Features

- Designed in Accordance with CAN Specification 2.0B
- CAN 2.0B Protocol Functions
- 4 Different Data Rates Using an Internal Programmable Prescaler
  - 1 Mbit/s, 500 kbit/s, 250 kbit/s, 125 kbit/s
- 16 MHz Cycle Frequency
- Power-down: Sleep Mode
- CAN Bus Line Arbitration
- All Required CAN Functions:
- Error Handling:
  - Stuff Bit Generation
  - CRC Generation
  - Acknowledge Generation
  - Remote Frame
- 1 TX Buffer and 3 RX Buffer
- 3 Individual Acceptance Filtering
- Interrupt Outputs Can be Generated for the Following Events
  - Telegram Sent Successfully
  - Telegram Received Successfully
  - Receive Buffer Overflow
  - Bus Off Condition
  - Error Passive Condition
- Technology: Atmel MG2RTP Radiation Hardened Sea of Gate 0.5 um
- Type: Semi-Custom Digital 5V
- Operating Frequency: 16 MHz (Test Frequency is 10MHz)
- Maximum Frequency: 18 MHz
- No Single Event Latch-up below a LET Threshold of 80 MeV/mg/cm^2
- Tested up to a Total Dose of 300 Krads(si) According to MIL STD 883 Method 1019
- QML-Q and V with SMD 5962-03A06
- Package: MLCC 44 pins

Description

The AT7908E is a CAN controller stand-alone device for space application. Redundant structures and special techniques are implemented in order to make the device SEU tolerant. The AT7908E CAN core provides all CAN 2.0B protocol functions except the overload frame generation. It includes the acceptance filtering. The core incorporates error-handling capabilities, the stuff bit generation, CRC, multiple sample points and remote frame generation. The AT7908E provides a programmable MCU 8-bit general-purpose interface to connect receive and transmit buffer, control register and status register to CPU.

The AT7908E was designed by Aurelia Microelettronica S.p.A., Italy, and the chip is known as CASA (CAN ASIC for Space Application).

CAN Controller for Space Application

AT7908E

4268D–AERO–11/09

Signal Pins Description

Pin Number	Signal Name	Type	Note	Description
9	Mode	I - CMOS		Interface operational mode
8	Cs	I - CMOS	AH	Chip select signal
11	Wr	I - CMOS	AL	Write signal
13	Rd	I - CMOS	AL	Read signal
10	ALE	I - CMOS	AH	Address latch enable
23	Xtalin	I - CMOS		Input to internal oscillators or clock input from external oscillator
22	Xtalout	IO - CMOS		Output from internal oscillator
16	Reset	I - CMOS	AL	reset signal
5, 4, 3, 2, 44, 43, 42, 41	addr<7:0>	I - CMOS		Input address(mode1) or output address(mode 0)
38, 37, 36, 35, 33, 32, 31, 30	Data<7:0>	IO - CMOS		Address data bus
25	Int	O - CMOS	AL	interrupt request
27	Can_tx	O - CMOS		tx signal
19	Can_rx	I - CMOS		rx signal
14	sena	I - CMOS	AH	scan enable
15	test	I - CMOS	AH	input signal to increase testability
26	hatrig	O - CMOS	AH	Output signal to trigger the message matching
20	hasync	O - CMOS	AH	Output synchronization signal

Note: Abbreviations: O = output, I = input, IO = bi-directional I/O, AL = Active Low, AH = Active High.

Oscillator

Xtalin and Xtalout are IOs of an internal inverting oscillator. To use the internal oscillator, the quartz must be connected between Xtalin and Xtalout pins.

To drive the device with an external clock source, Xtalin must be driven by clock signal and Xtalout must be left unconnected. The maximum operating frequency of the oscillator, in open-loop mode(without crystal) with 6 pF load on Xtalout at worst condition (Process slow, temperature = 145°C, Power supply = 4.5V) is 60 MHz.

The IO level of Xtalin and Xtalout are CMOS levels.

The internal clock could be put on off condition with external pins SENA = logical value 1 and TEST = logical value 0.

Reset Specification

The reset pin must be driven low for at least 3 clock cycles (190 ns at 16 MHz).

The reset signal must be driven low at power on of the AT7908E for at least 200 ns to avoid abnormal start condition of the AT7908E device.

Electrical Characteristics

Absolute Maximum Ratings

Ambient temperature under bias (TA)... Military –55 to +125°C Junction temperature .....175°C Storage temperature .....–65 to +150°C Power Dissipation: .....0.3W Themal Resistance Junction to Case: .....5.1°C/W TTL/CMOS : Supply voltage ..... VDD –0.5V to +6V I/O voltage ..... –0.5V to VDD + 0.5V	*NOTICE: Stresses above those listed may cause permanent damages to the device. Exposure to absolute maximum rating conditions for extended period may affect device reliability.

DC Characteristics

Specified at VDD = +5V +/– 10%

Symbol	Parameter	Min	Typ	Max	Unit	Conditions
VIL	Input LOW voltage CMOS input	0		0.3VDD	V	
VIH	Input HIGH voltage CMOS input	0.7VDD		VDD	V	
VOL	Output LOW voltage CMOS Output			0.4	V	IOL = +6 mA (1)
VOH	Output HIGH voltage CMOS Output	3.9			V	IOH = -6 mA (1)
IL	Input Leakage current NO Pull up/down		+/- 1	+/- 5	mA	
IOZ	3-State Output Leakage current		+/-1	+/-5	mA	
IOS	Output Short circuit current IOSN IOSP			 24 18	 mA mA	BOUT6 VOUT = 4.5V VOUT = VSS

Note: 1. According to the following buffers: BOUT6, BIOC6 @ VDD = 4.5V

AC Characteristics

Tj = 125 °C, Process typical (all values in ns)

Output Signals	Buffer Description	Load	Tp Xtalin to Output High	Tp Xtalin to Output Low	Set Up or Hold
Can_tx	BOUT6: Output buffer with 6 mA drive	75 pF	29	19	
Int					
Hasync			25		
Hatrig			26		
Set up Data<7:0>/Ale	BIOC6: Bi-directional Buffer CMOS input	75 pF			5
Hold Data<7:0>/Ale					6

Power Consumption

Parameter	Max	Min	Note
DC Curent Dissipation	50 mA @ 5V	-	CLK frequency = 16 MHz

Technology

MG2RTP 0.5 mm 3 Metal Layers Sea of Gate.

Matrix: MG2 –044: 33K usable gates.

Package: MLCC_J44: Ceramic Multi layer Package

Application

The core architecture has been implemented taking into account the typical constraints of a space application:

- No Single Event Latch-up below a LET Threshold of 80 MeV/mg/cm^2
- Tested up to a Total Dose of 300 Krads(si) According to MIL STD 883 Method 1019
- Design using the SEU hardened flip flops
- Sleep mode for low power consumption
- Insertion of test structures (internal scan chain) to reach a fault coverage > 95% according to ESA specification

Functional Description

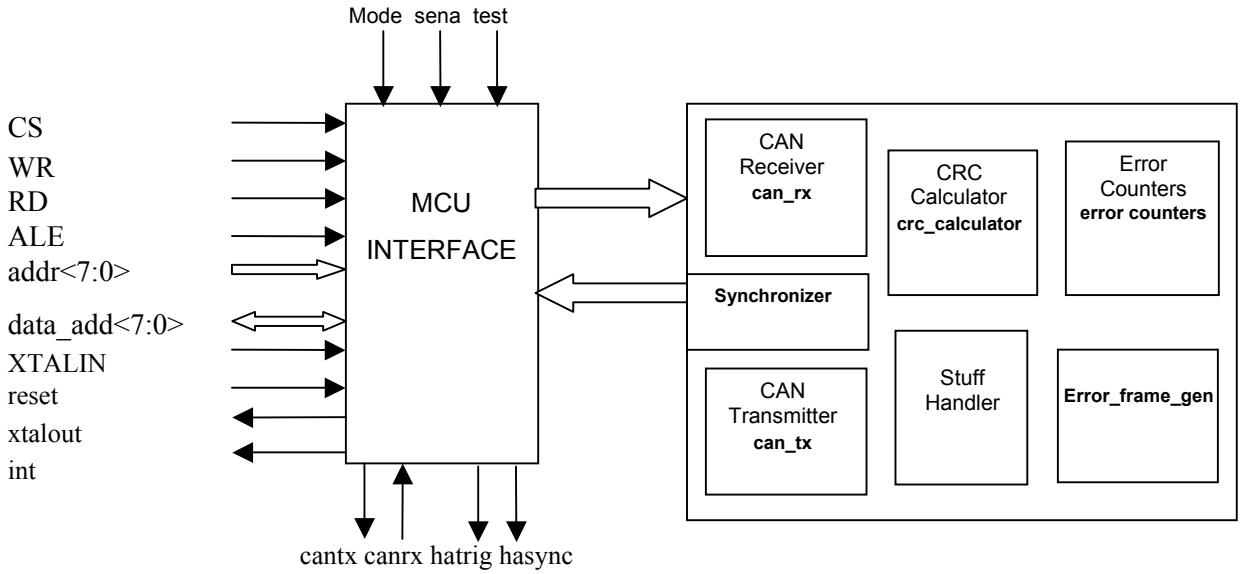
The AT7908E is an integrated device that performs serial communications according to the CAN protocol. The CAN protocol uses a multi-master bus configuration to transfer data packets between nodes on a network. It supports both, standard and extended, message frame formats as CAN Specification 2.0B. It can transmit and receive with 29 identifier bits and it can filter the first eleven bits of the receiving message (the filtering function is performed only on eleven bits).

The AT7908E has one transmit buffer and three receiving buffers. The filtering function is individual for every RX message buffer. Every RX message buffer is formed by an identifier (29 bits long), by the data and by the status register that store the status of the receiver buffer. The identifier permits to filter the receiving message together with the global mask that implements the don't care condition. A message is accepted and stored in the RX message buffer only if the identifier of the incoming message (first eleven bits) matches the identifier in the RX message buffer. If the receiving message matches more than one identifier, the message is stored in the lowest numbered message buffer.

The AT7908E implements a global acceptance-masking feature for the message filtering.

The AT7908E was developed to have a programmable general-purpose MCU interface. The MCU interface permits to program the internal register of the AT7908E and to read out the status of the controller and the received messages. The schematic representation of the AT7908E is shown in Fig. 1.

Figure 1. Block Schematic of AT7908E



To transmit a CAN message, the MCU must write in the interface internal registers the data and the configuration signals. The message bus (0 to 101) goes to **can_tx** that generates the output signal to the transceiver **can_tx** and the signals for **crc_calculator** and for **error_counters**.

The **crc_calculator** block calculates the CRC on the received message and, if the CAN is in Transmission State, sends the CRC to **can_tx**. If the CAN is receiving the message, the CRC is sent to **can_rx** (CRC<0:14>) that checks if the calculated CRC matches the received CRC.

The **error_counters** block is a counter that receives signals from **can_tx** and **can_rx** to calculate the error state of the CAN. If the number of errors is higher than 127, the AT7908E will be on the error passive state. If error counters is higher than 255, the AT7908E will be on the bus-off state.

The **synchronizer** block synchronizes the controller with the **can_rx** incoming signal and generates the internal clock and reset signals for the other blocks of the controller.

The stuff handler block handles the stuff bit generation, compliant with the standard BOSCH 2.0B specification. **error_frame_gen** block generates the error frame on the bus.

In the next sections, the interface block, the internal registers and the operational modes of the AT7908E will be described.

