 CAN Controller 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 segment directly before the sample point (see Figure 4).

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.

TSEG2 is programmable from 1 to 8 system clock cycles.

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

Propagation Delay Time (t_{PROP})
The propagation delay time 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}.

$t_{PROP} = 2 \times$ (physical bus delay
+ input comparator delay
+ output driver delay)

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:

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

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.

Hard Synchronization
This type of synchronization occurs only at the beginning of a message. The CAN Controller synchronizes on the first incoming recessive-to-dominant edge of a message (being the leading edge of a message's Start-Of-Frame bit).

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

- Variations in individual CAN Controller 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};
- 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}.

Synchronization Rules
The synchronization rules are as follows:

- Only one synchronization within one bit time 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.
- All other edges (recessive-to-dominant and optionally dominant-to-recessive edges if the Synch bit is set HIGH) which are candidates for resynchronization will be used with the following exception:
  - A transmitting CAN Controller 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.

CAN-PROTOCOL DESCRIPTION

Frame Types

The 8XC592's CAN Controller supports the four different CAN protocol frame types for communication:

- Data Frame, to transfer data
- Remote Frame, request for data
- Error Frame, globally signals a (locally) detected error condition
- Overload Frame, to extend delay time of subsequent frames (an Overload Frame is not initiated by the 8XC592 CAN Controller.

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 CAN Controllers 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.

Data Frame

A Data Frame carries data from a transmitting CAN Controller to one or more receiving ones. A Data Fram 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

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 CAN Controller in receive mode.

Arbitration Field

Consists of the message Identifier and the RTR bit. In the case of simultaneous message transmissions by two or more CAN Controllers 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 CAN Controller will receive the frame because a CAN-based communication network does not differentiate between a point-to-point, multicast or broadcast communication.

RTR Bit

A CAN Controller, 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.

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 (destuffed; number of data bytes to be transmitted/received) in the 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 (0,1, ..., 8). A logic 0 (logic 1) in the Data Length Code is transmitted as dominant (recessive) bus level, respectively.

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. The Data Field can contain from 0 up to 8 bytes. The most significant bit of the first data byte (lowest address) is transmitted/received first.

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 CAN Controller is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits, CRC Error detection. 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).

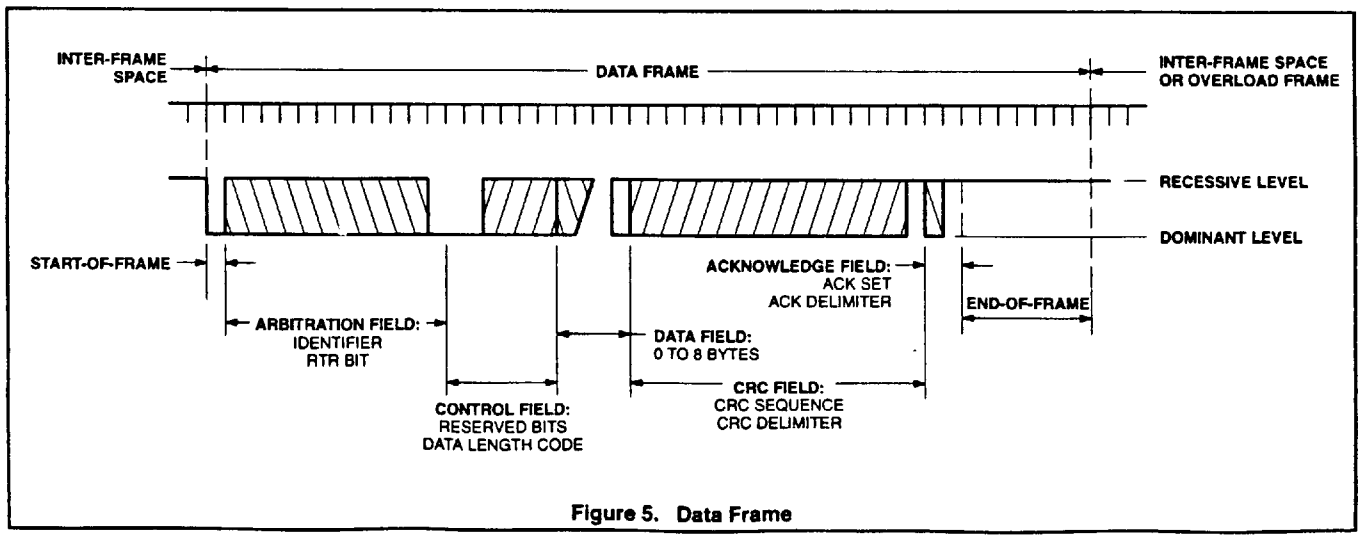


Figure 5. Data Frame

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 CAN Controllers having received the matching CRC Sequence, report this by overwriting the transmitter's recessive bit in the Acknowledge Slot with a dominant bit. 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 all CAN Controllers (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.

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. The bit-stuffing logic is deactivated during the End-of-Frame sequence.

Remote Frame

A CAN Controller 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 value of the Data Length Code should be the one of the corresponding Data Frame, although it 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

Error Frame

The Error Frame consists of two different fields. The first field is accomplished by the superimposing of Error Flags contributed from different CAN Controllers.

The second field is the Error Delimiter.

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 CAN Controllers.

An error-active CAN Controller detecting an error condition signals this by transmission of an Active Error Flag. This Error Flag's form violates the bit-stuffing rule 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. Consequently, all other CAN Controllers 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 CAN Controllers. The total length of this sequence varies between six (minimum) and twelve (maximum) bits.

An error-passive CAN Controller detecting an error condition tries to signal this by transmission of a Passive Error Flag. The error-passive CAN Controller 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.

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 CAN Controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN Controller has finished sending its Error Flag and has additionally sent the first out of the 8 recessive bits of the Error Delimiter. Afterwards all CAN Controllers transmit the remaining recessive bits. After this event and an Intermission Field all error-active CAN Controllers within the network can start a transmission simultaneously.

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

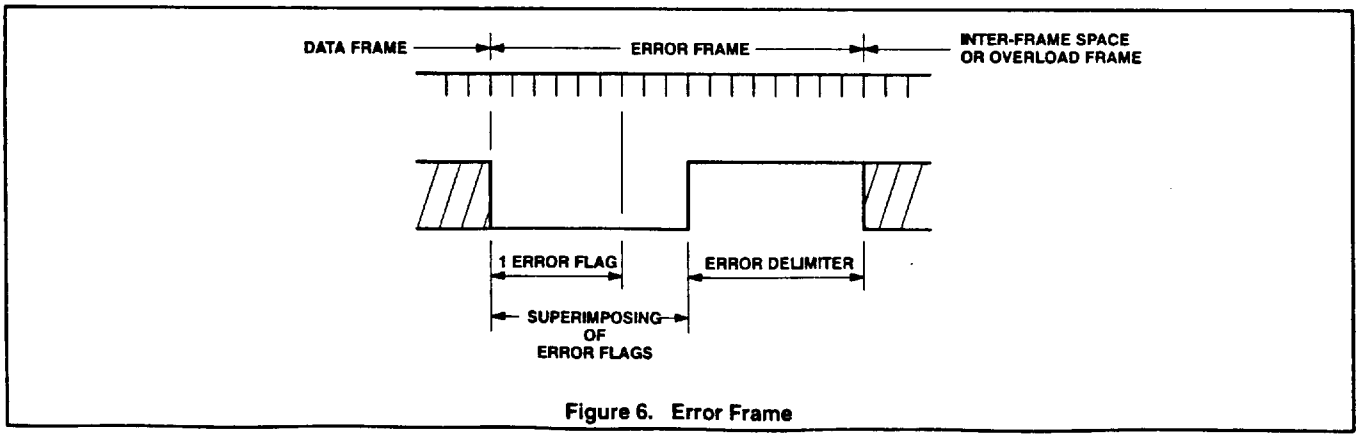


Figure 6. Error Frame

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

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

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.

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 8XC592's on chip CAN Controller 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 from 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.

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 CAN Controllers detecting the overload condition also transmit an Overload Flag (condition 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 CAN Controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN Controller has finished sending its Overload Flag and all CAN Controllers start simultaneously transmitting seven more recessive bits.

