Technical Specification

CASA2

AURELIA
Microelettronica S.p.A.

Via Giuntini 13 – 56023 – frazione Navacchio Cascina (PI) - ITALIA
Capitale sociale 350.000 int. vers.
Registro delle imprese di Pisa n° 2819/1998 REA 127299 C.F. P.I. n° 01428250508

INDEX

1 INTRODUCTION

1.1 Purpose

The aim of this document is to describe the functions and the specifications of CAN Asic for Space Application.

1.2 Reference Documents

[1] CAN Specification version 2.0 BOSCH, 1991.

[2] 82527 Serial Communication Controller – Architectural Overview – INTEL, 1996.

[3] ASIC Design and Manufacturing Requirements, WDN/PS/700, ESA, October 1994.

[4] MG2RTP Radiation Hardened 0.5 Micron Sea of Gates, TEMIC Semiconductors, July 2000

[5] CASA2 preliminary specification document Revision 1
(CASA2/PSD/01-04:CASA_PSD_rev1.pdf) – Aurelia Microelettronica – 27/07/2001

2 OVERVIEW

2.1 CASA2

FUNCTION CAN controller
TECHNOLOGY ATMEL MG2RTP: Radiation Hardened Sea of Gate 0.5 um
TYPE Semi-Custom – digital 5 V
Operating FREQ. 16 MHz
MAX FREQ. 18 MHz
Package MLCC 44 pins

PIN 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_add<7:0>	IO - CMOS		Address data bus
25	Int	O - CMOS	AL	interrupt request
27	cantx	O - CMOS		tx signal
19	canrx	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 synchronisation signal

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

CASA2 is a CAN controller stand-alone device for space application. Redundant structures and special techniques implemented in order to make the device SEU tolerant. CASA2 CAN core provides all CAN specification 2.0B protocol functions except overload frame generation. It includes acceptance filtering. The core incorporates error-handling capabilities, stuff bit generation, CRC, multiple sample points, remote frame generation. CASA2 provides a programmable MCU 8 bit general-purpose interface to connect receive and transmit buffer, control register and status register to CPU.

2.2 Oscillator

Xtalin and Xtalout are IO of an internal inverting oscillator. To use internal oscillator the quartz must be connected between Xtalin and Xtalout pad.

To drive the device with an external clock source Xtalin must be driven by clock signal

and xtalout must be left unconnected. The maximum work frequency of oscillator, in open-loop mode(without crystal) with 6pF Load on Xtalout at worst condition(Process slow, temperature =145°C, Power supply =4.5V), is 60MHz.

The IO level of Xtalin and Xtalout are CMOS level (see Paragraph 2.5).

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

2.3 RESET SPECIFICATION

Reset pin must be driven low for at least 3 clock cycle (190 ns at 16 Mhz)

2.4 OPERATING TEMPERATURE [4]

Ta -55°C to 125°C

2.5 ABSOLUTE MAXIMUM RATINGS[4]

Process: ATMEL MG2RTP 0.5µm, 3 metal CMOS sea of gates.

Ambient temperature under bias (TA)

ilitary –55 to +125..... . . . . .

Junction temperature TJ < TA + 20..... . . . . .

Storage temperature –65 to +150..... . . . . .

TTL/CMOS :

Supply voltage VDD –0.5 V to +6 V . . . . . .

I/O voltage –0.5 V to VDD + 0.5 V . . . . . .

NOTE:

Stresses above those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended period may affect device reliability.

2.6 DC Characteristics[4]

Specified at VDD = +5 V +/– 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 = –12, -6, -3 mA*
VOH	Output HIGH voltage CMOS Output	3.9			V	IOL = 12, 6, 3 mA*

IL	Input Leakage current NO Pull up/down Pull up Pull down	 -44 75	 +/- 1 -66 118	 +/- 5 -100 300	 µA µA	
IOZ	3-State Output Leakage current		+/-1	+/-5	µA	
IOS	Output Short circuit current IOSN IOSP			 48 36	 mA mA	BOUT12 VOUT = 4.5V VOUT = VSS

- **According buffer: Bout12, Bout6, Bout3, VDD = 4.5V**

2.7 AC Characteristics[4]

Tj = 25 °C, Process typical (all values in ns)

Output signals	**BUFFER Description**	LOAD	Transition TPLH	Transition TPHL
Int, cantx, hatrig, hasync,	BOUT6: Output buffer with 6 mA drive	75 pF	4,370	2,377
Data_add<7:0>,	BIOC6: Bi-directional Buffer CMOS input	75 pF	5,234	2,449

2.8 Power Consumption estimated

Parameter	**MAX**	**MIN**	**Note**
DC Current Dissipation	50 mA @5V	-	CLK frequency = 16 MHz

2.9 Technology[4]

MG2RTP 0.5 µm 3 Metal Layers Sea of Gate.