Transmission of Message

To transmit a message on the bus, the controller must program the configuration registers of the AT7908E (interrupt generator configuration, data rate, bit timing configuration, message length) and the transmit buffer. After this, the controller must write in the internal register the bit (Txreq=1) to start the transmission. At this point, the Transmit message buffer is sent to the bus line. At every moment the external MCU can readout the status of the controller (error condition, bus-off condition, transmission active or transmission OK).

Incoming Message

The incoming message passes through the global mask and is checked with the identifier of the first receiver message buffer. If the message is accepted, the data will be stored in the first message buffer and the status of the receiver message objects will be updated with the information of the received message: length of message, extended or standard frame message, reception OK or overrun condition. The incoming message is stored in the message buffer at the end of the END of FRAME field (7 recessive bits).

If the incoming message is a remote frame request from another CAN node, the incoming remote request will not be stored on the RX message buffer.

Remote Frame

The AT7908E supports remote frame features. To send on the bus a remote frame request, the MCU must write two internal bits of SETUP register (TMRMR=1 and TXRM=0). The arbitration registers of the transmitter message buffer must have the identifier for the remote frame request. The start of the remote frame request will be performed when the transmission request is set by the MCU on the AT7908E. The AT7908E that receives a REMOTE FRAME request (RTR bit =recessive) that matches the identifier stored on own TX_arbitration register (filtering functionality with don't care feature) send on the bus the TX message buffer as an answer to the remote frame request. To answer the remote frame request, the AT7908E must be pre-programmed with two bits of the internal register: TMRMR=0 and TXRM=1. The TX arbitration register must be initialized with the message identifier to which to answer and the transmitter data buffer must be loaded with the data to be sent as the answer to the remote frame request. The transmission request must be set as:

Answer to remote request if

for i = 28 down to 18

("MsgID(i)" XOR" TX_arbitration (i))AND "Accept. Mask(i)" = 0

Filtering of Message

The filtering function is implemented with a global mask and the arbitration register of the receiver message buffers. The Global Mask permits to define the don't care bit on the filtering operation. The incoming message ID (msg_ID(28:18) is checked (xor) with

the identifier stored in the message buffers (RXn_ARB0, RXn_ARB1). The result is masker able (logic **AND** operation) with global mask (FILTER_AM_0, FILTER_AM_1).

Message Valid if

for i = 28 downto 18

(“MsgID(i)” XOR” RXn_arbitration (i))AND “Accept. Mask(i)” = 0

If the incoming message is filtered out from the first message buffer, the arbitration field of the incoming message is checked with the identifier of the second message buffer, and if filtered out, with the identifier of the third message buffer. The global mask feature (used to accept a set of incoming messages) permits that the incoming message could match with all the three-messages buffers but the received message is stored on the lowest numbered message buffer.

Interrupt Generation

The interrupt generation on the INT pin can be programmed by the external MCU writing the internal configuration register. Interrupt can be generated for:

- Either message correctly transmitted
- Or message correctly received (data frame, not remote frame reception).
- Or overrun condition on the rx message buffer
- Or bus-off condition
- Or error passive condition

In the SETUP register, it is possible to configure which kind of interrupt will be generated by the AT7908E device.

Fault Confinement and Error Counters

The two internal error counters are readable from the external MCU to check the error status of the CAN controller. The error counter operates according to the Fault Confinement rules of the CAN Specification 2.0B, exception included. This means that if there is only one node of the AT7908E connected on the BUS and this node starts to transmit message, it meets the error passive condition but not the bus-off condition, because the transmitter error counter will no longer be incremented for acknowledge error when it is in error passive state.

Trigger Match Function

The trigger match function is a feature implemented to generate the HATRIG signal (a pulse for a clock cycle) if the Identifier of the incoming message matches the TRIGGER_MATCH register value. The incoming message identifier(msg_id28 down to msg_id18) is checked with the TM28 down to the TM18 bits of the TRIGGER_MATCH registers. This function is independent from the filtration of message and from the interrupt generation. The HATRIG signal, together with the CAN node, regularly generating the appropriate frame, could be used to force the bus switching in a system with two physical bus used as nominal and redundant.

Detailed Pin Description

MODE	Input pin to select the operational mode of the interface: mode = 0 : 8 bits of the data bus multiplexed with the lowest 8 bits of the address(register mapped between 8000Hex and 804Chex) mode = 1 : 8 bits not multiplexed with the address data bus (register mapped between 00 Hex and 4C Hex)
CS	Chip select pin to write or read the internal register. A high level on this pin enables the CPU to access the AT7908E.
WR	Pin to write the internal register. A low level on this pin enables the writing of the AT7908E register (if CS signal is HIGH).
RD	Pin to read the internal register. A low level on this pin enables the readout of the AT7908E register (if CS signal is HIGH).
ALE	Address-latch-enable. Used in mode=0.
XTALIN	Input pin for the clock. XTALIN, with XTALOUT, are the crystal connections for the internal oscillator.
XTALOUT	Input output pin. This pin could be used as the input, together with XTALIN, for the internal oscillator or as the clock output to drive the CPU.
RESET	Reset signal for the AT7908E. To reset the AT7908E, this signal must be set low.
DATA<7:0>	Bi-directional bus: Multiplexed data/address bus in the mode 0 or data bus in the mode1.
ADDR<7:0>	Highest input address in the mode0 or lowest input address in the mode 1.
INT	Output pin for the interrupt request to the MCU. This pin is active low (interrupt generation) and will be kept low until the MCU clears the interrupt request on AT7908E.
CAN_TX	Serial output pin to the CAN transceiver (dominant='0', recessive='1')
CAN_RX	Serial input from the CAN transceiver (dominant='0', recessive='1').
SENA	Input signal to enable the scan-test (with test = 1) or to put the AT7908E on power down mode (with test signal = 0). To use the AT7908E in normal functional mode, this signal must be logical 0.
TEST	Input signal to increase the testability. This pin will be used in the test modality. Test = 1 => test mode (used by the manufacturer that executes the test of the device. Test = 0 => functional mode (if SENA =0) or power down mode (if SENA =1).
HATRIG	Output signal that will be set high (duration of the pulse = 13 clock cycles) if the received message arbitration matches the TRIGGER_MATCH register (independently from the matching between the arbitration of the incoming message and the identifier of the message buffers). This pin could be used for the bus switching in a system with two physical buses (nominal and redundant).
HASYNC	Output signal that could be used to advise that the node started to transmit a message or started to receive a message (duration of the pulse = 13 clock cycles).

Internal Register Description

Register Name	**Type R = read R/W = read/write**	**Address Hex**	**Register function (bit7… bit0)**							
SETUP_0	R/W	00	BPR1	BPR0	Gensync Tx	Gensync Rx	Errint	Overint	Rxint	Txint
SETUP_1	R/W	01	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
SETUP_2	R/W	02	PS2_3	PS2_2	PS2_1	PS2_0	PS1_3	PS1_2	PS1_1	PS1_0
SETUP_3	R/W	03	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
SETUP_RX	R/W	04	reserved	reserved	reserved	Reserved	reserved	Rxclr3	Rxclr2	Rxclr1

STATUS	R	05	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
STATUS_RX	R	06	reserved	Rxovr3	Rxovr2	Rxovr1	reserved	RXOK3	RXOK2	RXOK1
FILTER_AM_0	R/W	07	Filter Mask: AM28:AM21							
FILTER_AM_1	R/W	08	Filter Mask: AM20:AM18 & 3 reserved bit							
ERR_COUNT_TX	R	09	Transmitter error counter							
ERR_COUNT_RX	R	0A	Receiver error counter							
TRIG_MATCH_0	R/W	0B	Trigger match register TM28..TM21							
TRIG_MATCH_1	R/W	0C	Trigger match register TM20, TM19, TM18 & 5 reserved bit							
TX_ARB_0	R/W	10	Tx Arbitration register – TXARB28:TXARB21							
TX_ARB_1	R/W	11	Tx Arbitration register – TXARB20:TXARB13							
TX_ARB_2	R/W	12	Tx Arbitration register – TXARB12:TXARB5							
TX_ARB_3	R/W	13	Tx Arbitration register - TXARB4: TXARB0 & 3 reserved bit							
TX_MESSAGE_0	R/W	14	Tx Data byte 0							
TX_MESSAGE_1	R/W	15	Tx Data byte 1							
TX_MESSAGE_2	R/W	16	Tx Data byte 2							
TX_MESSAGE_3	R/W	17	Tx Data byte 3							
TX_MESSAGE_4	R/W	18	Tx Data byte 4							
TX_MESSAGE_5	R/W	19	Tx Data byte 5							
TX_MESSAGE_6	R/W	1A	Tx Data byte 6							
TX_MESSAGE_7	R/W	1B	Tx Data byte 7							
RX1_ARB_0	R/W	20	Rx1 buffer Arbitration register – RX1ARB28:RX1ARB21							
RX1_ARB_1	R/W	21	Rx1 buffer Arbitration register – RX1ARB20:RX1ARB13							
RX1_ARB_2	R	22	Rx1 buffer Arbitration register – RX1ARB12:RX1ARB5							
RX1_ARB_3	R	23	Rx1 buffer Arbitration register- RX1ARB4:RX1ARB0 & 3 reserved bit							
RX1_MESSAGE_0	R	24	Rx1 buffer Data byte 0							
RX1_MESSAGE_1	R	25	Rx1 buffer Data byte 1							
RX1_MESSAGE_2	R	26	Rx1 buffer Data byte 2							
RX1_MESSAGE_3	R	27	Rx1 buffer Data byte 3							
RX1_MESSAGE_4	R	28	Rx1 buffer Data byte 4							
RX1_MESSAGE_5	R	29	Rx1 buffer Data byte 5							
RX1_MESSAGE_6	R	2A	Rx1 buffer Data byte 6							
RX1_MESSAGE_7	R	2B	Rx1 buffer Data byte 7							
RX1_STATUS	R	2C	RX1 reserved	RX1 reserved	RX1 reserved	RX1 Extfr	Rx1 DLC3	Rx1 DLC2	Rx1 DLC1	Rx1 DLC0
RX2_ARB_0	R/W	30	Rx2 buffer Arbitration register – RX2ARB28:RX2ARB21							
RX2_ARB_1	R/W	31	Rx2 buffer Arbitration register – RX2ARB20:RX2ARB13							
RX2_ARB_2	R	32	Rx2 buffer Arbitration register – RX2ARB12:RX2ARB5							
RX2_ARB_3	R	33	Rx2 buffer Arbitration register- RX2ARB4: RX2ARB0 & 3 reserved bit							
RX2_MESSAGE_0	R	34	Rx2 buffer Data byte 0							
RX2_MESSAGE_1	R	35	Rx2 buffer Data byte 1							
RX2_MESSAGE_2	R	36	Rx2 buffer Data byte 2							
RX2_MESSAGE_3	R	37	Rx2 buffer Data byte 3							
RX2_MESSAGE_4	R	38	Rx2 buffer Data byte 4							

RX2_MESSAGE_5	R	39	Rx2 buffer Data byte 5							
RX2_MESSAGE_6	R	3A	Rx2 buffer Data byte 6							
RX2_MESSAGE_7	R	3B	Rx2 buffer Data byte 7							
RX2_STATUS	R	3C	RX2 reserved	RX3 reserved	RX3 Reserved	RX2 Extfr	Rx2 DLC3	Rx2 DLC2	Rx2 DLC1	Rx2 DLC0
RX3_ARB_0	R/W	40	Rx3 buffer Arbitration register – RX3ARB28:RX3ARB21							
RX3_ARB_1	R/W	41	Rx3 buffer Arbitration register – RX3ARB20:RX3ARB13							
RX3_ARB_2	R	42	Rx3 buffer Arbitration register – RX3ARB12:RX3ARB5							
RX3_ARB_3	R	43	Rx3 buffer Arbitration register- RX3ARB4:RX3ARB0 & 3 reserved bit							
RX3_MESSAGE_0	R	44	Rx3 buffer Data byte 0							
RX3_MESSAGE_1	R	45	Rx3 buffer Data byte 1							
RX3_MESSAGE_2	R	46	Rx3 buffer Data byte 2							
RX3_MESSAGE_3	R	47	Rx3 buffer Data byte 3							
RX3_MESSAGE_4	R	48	Rx3 buffer Data byte 4							
RX3_MESSAGE_5	R	49	Rx3 buffer Data byte 5							
RX3_MESSAGE_6	R	4A	Rx3 buffer Data byte 6							
RX3_MESSAGE_7	R	4B	Rx3 buffer Data byte 7							
RX3_STATUS	R	4C	RX3 reserved	RX3 reserved	RX3 reserved	RX3 Extfr	Rx3 DLC3	Rx3 DLC2	Rx3 DLC1	Rx3 DLC0

Setup Registers

The five 8 bits SETUP registers are used to set specific CAN configurations. The 5 bits shown in grey on SETUP_3 register and the 3 bits on SETUP_RX register are not written directly inside the interface but are used to perform reset or other actions.