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 CAN Controllers also send a Suspend Transmission after transmission of a message. Overload Frames and Error Frames are not preceded by an Inter-Frame Space.

Intermission Field

The Intermission Field consists of three recessive bits. During an Intermission period, no frame transmissions will be started by the 8XC592's on chip CAN Controller. 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.

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.

Bus Organization

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

Bus Access

CAN Controllers only start transmission during the Bus-Idle state. All CAN Controllers synchronize on the leading edge of the Start-of-Frame (hard synchronization).

Arbitration

If two or more CAN Controllers 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 CAN Controller compares its transmitted bit level with the monitored bus level. Any CAN Controller 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 bit pattern ID = $1111111XXXX_b$ (X being bits of arbitrary level) is forbidden.

The number of available different Identifiers is 2032 ($2^{11} - 2^4$).

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

When a receiving CAN Controller 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 signaled.

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.

Error Signaling

A CAN Controller 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 or corrupts the fixed form bit fields. A violation of the bit-stuffing law affects any CAN Controller which detects the error condition. These devices will also transmit an Error Flag.

An error-passive CAN Controller which detects an error condition, transmits a Passive Error Flag. A Passive Error Flag is not able to interrupt a current message at different CAN Controllers but this type of Error Flag may be ignored (overwritten) by other CAN Controllers. After having detected an error condition, an error-passive CAN Controller 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 CAN Controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN Controller has finished transmitting its Error Flag and all CAN Controllers start transmitting seven additional recessive bits.

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

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

Overload Signaling
Some CAN Controllers (but not the one on-chip of the 8XC592) 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 Field. 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.

Error Detection

The processes described in the following paragraphs are implemented in the 8XC592's on-chip CAN Controller for error detection.

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

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

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:

- A disturbance generates more than the allowed five consecutive bits with identical polarity. These errors are detected by all CAN Controllers.
- 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 CAN Controllers or Bit Error detection by the transmitting CAN Controller.

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 CAN Controller of the respective message) used for error detection.

The code used by all CAN Controllers 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 the CAN Controller)
- 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 determined (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 polynomial with coefficients 0 or 1.
2. This polynomial is divided (modulo–2) by the following generator polynomial, which includes a parity check:

   $f(X) = (X^{14} + X^9 + X^8 + X^6 + X^5 + X^4 + X^2 + X + 1)(X + 1) = 1100010110011001_b$.
3. The remainder of this polynomial division is the CRC sequence.

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} ($\approx 3 \cdot 10^{-5}$) by CRC check only.

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

- CRC Delimiter
- Acknowledge Delimiter
- End-of-Frame
- Error Delimiter
- Overload 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.

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

Error Detection by an Error Flag of another CAN Controller
The detection of an error is signaled by transmitting an Error Flag. An Active Error Flag causes a Stuff Error, a Bit Error or a Form Error at all other CAN Controllers.

Error Detection Capabilities
Errors which occur at all CAN Controllers (global errors) are 100% detected. For local errors, i.e., for errors occurring at some CAN Controllers 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 $3 \cdot 10^{-5}$.

Error Confinement (definitions)

Bus-Off
A CAN Controller 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 set to 0.

Acknowledge (ACK)
A CAN Controller 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.

Error-Active
An error–active CAN Controller in its normal operating state is able to receive and to transmit normally and also to transmit an Active Error Flag.

Error-Passive
An error–passive CAN Controller 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-active CAN Controller (e.g., due to a malfunction) is reduced.

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

Start-Up
A CAN Controller which either was switched off or is in the Bus-Off state, must run a Start-Up routine in order to:

- Synchronize with other available CAN Controllers before starting to transmit. Synchronization is achieved, when 11 recessive bits, equivalent to Acknowledge Delimiter, End-of-Frame and Intermission Field, have been detected (Bus-Free).
- Wait for other CAN Controllers without passing into the Bus-Off state (due to a missing acknowledge), if there is no other CAN Controller currently available.

Aims of Error Confinement

Distinction of Short and Long Lasting Disturbances
The CPU must be informed when there are long-lasting disturbances and when bus activities have returned to normal operation.

During long-lasting disturbances, a CAN Controller enters the Bus-Off state and the CPU may use default values.

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

Detection and Localization of Hardware Disturbance and Defects
The rules for error confinement are defined by the CAN protocol specification (and implemented in the 8XC592's on-chip CAN Controller), in such a way that the CAN Controller, 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.

Error Confinement
All CAN Controllers 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 ration 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 signaled by the CAN Controller (Error Status, Error Interrupt).

A CAN Controller 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 CAN Controller entering the Bus-Off state.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronously to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution a multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response latency is from 2.25µs to 7.5µs when using a 16MHz crystal. The latency time depends on the sequence of instructions executed directly after an interrupt request. During a CAN-DMA transfer the interrupt system is disabled. The 8XC592 acknowledges interrupt requests from fifteen sources as follows:

- $\overline{INT0}$ and $\overline{INT1}$: externally via pins 27 and 28 respectively
- Timer 0 and Timer 1: from the two internal counters
- Timer T2 (8 separate interrupts): 4 capture interrupts, 3 compare interrupts and an overflow interrupt
- ADC end-of-conversion interrupt
- CAN Controller interrupt
- UART serial I/O port interrupt.

Interrupt Enable Registers

IEN0 (A8H)

7	6	5	4	3	2	1	0
EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0

Bit	Symbol	Function
IEN0.7	EA	General enable/disable control 0 = No interrupt is enabled 1 = Any individually enabled interrupt will be accepted
IEN0.6	EAD	Enable ADC interrupt
IEN0.5	ES1	Enable SIO1 (CAN) interrupt
IEN0.4	ES0	Enable SIO0 (UART) interrupt
IEN0.3	ET1	Enable Timer 1 interrupt
IEN0.2	EX1	Enable External 1 interrupt
IEN0.1	ET0	Enable Timer 0 interrupt
IEN0.0	EX0	Enable External 0 interrupt

IEN1 (E8H)

7	6	5	4	3	2	1	0
ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0

Bit	Symbol	Function
IEN1.7	ET2	Enable T2 overflow interrupt(s)
IEN1.6	ECM2	Enable T2 comparator 2 interrupt
IEN1.5	ECM1	Enable T2 comparator 1 interrupt
IEN1.4	ECM0	Enable T2 comparator 0 interrupt
IEN1.3	ECT3	Enable T2 capture register 3 interrupt
IEN1.2	ECT2	Enable T2 capture register 2 interrupt
IEN1.1	ECT1	Enable T2 capture register 1 interrupt
IEN1.0	ECT0	Enable T2 capture register 0 interrupt

Where: 0 = interrupt disabled.
1 = interrupt enabled.

IP0 (B8H)

7	6	5	4	3	2	1	0
–	PAD	PS1	PS0	PT1	PX1	PT0	PX0

Bit	Symbol	Function
IP0.7	–	Unused
IP0.6	PAD	ADC interrupt priority level
IP0.5	PS1	SIO1 (CAN) interrupt priority level
IP0.4	PS0	SIO0 (UART) interrupt priority level
IP0.3	PT1	Timer 1 interrupt priority level
IP0.2	PX1	External interrupt 1 priority level
IP0.1	PT0	Timer 0 interrupt priority level
IP0.0	PX0	External interrupt 0 priority level

IP1 (F8H)

7	6	5	4	3	2	1	0
PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0

Bit	Symbol	Function
IP1.7	PT2	T2 overflow interrupt(s) priority level
IP1.6	PCM2	T2 comparator 2 priority interrupt level
IP1.5	PCM1	T2 comparator 1 priority interrupt level
IP1.4	PCM0	T2 comparator 0 priority interrupt level
IP1.3	PCT3	T2 capture register 3 priority interrupt level
IP1.2	PCT2	T2 capture register 2 priority interrupt level
IP1.1	PCT1	T2 capture register 1 priority interrupt level
IP1.0	PCT0	T2 capture register 0 priority interrupt level

Where: 0 = interrupt disabled.
1 = interrupt enabled.

Table 10 shows the interrupt vectors. The vector indicates the Program memory location where the appropriate interrupt service routine starts.