Matrix: MG2 –044: 33K usable gates.

MG2RTP is Radiation Tolerant technology:

- Latchup free
- SEU free
- Total Dose > 300 Krad

Qualify : QML -Q

Package: MLCC_J44: Ceramic Multylayer Package

2.10 APPLICATION

The core architecture has been implemented taking into account the typical constraints of a space application: total dose > 300 Krad, latch up free, SEU free design, sleep mode for

low power consumption, insertion of test structure (internal scan chain) to reach a fault coverage > 95% as ESA specification.

3 FUNCTIONAL DESCRIPTION

3.1 FEATURES

CASA2 CAN controller is designed in accord with CAN specification 2.0B and it supports the following features.

- ☞ Supports 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
- ☞ Supports CAN bus line arbitration
- ☞ Supports 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

3.2 Functionality

CASA2 is integrated device that performs serial communication according to CAN protocol. The CAN protocol uses multi-master bus configuration to transfer data packet between nodes on network.

CASA2 CAN controller supports both standard and extended message frame format as CAN Specification 2.0B. It can transmit and receive with 29 bit identifier and it can filter the first eleven bit of receiving message(filtering function is performed only on eleven bit).

CASA2 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 bit long), by data and by status register that store the status of receiver buffer. The identifier permits to filter the receiving message together with global mask that implement don't care condition.

A message is accepted and stored in RX message buffer only if identifier of incoming message (first eleven bits) matches the identifier in RX message buffer. If the receiving message matches more than one identifier, the message is stored in the lowest numbered message buffer. CASA2 implements a global acceptance-masking feature for

message filtering.

CASA2 was developed to have a programmable general-purpose MCU interface. The MCU interface permit to program the internal register of CASA2 and to read out the status of controller and the received messages. The schematic representation of CASA2 is shown in Fig. 1.

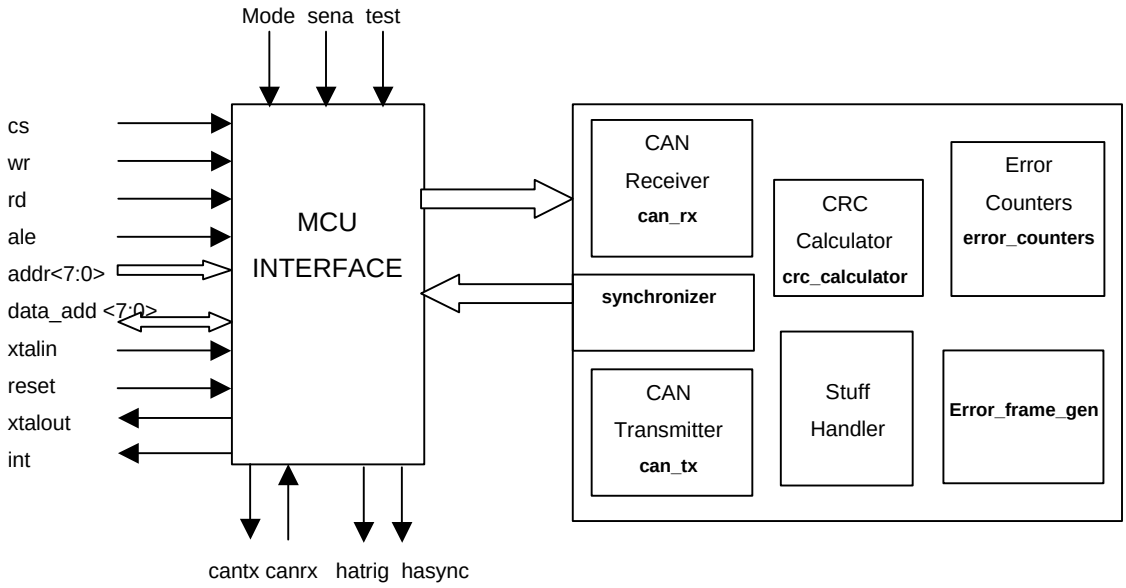


Fig. 1 - Block schematic of CASA2

To transmit a CAN message the MCU must write in the interface internal registers the data and configuration signals. The message bus (0 to 101) goes to **can_tx** that generates the output signal to the transceiver **cantx** 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 CRC calculated matches the CRC received.

The **error_counters** block is a counter that receives signals from **can_tx** and **can_rx** to calculate the error state of CAN. If the number of errors is higher than 127, CASA2 will be on error passive state. If error counters is higher than 255 CASA2 will be on bus-off state

The **synchronizer** block synchronises the controller with **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 bus.

In the next sections the interface block, the internal registers and the operational modes of CASA2 will be described.