SETUP_0 :

Txint**:** active high, the AT7908E generates the interrupt on the INT line after a successful transmission.

Rxint**:** active high, the AT7908E generates one interrupt on the INT line after a successful data frame reception.

Overint**:** active high, the AT7908E generates the interrupt on the INT line if the receiver buffer (number n) has not been cleared(with RxClr) before the next valid message for the same receiver buffer.

Errint**:** active high, the AT7908E generates the interrupt on the INT line if there's an error condition on the CAN Controller (bus-off or error-passive).

GensyncRx**:** the AT7908E generates a pulse on the hasync output signal when a receiving message is detected.

GensyncTx**:** the AT7908E generates a pulse on the hasync output signal when a sync pulse on the transmitting message is detected.

BPR0-BPR1**:** 2 bits to program the CAN clock pre-scaler (4 different system clocks of the CAN Core). In the next table, it is possible to see the different values of the system clock depending on the bits BPR0 and BPR1.

BPR0	BPR1	System clock frequency
0	0	f_{osc}
0	1	$f_{osc}/2$
1	0	$f_{osc}/4$
1	1	$f_{osc}/8$

SETUP_1:

***TXDLC*<3:0>:** length of the transmitted message.

***TMRMR*:** Active High, establishes if the transmitted message is a remote frame (if TXRM=0).

***TXEM*:** active high, it establishes if the transmitted message is an extended frame.

***TXRM*:** Active high**,** it establishes if the controller answers to the remote frame request (if TMRMR=0).

***Disabled*:** active high, by setting this bit, the CAN controller is disabled and disconnected from the bus. This condition could be used to set the other registers safely. The error counter values, when this bit is active, will be frozen at the last value.

SETUP_2:

PS1_3 … PS1_0: these 4 bits are the phase segment length 1 (acceptable value 1:15). The default value of this register (value after reset) is 9 decimal.

PS2_3 … PS2_0: these 4 bits are the phase segment length 2 (acceptable value 1:8). The default value of this register (value after reset) is 5 decimal.

SETUP_3 :

RSJ_2 … RSJ_0: 3 bits for the re-synchronization jump (correct value 1:4). The default value is 2.

Txreq: when this bit is set high, the AT7908E sends on the CAN bus the message contained in the TX buffer. This bit must be written from the MCU to start the transmission.

AbortTx: this bit must be set to inform the interface that the transmission request has to be aborted. The error counter values are not reset when this bit is set.

IntClr: this bit resets the INT signal of the device and clears the SyncTx and SyncRx bits of the STATUS register.

Reset: active high, this signal must be set to reset the CAN controller. The error condition is cleared and also the counter values.

RxClr: active high, this bit is set to clear all RXOKn and RXOVRn bits on the status register of the receiver message buffers (STATUS_RX).

SETUP_RX Register

RxClr3: active high, this bit is set to clear the RXOK3 and RXOVR3 bits on the status register of the receiver message buffer (STATUS_RX). This bit must be set after the read out operation of the received message otherwise, at the next reception on the same RX message buffer, an overrun condition is raised.

RxClr2: active high, this bit is set to clear the RXOK2 and RXOVR2 bits on the status register of the receiver message buffers. This bit must be set after the read out operation of the received message otherwise, at the next reception on the same RX message buffer, an overrun condition is raised.

RxClr1: active high, this bit is set to clear the RXOK1 and RXOVR1 bits on the status register of the receiver message buffers. This bit must be set after the read out operation of the received message otherwise, at the next reception on the same RX message buffer, an overrun condition is raised.

Status Registers

STATUS_0

BusOff: this bit indicates that the CAN controller has reached a bus off condition.

ErrPass: this bit indicates that the CAN controller has reached an error passive condition.

TxActive: this bit signals that there is an active transmission.

TxOK: this bit indicates that the transmission has been successfully terminated. This bit is reset automatically when the TXREQ bit on the SETUP_3 register is set.

Rxbuf1:Rxbuf0: these bits indicate which receiver buffer received a message:

00 : message filtered out

01 : message received by RX1 buffer

10 : message received by RX2 buffer

11 : message received by RX3 buffer

SyncRx: this bit indicates that a pulse on the sync bit of the received message has been generated.

SyncTx: this bit indicates that a pulse on the sync bit of the transmitted message has been generated.

STATUS_RX:

RXOVR3: This bit indicates an overrun condition (a new message received on the message buffer 3 with RXOK3 not cleared).

RXOVR2: This bit indicates an overrun condition (a new message received on the message buffer 2 with RXOK2 not cleared).

RXOVR1: This bit indicates an overrun condition (a new message received on the message buffer 1 with RXOK1 not cleared).

RXOK3: This bit indicates (active high) that the data frame message has been received correctly by the third message buffer.

RXOK2: This bit indicates (active high) that the data frame message has been received correctly by the second message buffer.

RXOK1: This bit indicates (active high) that the data frame message has been received correctly by the first message buffer.

Filter Registers

The 2 filter registers are used (together with the arbitration register of the receiver buffers or the arbitration registers of the transmit buffer to answer to the remote request) to filter out the messages that are not interesting for the receiving software. The filtering function is implemented only on 11 bits of the identifier (message_id28 downto message_id18) but works for both, extended and standard messages. The filtering function is:

Message Valid if:

for i = 28 downto 18

(“MsgID(i)” XOR “RXnARB(i)”)AND “AM(i)” = 0

The Message ID bits that are checked with the arbitration of the receiver buffer bits are the corresponding bits on the Acceptance Mask only if set to 1.

This filtering function is implemented for every message buffer.

TX Message Buffer

The AT7908E CAN controller has one transmitter buffer. The transmitter buffer is composed of 12 registers of 8 bits. The firsts 4 registers are used for the arbitration part that could be of 11 or 29 bits. If the message to transmit is a standard message, the AT7908E will send on the bus the identifier from ARB28 to ARB18. The other 8 registers contain the data byte of the message:

TX_ARB0: bits 28 down to 21 of the arbitration of the transmitter buffer(TXARB28:TXARB21)

TX_ARB1: bits 20 down to 13 of the arbitration of the transmitter buffer(TXARB20:TXARB13)

TX_ARB2: bits 12 down to 5 of the arbitration of the transmitter buffer (TXARB12:TXARB5)

TX_ARB3: bits 4 down to 0 of the arbitration of the transmitter buffer (TXARB4:TXARB0)

TX_Message_0: first 8 bits data of the transmitter buffer

TX_Message_1: second 8 bits data of the transmitter buffer

TX_Message_2: third 8 bits data of the transmitter buffer

TX_Message_3: fourth 8 bits data of the transmitter buffer

TX_Message_4: fifth 8 bits data of the transmitter buffer

TX_Message_5: sixth 8 bits data of the transmitter buffer

TX_Message_6: seventh 8 bits data of the transmitter buffer

TX_Message_7: eighth 8 bits data of the transmitter buffer

RX Message Buffers

The AT7908E CAN controller has three receiver message buffers. The incoming data frame message will be stored on the message object that matches the incoming ID (see filtering functionality). If the incoming ID matches more than one ID of the message object, the message will be stored in the lowest numbered message object. The incoming remote request is not stored on the message buffer.

The structure of the message buffer is:

RXn_STATUS: Status register of the message object n

RXn_ARB0: bits 28 down to 21 of the arbitration of the message object n

RXn_ARB1: bits 20 down to 13 of the arbitration of the message object n

RXn_ARB2: bits 12 down to 5 of the arbitration of the message object n

RXn_ARB3: bits 4 down to 0 of the arbitration of the message object n

RXn_Message_0: first 8 bits data of the message object n

RXn_Message_1: second 8 bits data of the message object n

RXn_Message_2: third 8 bits data of the message object n

RXn_Message_3: fourth 8 bits data of the message object n

RXn_Message_4: fifth 8 bits data of the message object n

RXn_Message_5: sixth 8 bits data of the message object n

RXn_Message_6: seventh 8 bits data of the message object n

RXn_Message_7: eighth 8 bits data of the message object n

The following bit composes the **RXn_STATUS** register of the message buffer:

RXn_DLC3:RXn_DLC0: length of the received message

(length = RXn_DLC0 +2 x RXn_DLC1 + 4 x RXn_DLC2 + 8 x RXn_DLC3)

RXn_extfr: if this bit is high, the received message has an extended identifier.

Error Counters Registers

The AT7908E has two internal counters for the RX and TX errors. The values of these counters are stored into two registers that can be read from the MCU.

Trigger Match Registers

The Trigger Match registers are implemented to generate a pulse on the HATRIG output signal when the received message arbitration match the Trigger Match registers (see the trigger match functionality).

Bit Timing

A bit period is composed of the following three segments:

Synchronization segment

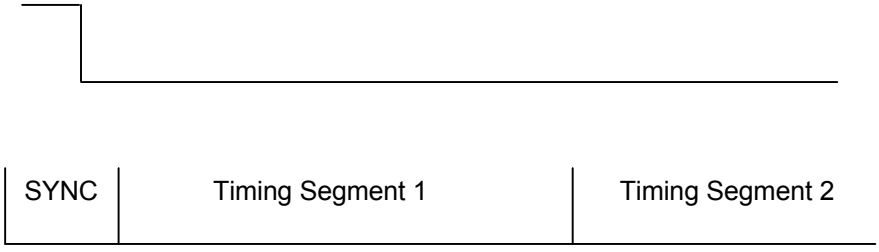
Timing segment 1

Timing segment 2

The sampling point is at the end of time segment 1.

Input signal

Figure 2. Bit Time Segments



During the **Sync segment** (1 system clock cycle = t_{scl}) the edge of the input signal is expected.

The **Timing segment 1** is programmable from 2 to 16 t_{scl} (see register SETUP_2: PS1_3:PS1_0: **TSEG1 =PS1+1**) and the end of this segment indicates the sample point.

The **Timing segment 2** is programmable from 1 to 8 t_{scl} (register SETUP_2: PS2_3:PS2_0) and this period is used to have extra time for the internal processing after the sample point.

The **resynchronization Jump Width** is used to compensate phase shifts between the oscillator frequency of the different CAN nodes on the network. This value is programmable (see register SETUP_3: RSJ_2:RSJ_0) from 1 to 4 t_{scl} and the value indicates the number of system clock pulses by which the bit period must be shortened or lengthened for resynchronization. If the falling edge of the incoming signal is on the TIMING segment 1, then the Bit period is lengthened (the sample point will be at TSEG1 +RSJ). If the falling edge of the incoming signal is on the Timing segment 2, then the bit period

will be shortened (TSEG2 is shortened of RSJ value). The resynchronization mechanism is shown in fig. 3 and fig. 4.