Table 10. Interrupt Vectors

SOURCE		VECTOR
External 0	X0	0003H
Timer 0 overflow	T0	000BH
External 1	X1	0013H
Timer 1 overflow	T1	001BH
Serial I/O 0 (UART)	S0	0023H
Serial I/O 1 (CAN)	S1	002BH
T2 capture 0	CT0	0033H
T2 capture 1	CT1	003BH
T2 capture 2	CT2	0043H
T2 capture 3	CT3	004BH
ADC completion	ADC	0053H
T2 compare 0	CM0	005BH
T2 compare 1	CM1	0063H
T2 compare 2	CM2	006BH
T2 overflow	T2	0073H

Interrupt Priority

Each interrupt source can be either high priority or low priority. If both priorities are requested simultaneously, the processor will branch to the high priority vector. If there are simultaneous requests from sources of the same priority, then interrupts will be serviced in the following order: X0, S1, ADC, T0, CT0, CM0, X1, CT1, CM1, T1, CT2, CM2, S0, CT3, T2.

A low priority interrupt routine can only be interrupted by a high priority interrupt. A high priority interrupt routine cannot be interrupted.

Table 11. Status of External Pins During Sleep, Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Port Data	Port Data	Port Data	Port Data	Port Data	High
Idle	External	1	1	Float	Port Data	Address	Port Data	Port Data	High
Power-down	Internal	0	0	Port Data	Port Data	Port Data	Port Data	Port Data	High
Power-down	External	0	0	Float	Port Data	Port Data	Port Data	Port Data	High

NOTE:
If the port pins P1.6 and P1.7 are used as the CAN transmitter outputs (CTX0 and CTX1), then during Sleep and Power-down mode these pins output a "recessive" level.

POWER REDUCTION MODES

The 8XC592 has three software-selectable modes to reduce power consumption. These are:

- Sleep mode, affecting the CAN Controller only
- Idle mode, affecting the
  - CPU ('halted)
  - Timer 2 (stopped and reset)
  - PWM0, PWM1 (reset, output = HIGH)
  - ADC (aborted if in progress)
- Power-down mode, affecting the whole 8XC592 device.

CAN Sleep Mode

In order to reduce power consumption of the P8XC592 the CAN Controller may be switched off (disconnecting the internal clock) by setting the CAN Command Register bit 4 (Sleep) HIGH. The CAN Controller leaves this Sleep mode by detecting either activity on the CAN-bus (dominant bit-level on CRX0/CXR1) or by setting the Sleep bit to LOW.

As the CPU cannot only write to the Sleep bit, but can also read it, the CAN Controller status can be determined directly.

Idle Mode

The instruction that sets PCON.0 is the last one executed in the normal operating mode before Idle mode is activated. Once in the Idle mode, the CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, RAM and all other registers maintain their data during Idle mode. The status of the external pins during Idle mode is show in Table 12.

There are three ways to terminate the Idle mode:

Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, provided that the interrupt source is active during Idle mode. After the interrupt is serviced, the program continues with the instruction immediately after the one, at which the interrupt request was detected.

The flag bits GF0 and GF1 may be used to determine whether the interrupt was received during normal execution or during the Idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When Idle mode is terminated by an interrupt, the service routine can examine the status of the flag bits.

Another way of terminating the Idle mode is an external hardware reset. Since the oscillator is still running, the reset signal is required to be active only for 2 machine cycles (24 oscillator periods) to complete the reset operation.

The third way is the internally generated watchdog reset after an overflow of Timer 3.

Power-down Mode

The instruction that sets PCON.1 to HIGH, is the last one executed before entering the Power-down mode. In Power-down mode the oscillator of the P8XC592 is stopped. If the CAN Controller is in use, it is recommended to set it into Sleep mode before entering Power-down mode. However, setting PCON.1 to HIGH also sets the Sleep bit (CAN Controller Command Register bit 4) to HIGH.

The P8XC592 leaves Power-down mode either by a hardware reset or by a CAN Wake-up interrupt (due to activity on the CAN bus), if the SI01 (CAN) interrupt source is enabled (contents of register IEN0 = 1x1xxxxxb). A hardware reset affects the whole P8XC592, but leaves the contents of the on-chip RAM unchanged (CAN Controller and CPU's Special Function Registers are reset). A CAN Wake-up interrupt during Power-down mode causes a reset output pulse with a width of 6144 machine cycles (4.6ms with f_{CLK}=16MHz). All hardware except from the CAN Controller of the P8XC592 is reset (the contents of all CAN Controller registers are preserved).

Note that a capacitance connected to the RST pin could lengthen the internally generated reset pulse. If the pulse exceeds 8192 machine cycles, the CAN Controller part is reset too.

RESET

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods). The CPU responds by executing an internal reset. During reset ALE and PSEN output at a HIGH level. In order to perform a correct reset, this level must not be affected by external elements.

Also with the P8XC592, the RST line can be pulled HIGH internally by a pull-up transistor activated by the watchdog timer T3. The length of the output pulse from T3 is 3 machine cycles. A pulse of such short duration is necessary in order to recover from a processor or system fault as fast as possible.

During Power-down a reset could be generated internally via the CAN Wake-up interrupt. Then the RST pin is pulled HIGH for 6144 machine cycles. In this case the CAN Controller is not reset.

If the watchdog timer or the CAN Wake-up interrupt is used to reset external devices, the usual capacitor arrangement for power-on reset should not be used. However, the internal reset is forced, independent of the external level on the RST pin.

The Main RAM and AuxRAM are not affected. When VDD is turned on, the RAM content is indeterminate.

Table 12.

After a reset the internal registers have the following contents:

	REGISTER	7	6	5	4	3	2	1	0
(CPU PART)	ACC	0	0	0	0	0	0	0	0
	ADCO	X	X	0	0	0	0	0	0
	ADCH	X	X	X	X	X	X	X	X
	B	0	0	0	0	0	0	0	0
	CML0 – CML2	0	0	0	0	0	0	0	0
	CMH0 – CMH2	0	0	0	0	0	0	0	0
	CTCON	0	0	0	0	0	0	0	0
	CTL0 – CTL3	X	X	X	X	X	X	X	X
	CTH0 – CTH3	X	X	X	X	X	X	X	X
	DPL	0	0	0	0	0	0	0	0
	DPH	0	0	0	0	0	0	0	0
	IEN0	0	0	0	0	0	0	0	0
	IEN1	0	0	0	0	0	0	0	0
	IP0	X	0	0	0	0	0	0	0
	IP1	0	0	0	0	0	0	0	0
	PCH	0	0	0	0	0	0	0	0
	PCL	0	0	0	0	0	0	0	0
	PCON	0	X	X	0	0	0	0	0
	PSW	0	0	0	0	0	0	0	0
	PWM0	0	0	0	0	0	0	0	0
	PWM1	0	0	0	0	0	0	0	0
	PWMP	0	0	0	0	0	0	0	0
	P0 – P4	1	1	1	1	1	1	1	1
	P5	X	X	X	X	X	X	X	X
	RTE	0	0	0	0	0	0	0	0
	SOBUF	X	X	X	X	X	X	X	X
	SOCON	0	0	0	0	0	0	0	0
	CANSTA	0	0	0	0	1	1	0	0
	CANCON	X	X	X	0	0	0	0	0
	CANDAT	X	X	X	X	X	X	X	X
	CANADR	0	X	1	0	0	1	0	0
	SP	0	0	0	0	0	1	1	1
	STE	1	1	0	0	0	0	0	0
	TCON	0	0	0	0	0	0	0	0
	TH0, TH1	0	0	0	0	0	0	0	0
	TMH2	0	0	0	0	0	0	0	0
	TL0, TL1	0	0	0	0	0	0	0	0
	TML2	0	0	0	0	0	0	0	0
	TMOD	0	0	0	0	0	0	0	0
	TM2CON	0	0	0	0	0	0	0	0
	TM2IR	0	0	0	0	0	0	0	0
	T3	0	0	0	0	0	0	0	0
(CAN PART)	REGISTER	7	6	5	4	3	2	1	0
	CR	0	X	1	X	X	X	X	1
	CMR	1	1	X	0	X	X	X	X
	SR	0	0	0	0	1	1	0	0
	IR	X	X	X	0	0	0	0	0
	ACR	X	X	X	X	X	X	X	X
	AMR	X	X	X	X	X	X	X	X
	BTR 0	X	X	X	X	X	X	X	X
	BTR 1	X	X	X	X	X	X	X	X
	OCR	X	X	X	X	X	X	X	X
	TR	X	X	X	X	X	X	X	X
	TXB 10 – 19	X	X	X	X	X	X	X	X
	RXB 20 – 29	X	X	X	X	X	X	X	X