Figure 3. Lengthening a Bit Period

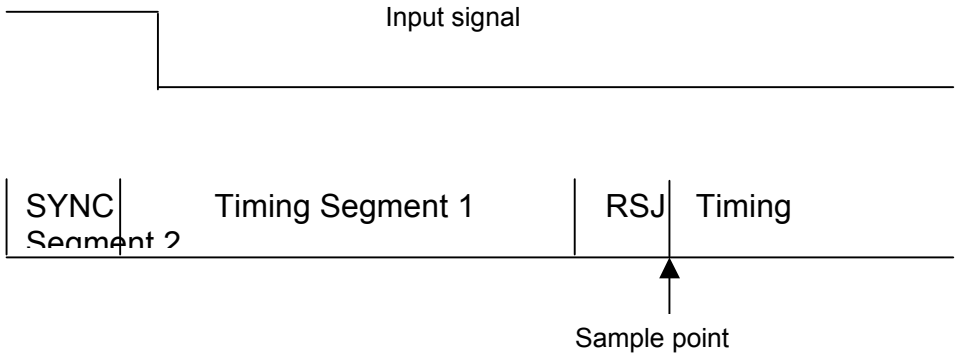
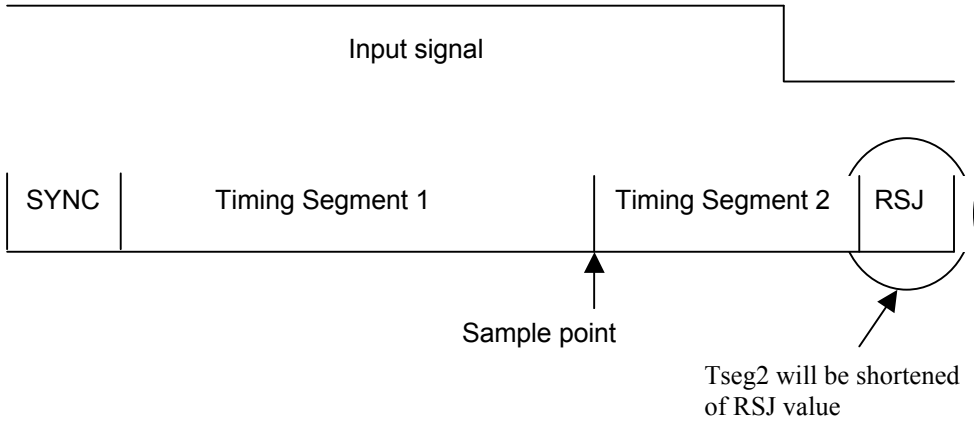


Figure 4. Shortening a Bit Period



The Bit rate of the message on the bus will be:

f_{osc}/(BPR(baud rate prescaler)x(TSEG1 +TSEG2+1)) =

= f_{osc} /(BPR(baud rate prescaler) x(PS1 +PS2 +2))

The TSEG1 and TSEG2 lengths must be programmed to respect these conditions:

TSEG1 ³ TSEG2 => PS1 +1 ³ PS2

TSEG2 ³ RSJ => PS2 ³ RSJ

Interface Block Description

The AT7908E provides a programmable (with external pin) MCU interface. Two modes can be selected. The first operational mode is an interface with 8-bit multiplexed address data bus (mode = 0) and an internal register addressable with 16-bit of address. In this operational mode, the AT7908E registers are mapped between 8000Hex and 804CHex.

The second operational mode (mode =1) is implemented with 8 bit not multiplexed address and data buses. In this mode the internal registers are accessible with 8-bit and are mapped between 00Hex and 4C Hex.

The pins for the interface block are:

Mode	Selection of the interface modality. This pin must be **0** to use 8 bits multiplexed data address (lowest) to map the AT7908E on the highest address space (8000Hex to 804C Hex). If MODE = 1, the data and addresses are not multiplexed and the registers are mapped on the lowest address space.
Data <7:0>	bi-directional 8-bit address (low address in mode 0) data bus
ALE	Input Address Latch Enable used for mode 0.
CS	Input Chip select to enable the internal registers access (active high).
WR	Input Write signal to write the internal registers (active low).
RD	Input Read signal to readout the internal registers (active low).
Addr <7:0>	Input highest address in **mode 0** (used with low address to map the AT7908E register between 8000Hex and 804Chex address space) or input lowest address in **mode 1**

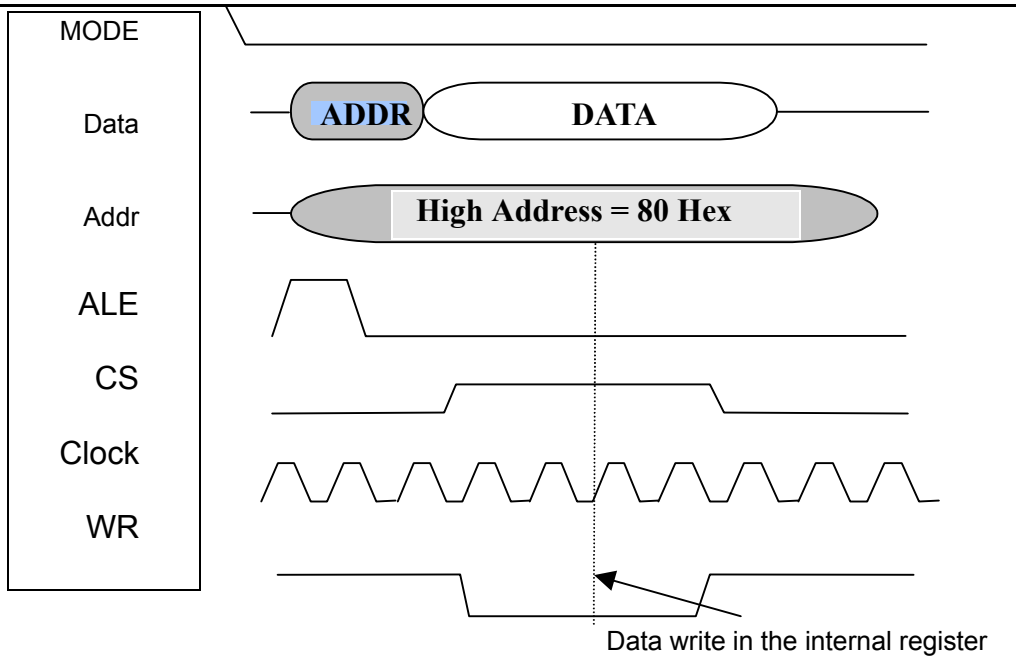
Operational Mode 0

This operational mode is selected with **MODE** pin = 0.

The bus Data_addr <7:0> must be connected to the data/address bus of the MCU. The ALE signal is used to latch the address. This latched address will be the address of the AT7908E internal registers. The bus addr (7:0) will contain the highest address generated by the MCU. The internal AT7908E registers will be accessed only if the 16-bit address generated by the MCU is between 8000 Hex and 804C Hex (the internal Chip Select will be generated). Moreover, to write in the register or to read from the register, the micro-controller must generate the CS (active high), WR, RD and ALE signals and must drive the Data_addr and addr buses. The signal WR must be low for, at least, 3 clock cycles. The data is latched at the rising edge of the clock when CS = 1, WR = 0 and RD = 1.

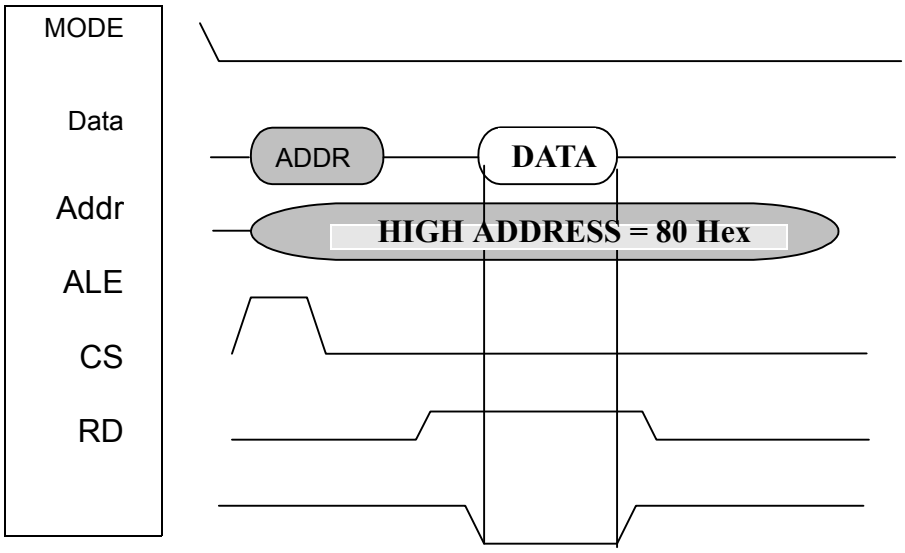
A write cycle is reported in Fig. 5.

Figure 5. Mode 0 Write Cycle



In Fig. 6 are represented the signals to readout the internal register of AT7908E in mode 0

Figure 6. Mode 0 Read Cycle



In mode 0, the internal registers of the AT7908E are mapped between 8000 Hex and 804C Hex and the lowest 8-bit address are multiplexed with the data.

Operational Mode 1

This operational mode is selected with **mode** pin = 1.

This interface operational mode is used to write the AT7908E registers and to readout from the AT7908E with two different 8-bit data and address buses. The ALE input pin is not used and the internal registers are mapped between 00Hex and 4C Hex. The Addr bus will be used as the 8 bits address for the internal registers**.** To write data on the internal registers, the MCU must control the CS, WR and RD signals of the AT7908E and must drive the addr and data buses. The signal WR must be low for, at least, 3 clock cycles. The data is latched at the rising edge of the clock when CS = 1, WR = 0 and RD = 1. The fig. 7 shows the write cycle in the operational mode 1.

Figure 7. Mode 1 Write Cycle

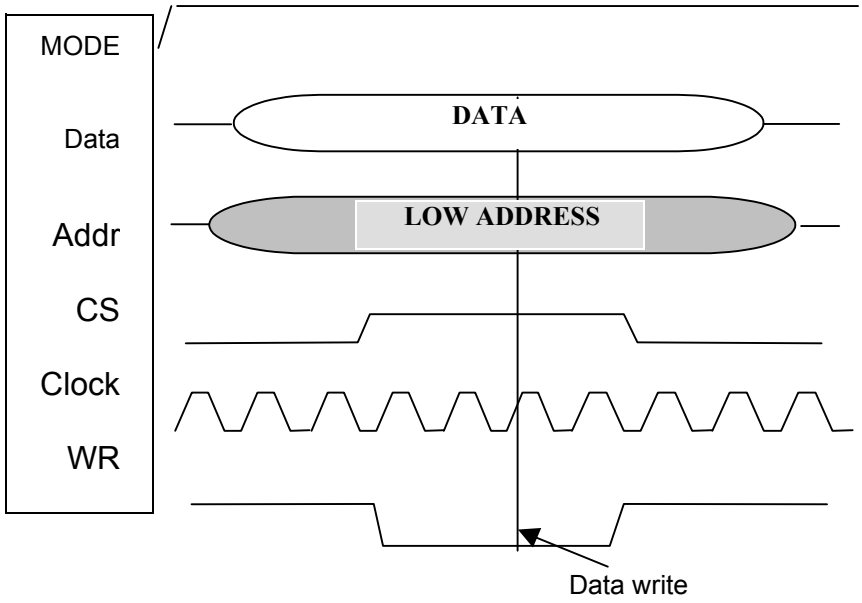
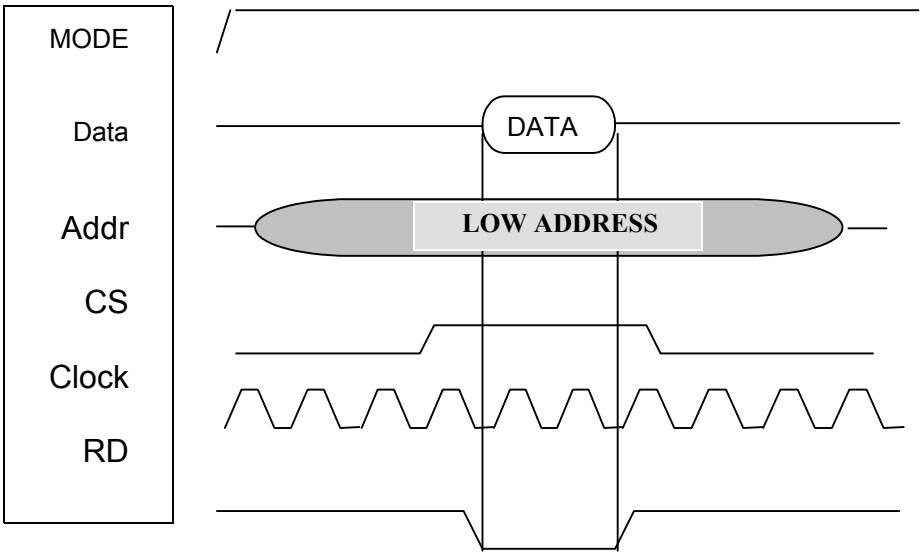


Fig. 8 shows a read cycle. To readout the data from the AT7908E internal registers, the MCU must drive CS=1, RD=0, WR=1.