NOTE:
X = Undefined State

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING		UNIT
	MIN	MAX	
Storage temperature range	–65	+150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS}	–0.5	+13	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0		W
Voltage on V_{DD} pin	–0.5	+6.5	V
Input voltage on any pin except from CTX0, CTX1, CRX0 and CRX1	–0.5	V_{DD}+0.5	V
Input output current on any I/O pin except fro CTX0 and CTX1	–	10	mA
Sink current of CTX0, CTX1 together	–	30	mA
Source current of CTX0, CTX1 together	–	20	mA

NOTES:
1. Stresses 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 conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C (83C592FFA, 87C592EFA), T_{amb} = –40°C to +125°C (83C592FHA); V_{DD}, AV_{DD} = 5V ±10%, V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL	MAX	
V_{DD}	Supply voltage		4.5		5.5	V
	Supply current:					
I_{DD}	operating	f_{CLK} = 16MHz	–		50	mA
I_{ID}	Idle mode	f_{CLK} = 16MHz	–		15	mA
I_{IS}	Idle and Sleep mode	f_{CLK} = 16MHz	–		10	mA
I_{PD}	Power-down mode (83C592 FHA)	Note 4			150	µA
I_{PD}	Power-down mode (8XC592 xFx)	Note 4			100	µA
Inputs						
V_{IL}	Input low voltage, except $\overline{EA}$, CRX0, CRX1		–0.5		$0.2V_{DD}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5		$0.2V_{DD}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST, CRX0, CRX1		$0.2V_{DD}$+0.9		V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$		V_{DD}+0.5	V
$-I_{IL}$	Input current Low, ports 1, 2, 3, 4	V_{IN} = 0.45V			–50	µA
$-I_{TL}$	High-to-Low transition current, ports 1, 2, 3, 4	V_I = 2.0V/0.45V			–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$, STADC, $\overline{EW}$	0.45V<V_I<V_{DD}			10	µA
$\pm I_{IL2}$	Input leakage current, port 5	0.45V<V_I<V_{DD}			1	µA
Outputs						
V_{OL}	Output low voltage, ports 1, 2, 3, 4, except P1.6, P1.7[5, 6]	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$, P1.6, P1.7[5, 6]	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4	$-I_{OH}$ = 60µA $-I_{OH}$ = 25µA $-I_{OH}$ = 10µA	2.4 $0.75V_{DD}$ $0.9V_{DD}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$)[3]	$-I_{OH}$ = 400µA $-I_{OH}$ = 150µA $-I_{OH}$ = 40µA	2.4 $0.75V_{DD}$ $0.9V_{DD}$			V V V
V_{OH2}	High level output voltage (RST)	$-I_{OH}$ = 400µA $-I_{OH}$ = 120µA	2.4 $0.8V_{DD}$			V
R_{RST}	Internal reset pull-down resistor		50		150	kΩ
C_{IO}	I/O buffer pin capacitance	Test freq = 1MHz, T_{amb} = 25°C			10	pF

DC ELECTRICAL CHARACTERISTICS (Continued)

T_{amb} = –40°C to +85°C (83C592FFA, 87C592EFA), T_{amb} = –40°C to +125°C (83C592FHA); V_{DD}, AV_{DD} = 5V ±10%, V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS TYPICAL	LIMITS MAX	UNIT
Analog Inputs						
AV_{DD}	Analog supply voltage[7]	$AV_{DD} = V_{DD}$±0.2V	4.5		5.5	V
AI_{DD}	Analog supply current: Operating Idle mode	Port 5 = AV_{DD}			 2.5 2.5	 mA mA
AI_{IS}	Idle and sleep mode 83C592FHA				400	µA
	8XC592xFx				350	µA
AI_{PD}	Power-down mode 83C592FHA				400	µA
	8XC592xFx				350	µA
AV_{IN}	Analog input voltage		AV_{SS}–0.2		AV_{DD}+0.2	V
AV_{REF}	Reference voltage: AV_{REF-} AV_{REF+}		 AV_{SS}–0.2		 AV_{DD}+0.2	 V V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		10		50	kΩ
C_{IA}	Analog input capacitance				15	pF
t_{ADS}	Sampling time				$8t_{CY}$	µs
t_{ADC}	Conversion time (including sampling time)				$50t_{CY}$	µs
DL_e	Differential non-linearity[8, 9, 10]				±1	LSB
IL_e	Integral non-linearity[8, 11]				±2	LSB
OS_e	Offset error[8, 12]				±2	LSB
G_e	Gain error[8, 13]				±0.4	%
A_e	Absolute Voltage Error[8, 14]				±3	LSB
M_{CTC}	Channel to channel matching				±1	LSB
C_I	Crosstalk between Port 5 inputs[9]	0–100kHz			–60	dB
CAN						
	CAN input comparator (CRX0, CRX1)	AV_{DD} = 5V ± 5% 1.4V<V_I<AV_{DD}–1.4V				
$±V_{DIF}$	Differential input voltage[15]		32		–	mV
V_{HYST}	Hysteresis voltage[15]		8		30	mV
$±I_I$	Input current		–		400	nA
	CAN output driver	V_{DD} = 5V ± 5%				
V_{OLT}	CTX0, CTX1, output voltage LOW	I_O = 1.2mA[15] I_O = 10mA	– –		0.1 0.6	V V
V_{OHT}	CTX0, CTX1, output voltage HIGH	I_O = 1.2mA[15] I_O = –10mA[16]	V_{DD} – 0.1 V_{DD} – 0.6		– –	V V
Reference						
	Reference	AV_{DD} = 5V ± 5%				
V_{REFOUT}	REF output voltage[15] (bit Reference Active = HIGH)	–0.1mA<I_L<0.1mA; C_L = 10nF	AV_{DD}/2–0.05		AV_{DD}/2+0.05	V
$±I_{REFIN}$	REF input current (bit Reference Active = LOW)	1.5V<V_{REFIN}< AV_{DD}–1.5V	–		10	µA

NOTES ON NEXT PAGE.

NOTES TO THE DC ECLECTRICAL CHARACTERISTICS:

1. The operating current is measured with all output pins unloaded; XTAL1 is driven with $t_R = t_F = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = $\overline{EW}$ = V_{DD}; STADC = V_{SS}; CRX0 = 2.7V; CRX1 = 2.3V.
2. The idle mode supply current is measured with all output pins unloaded; XTAL1 is driven with $t_R = t_F = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; Port 0 = P1.6 = P1.7 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = V_{SS}; CRX0 = 2.7V; CRX1 = 2.3V.
3. The idle and sleep mode supply current is measured with all output pins unloaded; XTAL1 is driven with $t_R = t_F = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; Port 0 = P1.6 = P1.7 = $\overline{EW}$ = CRX0 = V_{DD}; $\overline{EA}$ = RST = STADC = CRX1 = V_{SS}; CAN: register 6: = 00H, register 7: = 12H, register 8: = 02H, register 0: = 20H, wait $15t_{CY}$, register 1: = 10H, wait for bit Sleep = 1.
4. The power-down current is measured with all output pins unloaded; Port 0 = P1.6 = P1.7 = $\overline{EW}$ = CRX0 = V_{DD}; $\overline{XTAL1}$ = $\overline{EA}$ = RST = STADC = CRX1 = V_{SS}. Windowed devices must have the window covered during testing.
5. Under steady state (non-transient) conditions, I_{OL} must be limited externally as follows:

   Maximum I_{OL} per 8 bit port
   - Port 0: 26mA
   - Port 1: 32mA
   - Ports 2, 3, and 4: 15mA

   Maximum I_{OL} for all output pins: 71mA

   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
6. Capacitive loads on Port0 and Port2 may degrade the LOW level output voltage of ALE, Port1 and Port2. During a 1-to-0 transition on the Port0 and Port2 pins and a capacitive load > 100pF, the ALE LOW level may exceed 0.8V. In that case, it is necessary to connect ALE to a Schmitt trigger input respectively use an address latch with a Schmitt trigger STROBE input.
7. Capacitive loads on Port0 and Port2 may cause a HIGH level output voltage degradation of ALE and $\overline{PSEN}$ below 0.9 V_{DD} during the address bits are stabilizing.
8. $AV_{REF+} = 5.12V$; $AV_{REF-} = 0V$; $AV_{DD} = 5.0V$.
9. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width.
10. The ADC is monotonic, there are no missing codes.
11. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error.
12. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve after removing gain error, and a straight line which fits the ideal transfer curve. The offset error is constant at every point of the actual transfer curve.
13. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve after removing offset error and the straight line which fits the ideal transfer curve. The gain error is constant at every point on the transfer curve.
14. The absolute voltage error (A_e) is the maximum difference between the center of the steps of the actual transfer curve of the not calibrated ADC and the ideal transfer curve.
15. Not tested during production.
16. Source current for the CTX0, CTX1 outputs together.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{DD}, AV_{DD} = 5V ±10%, V_{SS}, AV_{SS} = 0V

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK		UNIT
			MIN	MAX	
$1/t_{CLK}$	7	Oscillator frequency	1.2	16	MHz
t_{LHLL}	7	ALE pulse width	$2t_{CLK}$–40		ns
t_{AVLL}	7	Address valid to ALE low	t_{CLK}–55		ns
t_{LLAX}	7	Address hold after ALE low	t_{CLK}–35		ns
t_{LLIV}	7	ALE low to valid instruction in		$4t_{CLK}$–100	ns
t_{LLPL}	7	ALE low to $\overline{PSEN}$ low	t_{CLK}–40		ns
t_{PLPH}	7	$\overline{PSEN}$ pulse width	$3t_{CLK}$–45		ns
t_{PLIV}	7	$\overline{PSEN}$ low to valid instruction in		$3t_{CLK}$–105	ns
t_{PXIX}	7	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	7	Input instruction float after $\overline{PSEN}$		t_{CLK}–25	ns
t_{AVIV}	7	Address to valid instruction in		$5t_{CLK}$–105	ns
t_{PLAZ}	7	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{AVLL}	8, 9	Address valid to ALE low	t_{CLK}–55		ns
t_{RLRH}	8, 9	$\overline{RD}$ pulse width	$6t_{CLK}$–100		ns
t_{WLWH}	8, 9	$\overline{WR}$ pulse width	$6t_{CLK}$–100		ns
t_{RLDV}	8, 9	$\overline{RD}$ low to valid data in		$5t_{CLK}$–165	ns
t_{RHDX}	8, 9	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	8, 9	Data float after $\overline{RD}$		$2t_{CLK}$–70	ns
t_{LLDV}	8, 9	ALE low to valid data in		$8t_{CLK}$–150	ns
t_{AVDV}	8, 9	Address to valid data in		$9t_{CLK}$–165	ns
t_{LLWL}	8, 9	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLK}$–50	$3t_{CLK}$+50	ns
t_{AVWL}	8, 9	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLK}$–130		ns
t_{QVWX}	8, 9	Data valid to $\overline{WR}$ transition	t_{CLK}–60		ns
t_{WHQX}	8, 9	Data hold after $\overline{WR}$	t_{CLK}–50		ns
t_{RLAZ}	8, 9	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	8, 9	$\overline{RD}$ or $\overline{WR}$ high to ALE high	t_{CLK}–40	t_{CLK}+40	ns
External Clock					
t_{CHCX}	11	High time[3]	20		ns
t_{CLCX}	11	Low time[3]	20		ns
t_{CLCH}	11	Rise time[3]		20	ns
t_{CHCL}	11	Fall time[3]		20	ns
UART Timing in Shift Register Mode					
t_{XLXL}	10	Serial port clock cycle time[3]	$12t_{CLK}$		µs
t_{QVXH}	10	Output data setup to clock rising edge	$10t_{CLK}$–133		ns
t_{XHQX}	10	Output data hold after clock rising edge	$2t_{CLK}$–117		ns
t_{XHDX}	10	Input data hold after clock rising edge	0		ns
t_{XHDV}	10	Clock rising edge to input data valid		$10t_{CLK}$–133	ns
CAN Input Comparator / Output Driver (AV_{DD} = 5V ± 5%)					
t_{SD}		Sum of input and output delay (V_{DIF} = ± 32mV; 1.4V < V_I < AV_{DD} – 1.4V) AV_{DD} = 5V ± 5%	–	60	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. the designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