Figure 8. Mode 1 Read Cycle



Timing

In this paragraph are provided the AC specification for the AT7908E interface signals in both operational modes.

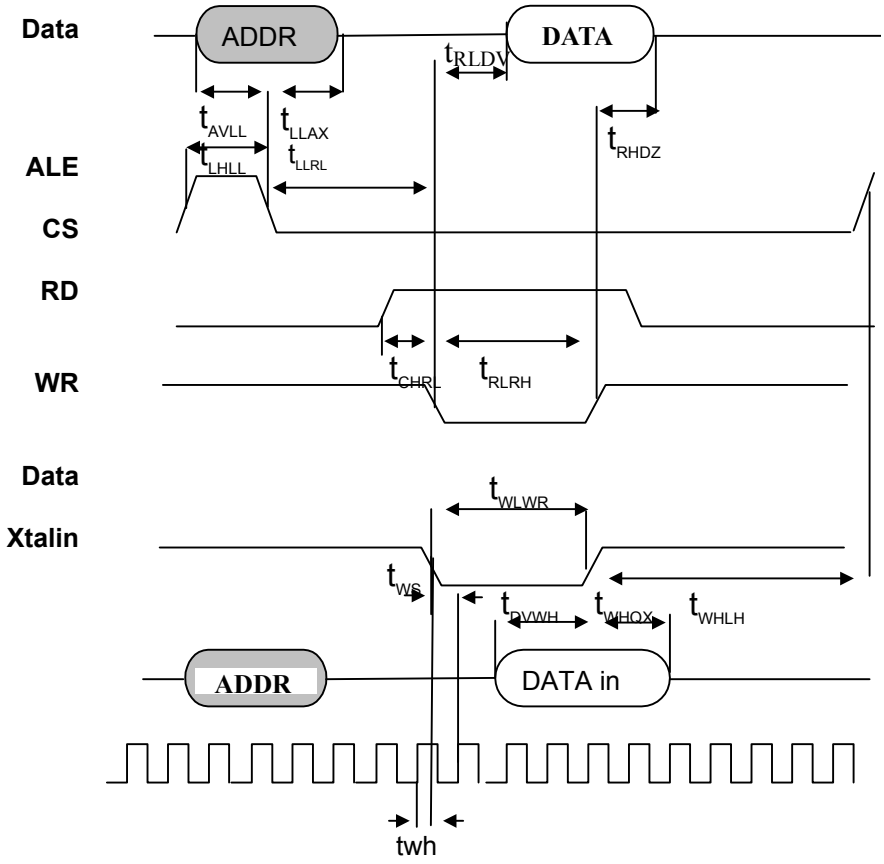
AC Specification for 8 bit Multiplexed Mode (mode =0)

Conditions: VCC = 5V ±10%, VSS = 0V, Ta = -55°C to +125 °C, Cl = 80 pF , Tclk = 100 ns

Symbol	Parameter	Min	Max	Note
t_{AVLL}	Address valid to ALE low	4 ns		Mode 0
t_{LLAX}	Address Hold after ALE low	4 ns		Mode 0
t_{LHLL} (*)	ALE HIGH Time	Tclk		Mode 0
t_{LLRL} (*)	ALE low to RD low	2 Tclk		Mode 0
t_{CHRL}	CS high to RD low	100 ns		Mode 0
$t_{DVWH]}$ (*)	Input Data valid to WR High	3 Tclk		Mode 0, mode 1
t_{WHQX} (*)	Input data hold after WR high	10 ns		Mode 0, mode 1
t_{WLWH} (*)	WR pulse width	3 Tclk		Mode 0, mode 1
t_{WHLH} (*)	WR high to next ALE high	Tclk		
t_{WS}	WR setup time before clock	-3 ns		Mode 0, mode 1
t_{WH} (*)	WR hold time after clock	7 ns		Mode 0, mode 1
t_{RLRH} (*)	RD pulse width	3 Tclk		Mode 0, mode 1
t_{RLDV} (*)	RD low to data valid	6 ns	82 ns	Mode 0, mode 1
t_{RHDZ} (*)	Data float after RD High	4 ns	20 ns	Mode 0, mode 1

Note: 1. Guaranteed, not tested.

Figure 9. Timing information for interface signals

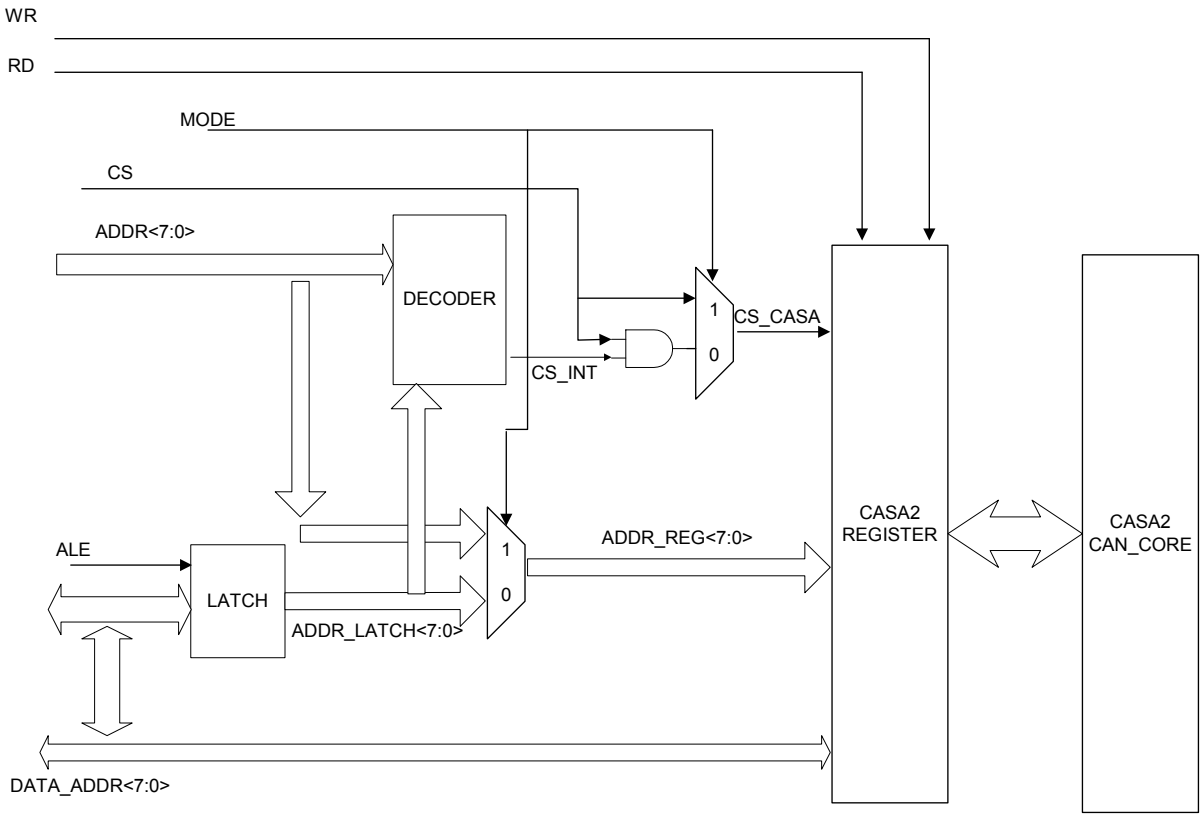


Interface Internal Structure

This paragraph is intended to explain the implementation of the interface internal structure.

In the Figure 10, one can see the realization of the two operational modes and the access to the AT7908E internal registers.

Figure 10. Interface Block Scheme



The MCU 80C32 generates **DATA<7:0>, ADDR<7:0>** (highest 8-bit address for the 16-bit external data access), **and ALE to latch** the **lowest address, WR, CS** and **RD** signals.

The external data bus is connected to the internal latch block of the AT7908E. The **ALE** signal latches and extracts the **ADDR_LATCH** (lowest 8-bit address). The **ADDR_LATCH** is used to address the internal registers in mode 0 and to generate, with **ADDR<7:0>** (highest 8-bit address), the internal chip select signal (**CS_INT**). In mode 0, the access to the internal registers is selected by CS_INT and CS; in mode 1, the access is selected by **CS**. In mode 1, moreover, the address of the internal register is **ADDR<7:0>**. The two different operational modes are established by the **mode** pin that is the selector for the two multiplexers in Figure 5. In Figure 5 are reported, in addition, the other two blocks of the AT7908E device: the AT7908E registers and the AT7908E CAN_CORE.

Address Table

In the next table, one can see how the MCU can address the RAM or the internal register of the AT7908E in the operational mode 0.

ADDRESS Generated by the MCU (15 down to 0)	Device
0000 – 7FFF Hex	32 Kword of External RAM
8000 – 800C Hex	AT7908E internal register 00 – 0C Hex (Configuration and status register)
8010 – 801B Hex	AT7908E internal register 10 – 1B Hex Transmitter message buffer
8020 – 802C Hex	AT7908E internal register 20 – 2C Hex First receiver message object
8030 – 803C Hex	AT7908E internal register 30 – 3C Hex Second receiver message object
8040 – 804C Hex	AT7908E internal register 40 – 4C Hex Third receiver message object

Optional Features

Internal Clock Frequency

The actual internal system clock can be the external clock divided by 1, 2, 4, or 8. If the bit time length is (default configuration) 16 system clock pulses (see PS1, PS2 programmable bit timing registers), it is possible to have four different data rates by the prescaler programming.

AT7908E Operational Mode

The AT7908E device could be put on three different operational modes:

- Functional Mode
- Test mode
- Power down mode

Functional Mode

This mode is the normal operational mode for the AT7908E device. The two input pins **test** and **sena** must be put on logical value 0.

Test Mode

This mode is used by the chip manufacturer to test the AT7908E. The input pin **test** must be logical value 1 and **sena** input pin could be logical values 0 or 1 according to the execution of the scan chain test or not.

Power Down Mode

This mode could be used to put the AT7908E device on the sleep mode (internal clock is off). To put the AT7908E in this mode, the **test** input pin must be 0 and **sena** input pin must be 1 .

Application Notes

Registers Value After Reset

In the next table are reported the registers value after reset of the AT7908E device:

Register Name	Address Hex	Reset Value (bit7… bit0)							
SETUP_0	00	BPR1	BPR0	GensyncTx	GensyncRx	Errint	Overint	Rxint	Txint
		0	0	0	0	0	0	0	0
SETUP_1	01	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
		1	0	0	0	0	0	0	0
SETUP_2	02	PS2_3	PS2_2	PS2_1	PS2_0	PS1_3	PS1_2	PS1_1	PS1_0
		0	1	0	1	1	0	0	1
SETUP_3	03	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
		0	0	0	0	0	0	1	0
SETUP_RX	04	Reserved	Reserved	Reserved	Reserved	Reserved	Rxclr3	Rxclr2	Rxclr1
		X	X	X	X	X	0	0	0
STATUS	05	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
		0	0	0	0	0	0	0	0
STATUS_RX	06	reserved	Rxovr3	Rxovr2	Rxovr1	Reserved	RXOK3	RXOK2	RXOK1
		0	0	0	0	0	0	0	0
FILTER_AM_0	07	00000000							
FILTER_AM_1	08	00000000							
ERR_COUNT_TX	09	00000000							
ERR_COUNT_RX	0A	00000000							
TRIG_MATCH_0	0B	00000000							
TRIG_MATCH_1	0C	00000000							
TX_ARB_0	10	00000000							
TX_ARB_1	11	00000000							
TX_ARB_2	12	00000000							
TX_ARB_3	13	00000000							
TX_MESSAGE_0	14	00000000							
TX_MESSAGE_1	15	00000000							
TX_MESSAGE_2	16	00000000							
TX_MESSAGE_3	17	00000000							
TX_MESSAGE_4	18	00000000							
TX_MESSAGE_5	19	00000000							
TX_MESSAGE_6	1A	00000000							
TX_MESSAGE_7	1B	00000000							
RX1_ARB_0	20	00000000							
RX1_ARB_1	21	00000000							
RX1_ARB_2	22	00000000							

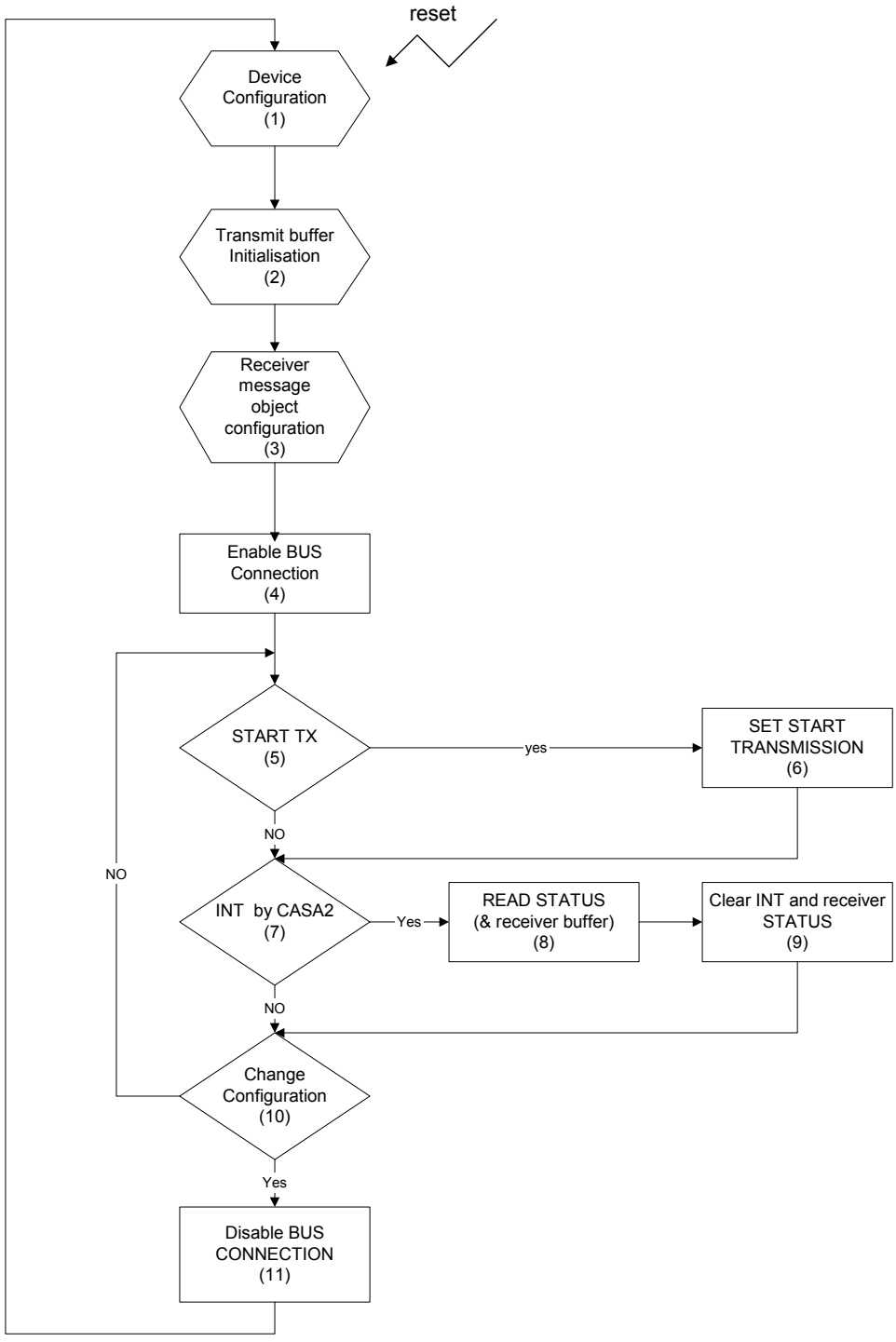
Register Name	Address Hex	Reset Value (bit7… bit0)							
RX1_ARB_3	23	00000000							
RX1_MESSAGE_0	24	00000000							
RX1_MESSAGE_1	25	00000000							
RX1_MESSAGE_2	26	00000000							
RX1_MESSAGE_3	27	00000000							
RX1_MESSAGE_4	28	00000000							
RX1_MESSAGE_5	29	00000000							
RX1_MESSAGE_6	2A	00000000							
RX1_MESSAGE_7	2B	00000000							
RX1_STATUS	2C	reserved	reserved	Reserved	RX1_extfr	Rx1_DLC3	Rx1_DLC2	Rx1_DLC1	Rx1_DLC0
		X	X	X	0	0	0	0	0
RX2_ARB_0	30	00000000							
RX2_ARB_1	31	00000000							
RX2_ARB_2	32	00000000							
RX2_ARB_3	33	00000000							
RX2_MESSAGE_0	34	00000000							
RX2_MESSAGE_1	35	00000000							
RX2_MESSAGE_2	36	00000000							
RX2_MESSAGE_3	37	00000000							
RX2_MESSAGE_4	38	00000000							
RX2_MESSAGE_5	39	00000000							
RX2_MESSAGE_6	3A	00000000							
RX2_MESSAGE_7	3B	00000000							
RX2_STATUS	3C	reserved	reserved	Reserved	RX2_extfr	Rx2_DLC3	Rx2_DLC2	Rx2_DLC1	Rx2_DLC0
		X	X	X	0	0	0	0	0
RX3_ARB_0	40	00000000							
RX3_ARB_1	41	00000000							
RX3_ARB_2	42	00000000							
RX3_ARB_3	43	00000000							
RX3_MESSAGE_0	44	00000000							
RX3_MESSAGE_1	45	00000000							
RX3_MESSAGE_2	46	00000000							
RX3_MESSAGE_3	47	00000000							
RX3_MESSAGE_4	48	00000000							
RX3_MESSAGE_5	49	00000000							
RX3_MESSAGE_6	4A	00000000							

Register Name	Address Hex	Reset Value (bit7… bit0)							
RX3_MESSAGE_7	4B	00000000							
RX3_STATUS	4C	reserved	reserved	Reserved	RX3_extfr	Rx3_DLC3	Rx3_DLC2	Rx3_DLC1	Rx3_DLC0
		X	X	X	0	0	0	0	0

Configuration Flow

The following flow diagram explains the action that the MCU must perform to program the AT7908E CAN controller to send or to receive CAN message.

Figure 11. Operating Flow Diagram



In the examples of the followings paragraph the different operations are referred to the number reported on flow diagram. After the commands, the old register and the new register values are reported.

Initialization and Transmission of the Data Frame with the INT Generation and Behavior of the MCU After the Interrupt is Received

Configuration of all the setup registers after the RESET signal.

Operation 1

Table 1. write reg 00 H, data =01Hex

(SETUP_0 register) system clock = external clock, enable transmission completed interrupt

	BPR1	BPR0	GensyncTx	GensyncRx	Errint	Overint	Rxint	Txint
SETUP_0	0	0	0	0	0	0	0	0

	BPR1	BPR0	GensyncTx	GensyncRx	Errint	Overint	Rxint	Txint
SETUP_0	0	0	0	0	0	0	0	1

Table 2. write reg 02 H, data =69Hex

(SETUP_2 register) Time segment 2 = 6 pulse of system clock; Time segment 1 = 8 pulse of system clock.

SETUP_2	PS2_3	PS2_2	PS2_1	PS2_0	PS1_3	PS1_2	PS1_1	PS1_0
	0	1	0	1	1	0	0	1

SETUP_2	PS2_3	PS2_2	PS2_1	PS2_0	PS1_3	PS1_2	PS1_1	PS1_0
	0	1	1	0	1	0	0	1

SETUP_3 register will be not written (default configuration)

Operation 2

After SETUP registers is necessary to configure TX message object:

Table 3. write reg 10 H, data =AA Hex
write reg 11 H, data =AA Hex

(TX_ARB_0= and TX_ARB_1 registers) identifier of message that will be transmitted = 10101010101

TX_ARB_0	00000000
TX_ARB_1	00000000

TX_ARB_0	10101010
TX_ARB_1	10101010

Table 4. write reg 14 H, data =00Hex

TX_MESSAGE_0	00000000

⇩

TX_MESSAGE_0	00000000

Table 5. write reg 15 H, data =FF Hex

TX_MESSAGE_1	00000000

⇩

TX_MESSAGE_1	11111111

Table 6. write reg 16 H, data =0FHex

TX_MESSAGE_2	00000000

⇩

TX_MESSAGE_2	00001111

Table 7. write reg 17 H, data =F0Hex

TX_MESSAGE_3	00000000

⇩

TX_MESSAGE_3	11110000

Table 8. write reg 18 H, data =AA Hex

TX_MESSAGE_4	00000000

⇩

TX_MESSAGE_4	10101010

Table 9. write reg 19 H, data =55Hex

TX_MESSAGE_5	00000000

⇩

TX_MESSAGE_5	01010101

Table 10. write reg 1A H, data =66Hex

TX_MESSAGE_6	00000000

⇩

TX_MESSAGE_6	01100110

Table 11. write reg 1B H, data =99Hex

TX_MESSAGE_7	00000000

⇩

TX_MESSAGE_7	10011001

(TX_MESSAGE_n registers) Setting of data message that must be transmitted

When TX message object initialized,the MCU must connect the AT7908E to the CAN bus.

Operation 4

Table 12. write reg 01 H, data =08Hex

(SETUP_1 register) AT7908E connected to BUS, data frame(no remote frame), standard message, length of message = 8 byte.