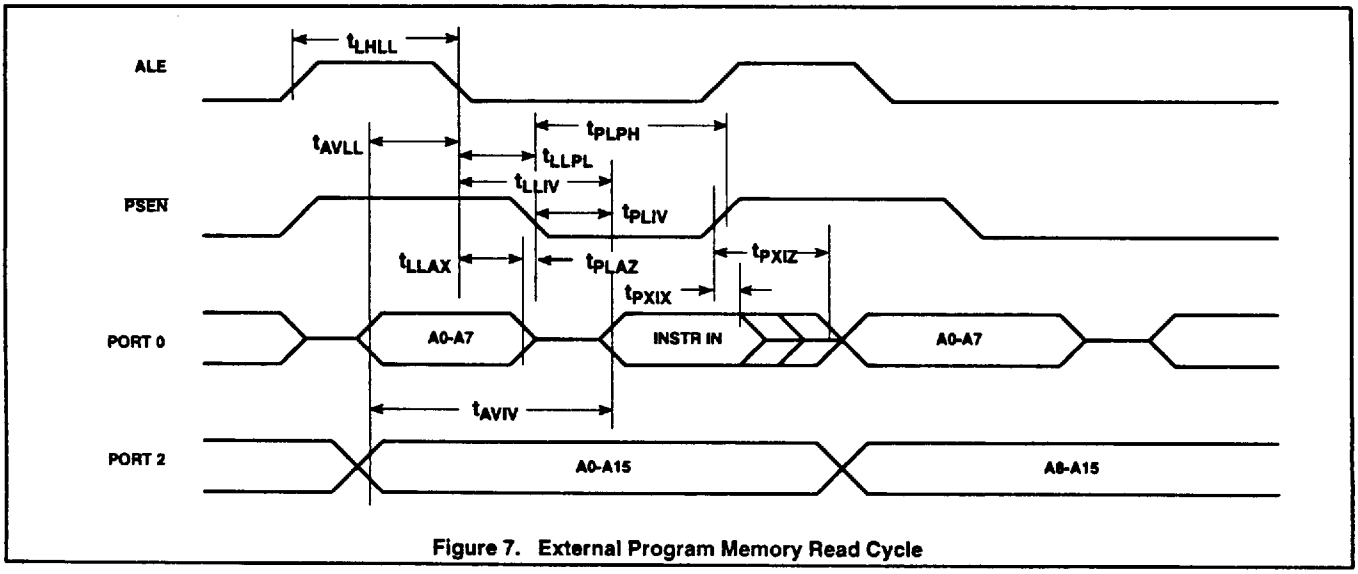


Figure 7. External Program Memory Read Cycle

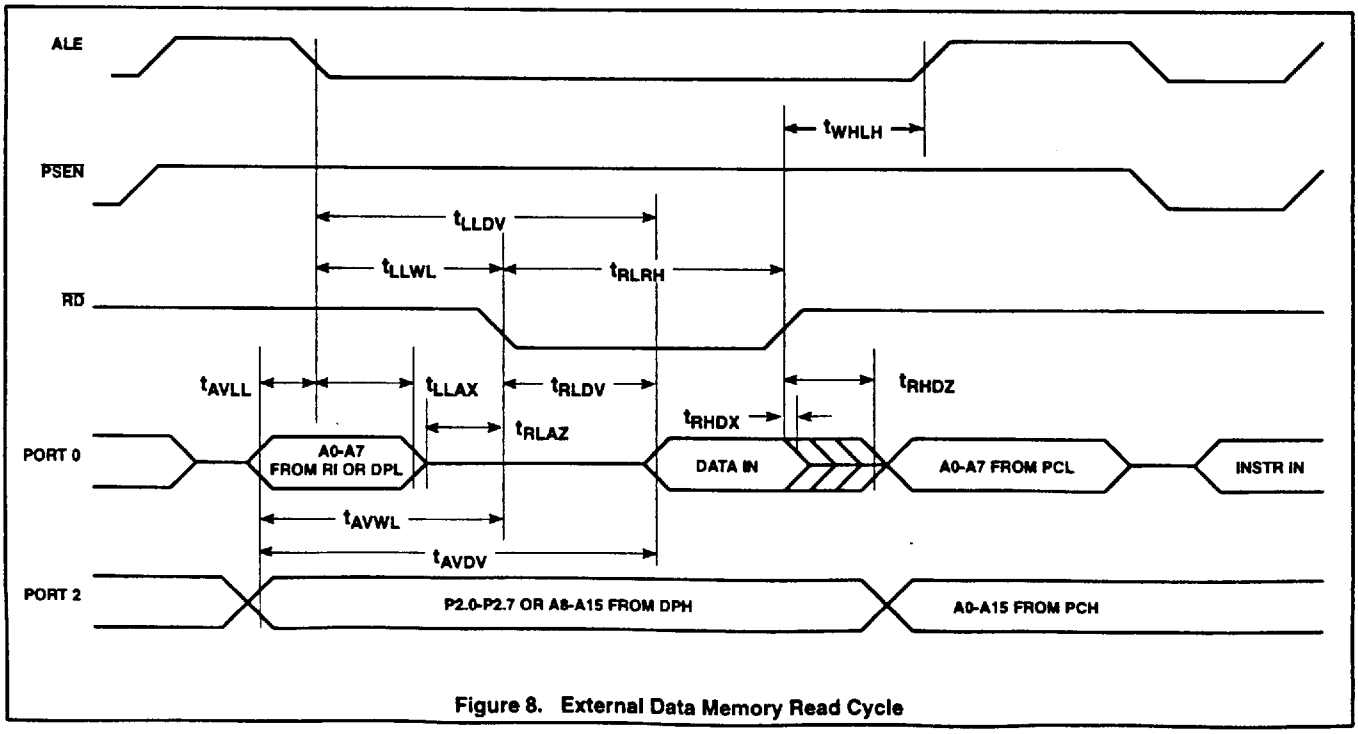


Figure 8. External Data Memory Read Cycle

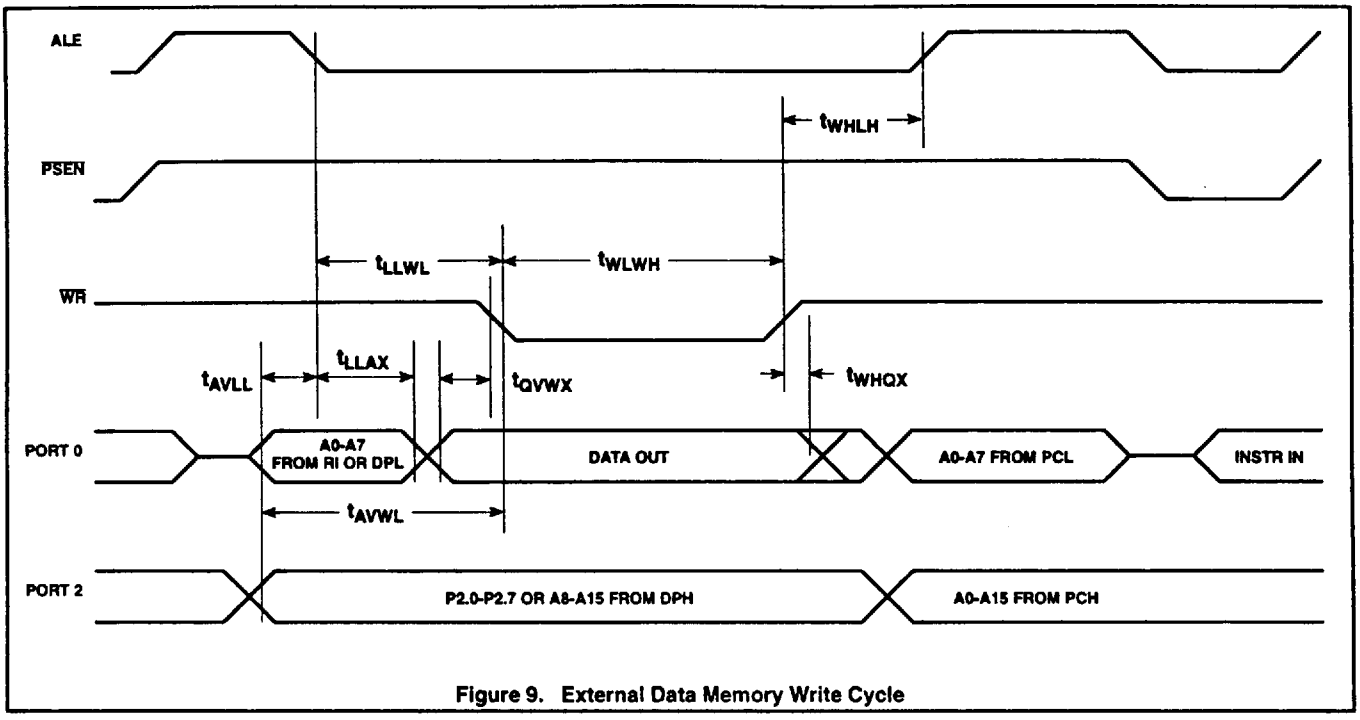


Figure 9. External Data Memory Write Cycle

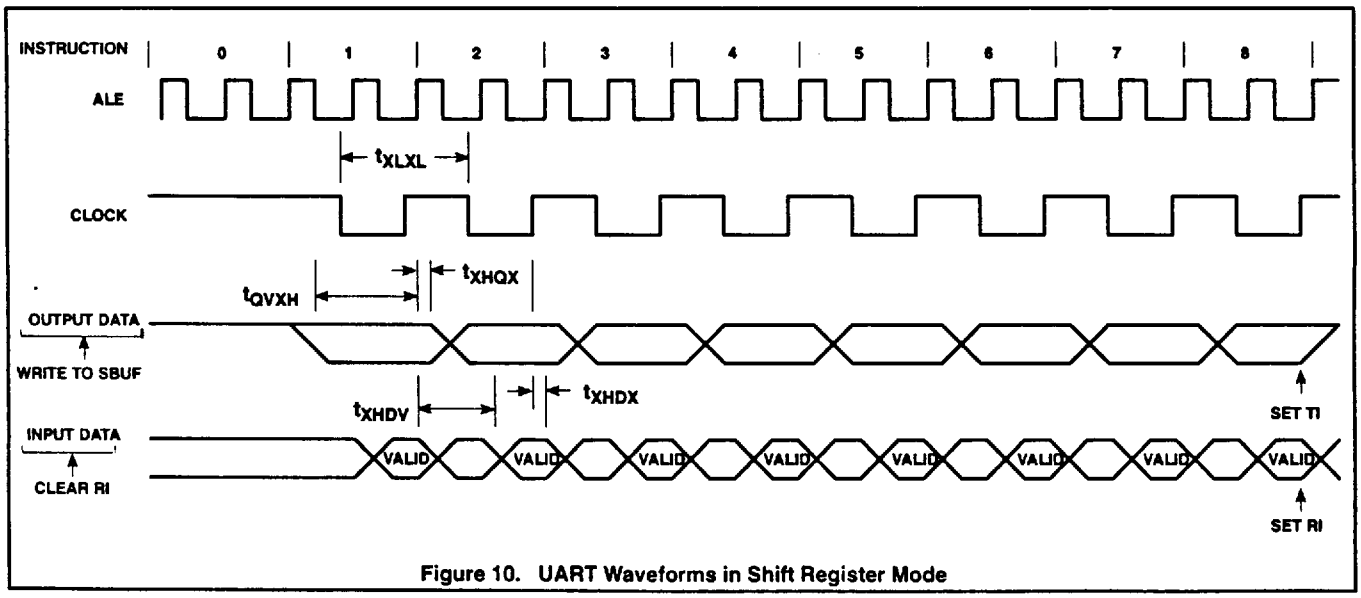


Figure 10. UART Waveforms in Shift Register Mode

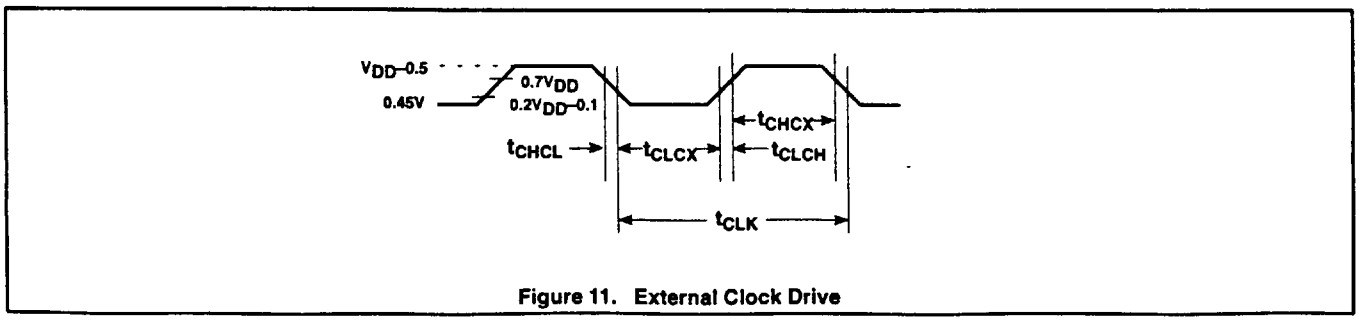
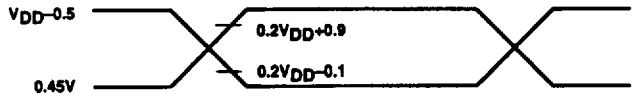


Figure 11. External Clock Drive

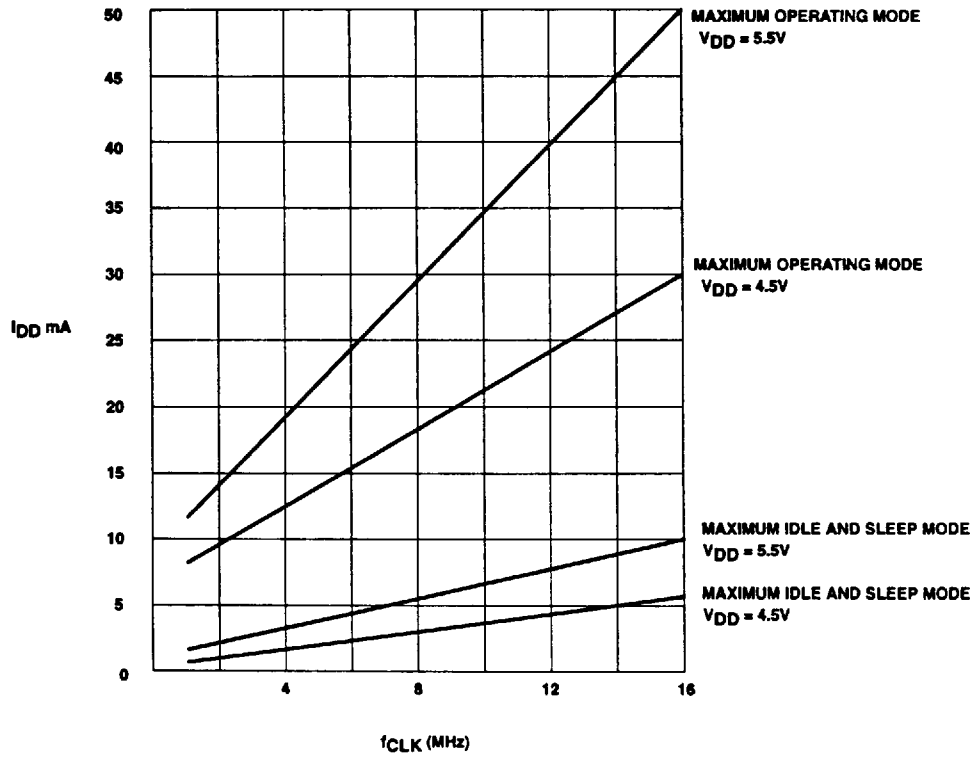


NOTE:
AC inputs during testing are driven at V_{DD} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} max for a logic '0'.

Figure 12. AC Testing Input/Output

NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20mA$.

Figure 13. Float Waveform



NOTE:
Valid only within frequency specifications of the device under test. Maximum values taken at worst case temperature.

Figure 14. I_{DD} vs. FREQ

EPROM CHARACTERISTICS

The 87C592 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C592 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C592 manufactured by Philips.

Table 13 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 15 and 16. Figure 17 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 15. Note that the 87C592 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 15. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$, and pins of ports 2 and 3 specified in Table 13 are held at the "Program Code Data" levels indicated in Table 13. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 16.

To program the encryption table, repeat the 25-pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table" levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25-pulse programming sequence using the "Pgm Lock Bit" levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be red is applied to ports 1 and 2 as shown in Figure 17. The other pins are held at the "Verify Code Data" levels indicated in Table 13. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 9CH indicates 87C592

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 13, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to the light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345-5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 13. EPROM Programming Modes

MODE	RST	PSEN	ALE/PROG	EA/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. 0 = Valid low for that pin; 1 = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{DD} = 5V ±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