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC 3	TXDLC 1	TXDLC 1	TXDLC 0
	1	0	0	0	0	0	0	0

⇩

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC 3	TXDLC 1	TXDLC 1	TXDLC 0
	0	0	0	0	1	0	0	0

Setting of the transmission request

Operations 5 and 6

Table 13. write reg 03 H, data =0Ahex (1)

(SETUP_3 register) Resynchronisation jump length = 2 system clock and transmission request.

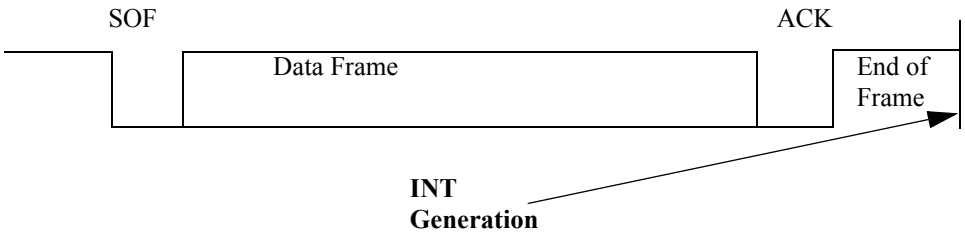
SETUP_3	RxClr	Reset	IntClr	Abort Tx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

⇩

SETUP_3	RxClr	Reset	IntClr	Abort Tx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

Then, the AT7908E starts to transmit the message until an acknowledge comes from another CAN node. The next figure illustrates the standard CAN format on the CAN BUS:

Figure 12. DATA frame



STATUS register before TX request:

STATUS	SyncTx	Sync Rx	Rxbuf 1	Rxbuf 0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	0	0	0

STATUS register after start of transmission:

STATUS	SyncTx	Sync Rx	Rxbuf 1	Rxbuf 0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	1	0	0

STATUS register after completion of correct transmission:

1. The five Gray bit are not updated after setting (Txreq). These bit are not written inside the registers.

Operation 7

STATUS	SyncTx	Sync Rx	Rxbuf 1	Rxbuf 0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	1	0	0	0

At this point, the AT7908E sends an interrupt generation ;INT signal (Active low).

The MCU clears the interrupt generation, setting the IntClr bit on the SETUP_3 register.

Now we assume that the MCU receives the INT signal and it wants to program the AT7908E to send the same data frame on the CAN bus.

Operation 8

Table 14. read reg 05 H

(MCU read status register after INT received)

STATUS	SyncTx	Sync Rx	Rxbuf 1	Rxbuf 0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	1	0	0	0

Operations 5, 6, 9, and 10

Table 15. write reg 03 H, data =2AHex

(SETUP_3 register) Clear INT signal and start new transmission(TXREQ)Resynchronisation jump length = 2 system clock and transmission request.

SETUP_3	RxClr	Reset	IntClr	Abort Tx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

⇩

SETUP_3	RxClr	Reset	IntClr	Abort Tx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

After this setting, the INT signal becomes high (interrupt request disabled), the AT7908E starts a new transmission and the STATUS register is automatically updated as follows:

STATUS	SyncTx	Sync Rx	Rxbuf 1	Rxbuf 0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	1	0	0	0

⇩

STATUS	SyncTx	Sync Rx	Rxbuf 1	Rxbuf 0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	1	0	0

Initialisation of the Receiver Message Objects and the Filtering Function

In this paragraph is reported an example about configuration of the AT7908E to receive message with the filtering function. In this example the MCU, after configuration of the AT7908E, waits for Interrupt (from the AT7908E) and then the MCU reads the status registers, the RX message object and it clears the interrupt request and the receiver status.

Configuration of the AT7908E:

Operation 1

Table 16. Write reg 00 H, data = 0FHex

(SETUP_0 register) system clock = external clock, all interrupts enabled (rx completed, tx completed, overrun, error state).

SETUP_0	BPR1	BPR0	Gensy ncTx	Gensy ncRx	Errint	Overint	Rxint	Txint
	0	0	0	0	0	0	0	0

⇩

SETUP_0	BPR1	BPR0	Gensy ncTx	Gensy ncRx	Errint	Overint	Rxint	Txint
	0	0	0	0	1	1	1	1

SETUP_2 register will be not written (default configuration)

SETUP_3 register will be not written (default configuration)

Receiver Message Object and Filtering function configuration

Operation 3

Table 17. write reg 07 H, data =FF Hex

write reg 08 H, data =FF Hex

(FILTER_AM_0, FILTER_AM_1 registers)Global mask set to check all bits of incoming identifier.

FILTER_AM_0	00000000
FILTER_AM_1	00000000

⇩

FILTER_AM_0	11111111
FILTER_AM_1	11111111

Table 18. write reg 20 H, data =55Hex

write reg 21 H, data =55Hex

(RX1_ARB_0, RX1_ARB_1 registers)Set the identifier for receiver message object 1(ID =01010101010)

RX1_ARB_0	**00000000**
RX1_ARB_1	**00000000**

⇩

RX1_ARB_0	**01010101**
RX1_ARB_1	**01010101**

Table 19. write reg 30 H, data =AA Hex
write reg 31 H, data =AA Hex

(RX2_ARB_0, RX2_ARB_1 registers)Set the identifier for receiver message object 2(ID =10101010101)

RX2_ARB_0	**00000000**
RX2_ARB_1	**00000000**

⇩

RX2_ARB_0	**10101010**
RX2_ARB_1	**10101010**

Table 20. write reg 40 H, data =FF Hex

write reg 41 H, data =FF Hex

(RX3_ARB_0, RX3_ARB_1 registers)Set the identifier for receiver message object 3 (ID =11111111111)

RX3_ARB_0	**00000000**
RX3_ARB_1	**00000000**

⇩

RX3_ARB_0	**11111111**
RX3_ARB_1	**11111111**

Operation4

Table 21. write reg 01 H, data =00Hex

(SETUP_1 register) AT7908E connected to BUS, data frame(no remote frame), standard message

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
	1	0	0	0	0	0	0	0

⇩

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
	0	0	0	0	0	0	0	0

At this point, the AT7908E is ready to receive a message from the other CAN nodes and to generate an interrupt if the message has been received.

When the AT7908E receives a message (for example with Identifier = 11111111111), it stores the message in the receiver message object that satisfies the filtering function (in this case, the RX3 message object). In the following tables are reported the STATUS registers before and after message receiving (Rxbuf0 and Rxbuf1 bits, after the received message, contain the indication about the receiver message object that stored the message incoming).

Table 22. STATUS register before receiving.

STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	0	0	0

⇩

Table 23. STATUS register after received message

STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	1	1	0	0	0	0

Operations 5 and 7

Table 24. Automatically update of STATUS_RX register

STATUS_RX	reserved	Rxovr3	Rxovr2	Rxovr1	Reserved	RXOK3	RXOK2	RXOK1
	0	0	0	0	0	0	0	0

⇩

STATUS_RX	Reserved	Rxovr3	Rxovr2	Rxovr1	Reserved	RXOK3	RXOK2	RXOK1
	0	0	0	0	0	1	0	0

Table 25. Automatically update of RX3_STATUS register

RX3_STATUS	RX3 Reserved	RX3 reserved	RX3 reserved	RX3 extfr	Rx3 DLC3	Rx3 DLC2	Rx3 DLC1	Rx3 DLC0
	X	X	X	0	0	0	0	0

⇩

RX3_STATUS	RX3 Reserved	RX3 reserved	RX3 reserved	RX3 extfr	Rx3 DLC3	Rx3 DLC2	Rx3 DLC1	Rx3 DLC0
	X	X	X	0	1	0	0	0

The arbitration registers of the RX3 message object will store the arbitration of the received message and the data registers of the RX3 message object will store the data of the received message. At the end of frame, the INT will be generated and the action that the MCU must perform could be the followings (these operations are only read operations and the internal registers of the AT7908E are not updated):

Operation 8

read reg 06 H

(STATUS_RX register)readout of AT7908E status_rx: rxok3, rxok2, rxok1 indicates the message object that received the message(if 00 the message is not received). For example : reg 06 = 04Hex => the message has been received correctly on message object 3.

read reg 4C H

(RX3_STATUS register)readout of message object 3 status. Is possible to check extended or standard frame of received message and length of received message. For example : reg 4C = 08Hex, standard frame, length =8.

read reg 40 H

read reg 41 H

(RX3_ARB_n registers)readout of stored message identifier

read reg 44 H

read reg 45 H

read reg 46 H

read reg 47 H

read reg 48 H

read reg 49 H

read reg 4A H

read reg 4B H

(RX_3_MESSAGE_n register)readout of stored message data.

After the read out of the received message, the MCU must clear the receiver message object status (principally Rxok bit) and the INT signal setting RXCLRn bit in the

SETUP_RX register and the IntClr bit in the SETUP_3 register (see how to clear interrupt in the precedent paragraph).

Operation 5 and 6

Table 26. write reg 03 H, data =22Hex

(SETUP_3 register) Resynchronisation jump length = 2 system clock and clear interrupt

SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

⇩

SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

Operation 9

Table 27. write reg 04 H, data =04Hex

(SETUP_RX register)Clear rx ok and rx overrun bit of receiver message objects 3

SETUP_RX	Reserved	Reserved	Reserved	Reserved	Reserved	Rxclr3	Rxclr2	Rxclr1
	X	X	X	X	X	0	0	0

⇩

SETUP_RX	Reserved	Reserved	Reserved	Reserved	Reserved	Rxclr3	Rxclr2	Rxclr1
	X	X	X	X	X	0	0	0

After this operation, the RX3OK bit in STATUS_RX register will be cleared

STATUS_RX	reserved	Rxovr3	Rxovr2	Rxovr1	reserved	RXOK3	RXOK2	RXOK1
	X	0	0	0	x	1	0	0

⇩

STATUS_RX	reserved	Rxovr3	Rxovr2	Rxovr1	reserved	RXOK3	RXOK2	RXOK1
	X	0	0	0	0	0	0	0

If the MCU doesn't clear the receiver message object status (principally the Rxok bit) and the AT7908E receives a new message on the same message object (for example the RX3 message object), the message will be stored in the message object; INT will be generated (OVERRUN condition) and the STATUS_RX will be updated as following:

STATUS_RX	reserved	Rxovr3	Rxovr2	Rxovr1	reserved	RXOK3	RXOK2	RXOK1
	x	0	0	0	x	1	0	0

⇩

STATUS_RX	reserved	Rxovr3	Rxovr2	Rxovr1	reserved	RXOK3	RXOK2	RXOK1
	x	1	0	0	0	1	0	0

Setting to answer to remote frame request

In this example is reported a typical configuration to answer to the remote frame request.

Operation 1

Table 28. write reg 00 H, data =01Hex
(SETUP_0 register) system clock = external clock, transmitter interrupts enabled .

SETUP_0	BPR1	BPR0	Gensync Tx	Gensync Rx	Errint	Overint	Rxint	Txint
	0	0	0	0	0	0	0	0

⇩

SETUP_0	BPR1	BPR0	Gensync Tx	Gensync Rx	Errint	Overint	Rxint	Txint
	0	0	0	0	0	0	0	1

SETUP_2 register will be not written (default configuration)

SETUP_3 register will be not written (default configuration)

After the SETUP registers, it is necessary to configure the TX message object:

Operation 2

Table 29. write reg 10 H, data =AA Hex

write reg 11 H, data =AA Hex

(TX_ARB_0= and TX_ARB_1 registers) identifier of message that will be transmitted = 10101010101

TX_ARB_0	00000000
TX_ARB_1	00000000

⇩

TX_ARB_0	10101010
TX_ARB_1	10101010

Table 30. write reg 14 H, data =00H

TX_MESSAGE_0	00000000

⇩

TX_MESSAGE_0	00000000

Table 31. write reg 15 H, data =FF Hex

TX_MESSAGE_1	00000000

⇩

TX_MESSAGE_1	11111111

Table 32. write reg 16 H, data =0FHex

TX_MESSAGE_2	00000000

⇩

TX_MESSAGE_2	00001111

Table 33. write reg 17 H, data =F0Hex

TX_MESSAGE_3	00000000

⇩

TX_MESSAGE_3	11110000

Table 34. write reg 18 H, data =AA Hex

TX_MESSAGE_4	00000000

⇩

TX_MESSAGE_4	10101010

Table 35. write reg 19 H, data =55 Hex

TX_MESSAGE_5	00000000

⇩

TX_MESSAGE_5	01010101

Table 36. write reg 1A H, data =66Hex

TX_MESSAGE_6	00000000

⇩

TX_MESSAGE_6	01100110

Table 37. write reg 1B H, data =99Hex

TX_MESSAGE_7	00000000

⇩

TX_MESSAGE_7	10011001

(TX_MESSAGE_n registers) Setting of data message that must be transmitted as answer to remote frame request.

Operation 4

Table 38. write reg 01 H, data =48Hex
(SETUP_1 register) AT7908E connected to BUS ,setting to answer to remote frame request, standard message, length of message = 8 byte.

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
	1	0	0	0	0	0	0	0

⇩

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
	0	1	0	0	1	0	0	0

To be ready to answer to a remote frame request, the MCU must set the Txreq bit on the SETUP_3 register. After the Txreq setting, the AT7908E waits for a remote frame request that matches (see the filtering functionality) the identifier stored on the TX_ARB registers.

Operations 5 and 6

Table 39. write reg 03 H, data =0AHex
(SETUP_3 register) Resynchronisation jump length = 2 system clock and transmission request.

SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

⇩

SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

After this setting, the AT7908E waits for a remote frame request and the STATUS register will be not updated.

If the request arrives, the AT7908E will send an acknowledge and then starts to transmit the data frame in reply to the remote frame request. The STATUS register, during the answer, will be automatically updated (TX_active bit become 1).

STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	0	0	0

⇩

STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	1	0	0

At the end of the transmission, the AT7908E will send the INT to the MCU that can read out the STATUS register to check that all is OK.

Setting to send remote frame request

In this paragraph will be described the procedure to send a remote frame request.

After the Device configuration, the MCU must initialize the transmit buffer to send the remote frame request and the receiver buffers to receive the remote frame answer.

The configuration of all the setup registers after RESET signal.

Operation 1
Table 40. write reg 00 H, data =02Hex
(SETUP_0 register) system clock = external clock, enable receiving completed interrupt

SETUP_0	BPR1	BPR0	Gensync Tx	Gensync Rx	Errint	Overint	Rxint	Txint
	0	0	0	0	0	0	0	0

SETUP_0	BPR1	BPR0	Gensync Tx	Gensync Rx	Errint	Overint	Rxint	Txint
	0	0	0	0	0	0	1	0

SETUP_2 register will be not written (default configuration)

SETUP_3 register will be not written (default configuration)

After the SETUP registers, it is necessary to configure the TX message object:

Operation 2

Table 41. write reg 10 H, data =AA Hex
write reg 11 H, data =AA Hex

(TX_ARB_0= and TX_ARB_1 registers) identifier of message that will be transmitted = 10101010101

TX_ARB_0	00000000
TX_ARB_1	00000000

TX_ARB_0	10101010
TX_ARB_1	10101010

Setting of the RX message buffers (the incoming answer to the remote frame request will be stored on the RX message buffer that matches the incoming Identifier, 10101010101 in this case).

Operation 3

Table 42. write reg 07 H, data =FF Hex
write reg 08 H, data =FF Hex

(FILTER_AM_0, FILTER_AM_1 registers)Global mask set to check all bits of incoming identifier.

FILTER_AM_0	00000000
FILTER_AM_1	00000000

FILTER_AM_0	11111111
FILTER_AM_1	11111111

Table 43. write reg 20 H, data =55Hex

write reg 21 H, data =55Hex

(RX1_ARB_0, RX1_ARB_1 registers) Set the identifier for receiver message object 1(ID =01010101010)

RX1_ARB_0	**00000000**
RX1_ARB_1	**00000000**

⇩

RX1_ARB_0	**01010101**
RX1_ARB_1	**01010101**

Table 44. write reg 30 H, data =AA Hex
write reg 31 H, data =AA Hex

(RX2_ARB_0, RX2_ARB_1 registers)Set the identifier for receiver message object 2(ID =10101010101)

RX2_ARB_0	**00000000**
RX2_ARB_1	**00000000**

⇩

RX2_ARB_0	**10101010**
RX2_ARB_1	**10101010**

Table 45. write reg 40 H, data =FF Hex

write reg 41 H, data =FF Hex

(RX3_ARB_0, RX3_ARB_1 registers)Set the identifier for receiver message object 3 (ID =11111111111)

RX3_ARB_0	**00000000**
RX3_ARB_1	**00000000**

⇩

RX3_ARB_0	**11111111**
RX3_ARB_1	**11111111**

Operation 4

Table 46. write reg 01 H, data = 10Hex
(SETUP_1 register) AT7908E connected to BUS, remote frame request, standard message, length of data can be any value.

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
	1	0	0	0	0	0	0	0

⇩

SETUP_1	Disabled	TXRM	TXEM	TMRMR	TXDLC3	TXDLC1	TXDLC1	TXDLC0
	0	0	0	1	0	0	0	0

Transmission request

Operation 5 and 6

Table 47. write reg 03 H, data =0AHex
(SETUP_3 register) Resynchronisation jump length = 2 system clock and transmission request.

SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

⇩

SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

At this point, the AT7908E starts to transmit the remote frame request until an acknowledge comes from another CAN node.

In the next figure is reported the standard CAN format on the CAN BUS for the REMOTE FRAME request.

Figure 13. Remote frame

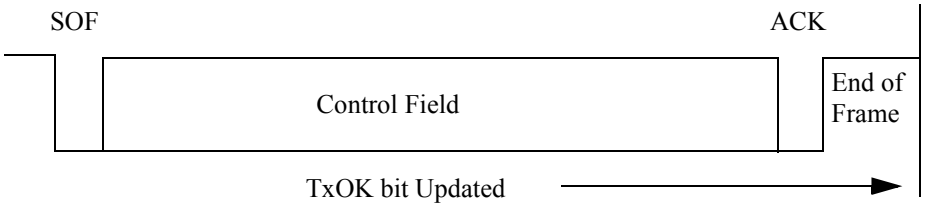


Table 48. STATUS register before the TX request:

STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	0	0	0

Table 49. STATUS register after start of transmission:

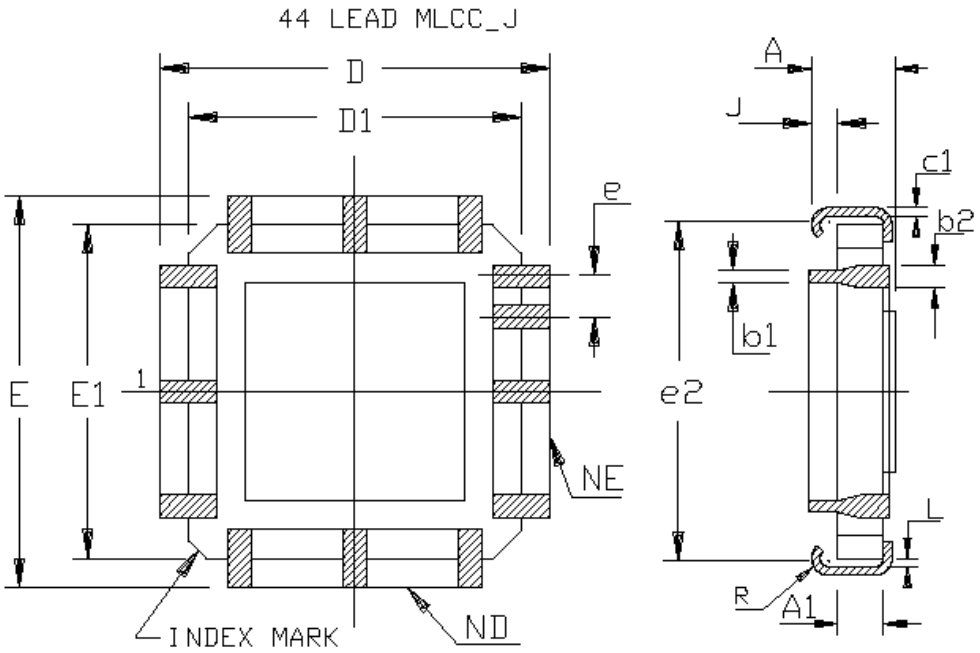
STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	0	1	0	0

Table 50. STATUS register after end of completed correctly transmission:

STATUS	SyncTx	SyncRx	Rxbuf1	Rxbuf0	TxOK	TxActive	ErrPass	BusOff
	0	0	0	0	1	0	0	0

At this point, the AT7908E waits for a remote frame answer. If the remote frame answer arrives, the behavior of the AT7908E, from all points of view, is the same as for a normal receiving message (see paragraph 6.2.3).

Package



	MM		INCH	
A	2.67	4.95	.105	.195
A1	1.65 NOM		.065 NOM	
b1	0.33	0.56	.013	.022
b2	0.55	0.88	.022	.035
c1	0.17	0.25	.007	.010
D/E	17.14	17.78	.675	.700
D1/E1	15.74	16.76	.620	.660
e	1.27 BSC		.050 BSC	
e2	16.00 BSC		.630 BSC	
L	0.12	-	.005	-
ND/NE	11		11	
R	0.50	1.01	.020	.040
J	0.58	----	.023	----

Mechanical Characteristics

44 Pin Multilayer Ceramic Package (MLCC)

Dimension: 17.14 x 17.14 mm

Pin to Pin Spacing : 1.27 mm

Pin Assignment

PIN Number	Signal Name	Note
1	VCCA1	Power for array
2	Addr[4]	
3	Addr[5]	
4	Addr[6]	
5	Addr[7]	
6	VSSB1	Ground for periphery
7	VCCB2	Power for periphery
8	Cs	
9	Mode	
10	Ale	
11	Wr	
12	VSSA1	Ground for Array
13	Rd	
14	Sena	
15	Test	
16	Reset	
17	VSSB2	Ground for periphery
18	VCCB3	Power for periphery
19	Can_rx	
20	Hasync	
21	----------	Not connected
22	Xtalout	
23	Xtalin	
24	VCCA2	Power for array
25	Int	
26	Hatrig	
27	Can_tx	
28	VSSB3	Ground for periphery
29	VCCB4	Power for periphery
30	Data[0]	
31	Data[1]	
32	Data[2]	
33	Data[3]	
34	VSSA2	Ground for Array
35	Data[4]	
36	Data[5]	
37	Data[6]	

PIN Number	Signal Name	Note
38	Data[7]	
39	VSSB4	Ground for periphery
40	VCCB1	Power for periphery
41	Addr[0]	
42	Addr[1]	
43	Addr[2]	
44	Addr[3]	

Note: VCCA1 = VCCA2 = VCCB1 = VCCB2 = VCCB3 = VCCB4 = 5V
VSSA1 = VSSA2 = VSSB1 = VSSB2 = VSSB3 = VSSB4 = 0V

Ordering Information

Part Number	Temperatue Range	Quality Flow
AT7908EJL-E	25°C	Engineering Sample
5962-03A0601QXC	-55°C to +125°C	QML Q
5962-03A0601VXC	-55°C to +125°C	QML V

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe

Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia

Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Data-com

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

e-mail

literature@atmel.com

Web Site

http://www.atmel.com

4268D–AERO–11/09 xM