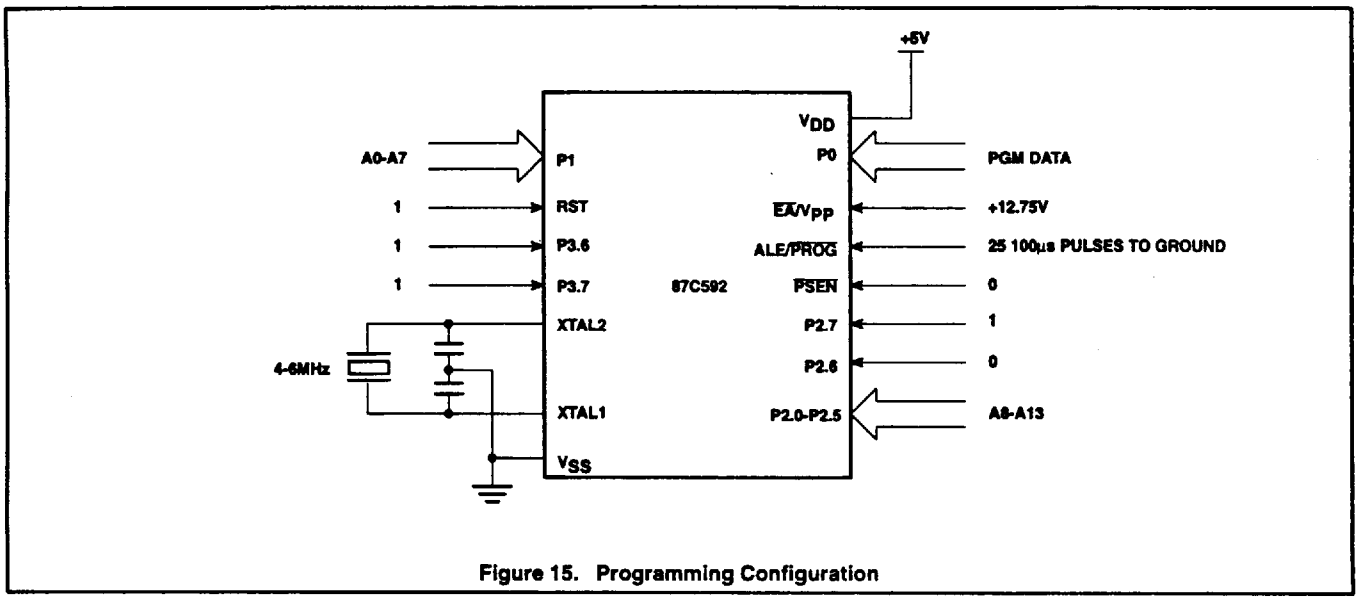


Figure 15. Programming Configuration

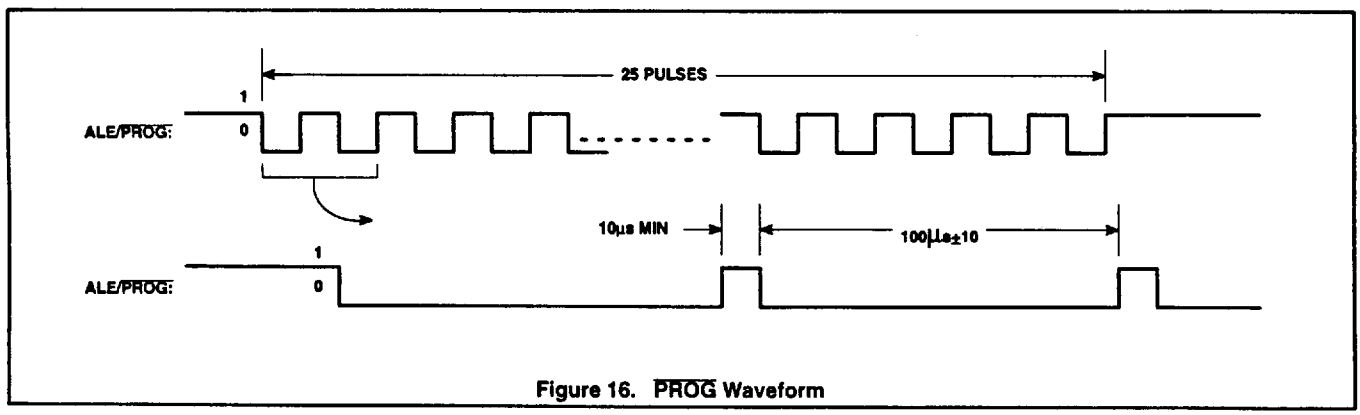


Figure 16. PROG Waveform

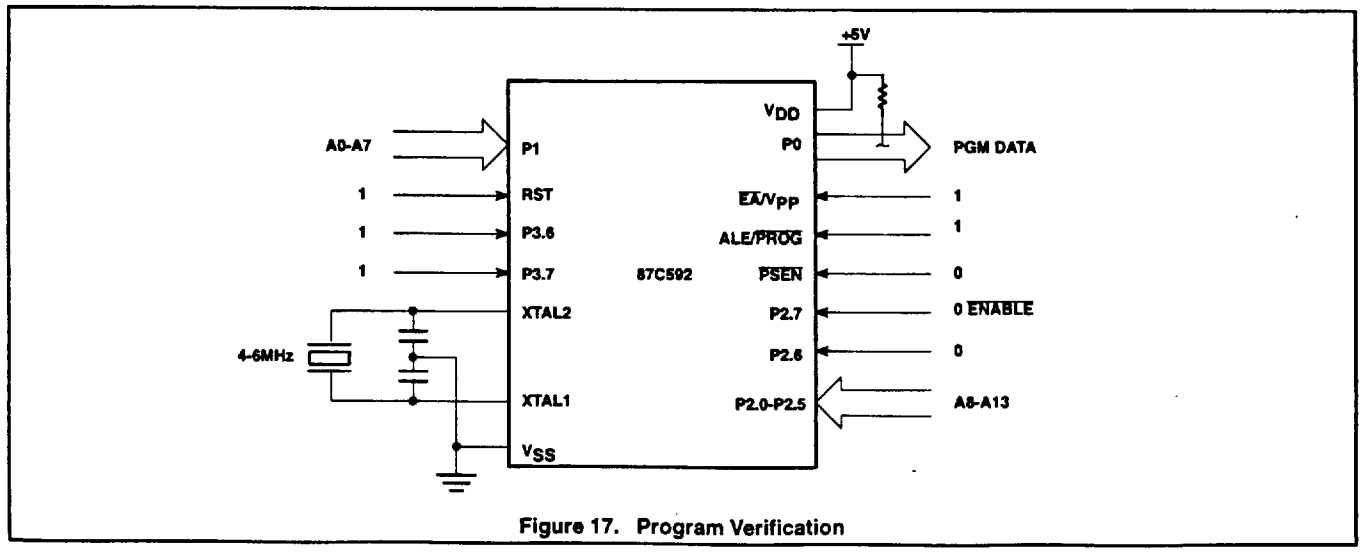


Figure 17. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{DD} = 5V±10%, V_{SS} = 0V (See Figure 18)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLK}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLK}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLK}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLK}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLK}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLK}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLK}$	
t_{ELQV}	$\overline{ENABLE}$ low to data valid		$48t_{CLK}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLK}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

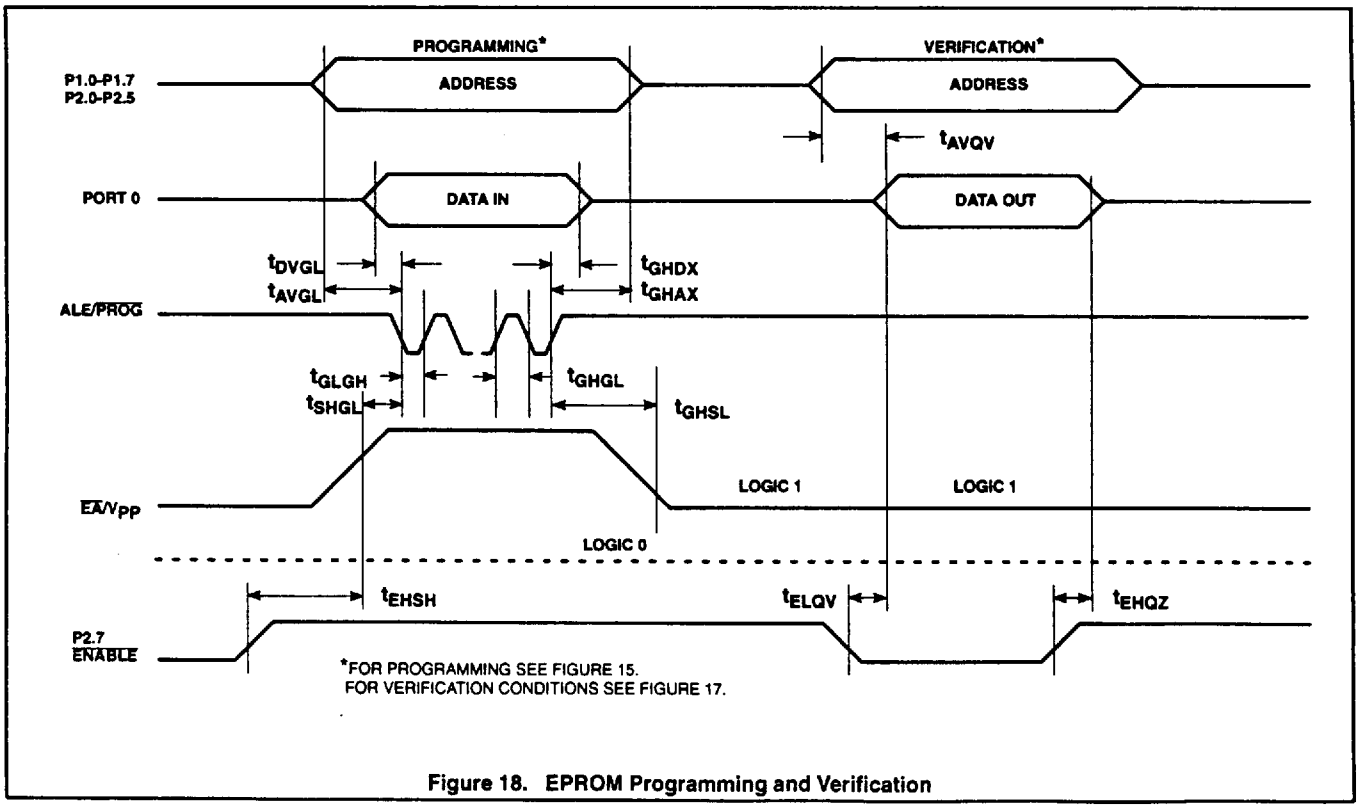


Figure 18. EPROM Programming and Verification