3.2.1 Transmission of message

To transmit a message on bus the controller must program the configuration registers of CASA2 (interrupt generator configuration, data rate, bit timing configuration, message length) and the transmit buffer. After this the controller must write in internal register the bit (Txreq=1) to start transmission. At this point, the Transmit message buffer is sent to 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).

3.2.2 Incoming message

The incoming message passes through the global mask and it 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 receiver message objects will be updated with the information of 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 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 be not stored on RX message buffer.

3.2.3 Remote Frame

CASA2 CAN controller supports remote frame features.

To send on the bus a remote frame request MCU must write two internal bit of SETUP register (TMRMR = 1 and TXRM =0). The arbitration registers of transmitter message buffer must have the identifier for remote frame request. The start of remote frame request will be performed when transmission request is set by MCU on CASA2.

The CAN CASA2 that receive 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 answer to remote frame request. To answer at remote frame request CASA2 must be pre-programmed with two bit of internal register: TMRMR=0,TXRM=1. TX arbitration register must be initialised with the message identifier to which to answer and the transmitter data buffer must be loaded with the data to send as answer to remote frame request. The transmission request must be set.

Answer to remote request if

for i = 28 downto 18

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

3.2.4 Filtering of message

The filtering function is implemented with a global mask and the arbitration register of receiver message buffers. Global Mask permit to define don’t care bit on filtering operation. The incoming message ID (msg_ID(28:18) is checked (xor) with identifier stored in message buffers (RXn_ARB0, RXn_ARB1). The result is maskerable (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 incoming message is checked with identifier of 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-message buffer but the received message is stored on the lowest numbered message buffer.

3.2.5 Interrupt generation

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

- message correctly transmitted
- message correctly received (data frame, not remote frame reception).
- Overrun condition on rx message buffer
- Bus-off condition
- Error passive condition

In the SETUP register is possible to configure which kind of interrupt will be generated by CASA2 device.

3.2.6 Fault confinement and error counters

The two internal error counters are readable from external MCU to check the error status of CAN controller. The error counter according Fault Confinement rules of CAN Specification 2.0B exception included. This means that if there is only one node CASA2 connected on the BUS and this node start to transmit message, it reaches error passive condition but not bus-off condition because the transmitter error counter will be not more

incremented for acknowledge error when it is in error passive state.

3.2.7 Trigger match function

Trigger match function is a feature implemented to generate HATRIG signal (a pulse for a clock cycle) if the Identifier of incoming message matches the TRIGGER_MATCH register value. The incoming message identifier(msg_id28 down to msg_id18) is checked with TM28 downto TM18 bits of TRIGGER_MATCH registers. This function is independent from filtration of message and from interrupt generation. HATRIG signal, together with 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.

3.3 Detailed Pin Description

In this paragraph will be described the PIN of CASA2 can Controller:

MODE	Input pin to select operational mode of interface: mode = 0 : 8 bit data bus multiplexed with lowest 8-bit address(register mapped between 8000Hex and 804Chex) mode = 1 : 8 bit not multiplexed address data bus (register mapped between 00 Hex and 4C Hex)
CS	Chip select pin to write or read internal register. A high level of this pin enables the CPU to access CASA2.
WR	Pin to write internal register. A low level of this pin enables the writing of CASA2 register (if CS signal is HIGH).
RD	Pin to read internal register. A low level of this pin enables the readout of CASA2 register (if CS signal is HIGH).
ALE	Address-latch-enable. Used in mode=0.
XTALIN	Input pin for clock. XTALIN (with XTALOUT) are the crystal connection for internal oscillator.
XTALOUT	Input output pin. This pin could be used as input (together XTALIN) for internal oscillator or as clock output to drive CPU.
RESET	Reset signal for CASA2. To reset CASA2 this signal must be set low.
DATA_ADD<7:0>	Bi-directional bus: Multiplexed data/address bus in mode 0 or data bus in mode1.
ADDR<7:0>	High input address in mode0 or lowest input address in mode 1
INT	Output pin for interrupt request to MCU. This pin is active low (interrupt generation) and will be kept low until the MCU clear interrupt request on CASA2.
CANTX	Serial output pin to CAN transceiver(dominant = ‘0’, recessive = ‘1’)

CANRX	Serial input from CAN transceiver (dominant = '0', recessive = '1').
SENA	Input signal to enable scan-test (with test = 1) or to put CASA2 on power down mode (with test signal = 0). To use CASA2 in normal functional mode this signal must be logical 0.
TEST	Input signal to increase testability. This pin will be used in test modality. Test = 1 => test mode (used by foundry that execute test of devices. Test = 0 => functional mode (if sena =0) or power down mode (if sena =1).
HATRIG	Output signal that will be set high (duration of pulse = 13 clock cycles) if the received message arbitration match the TRIGGER_MATCH register (independently from the matching between arbitration of incoming message and identifier of 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 pulse = 13 clock cycles).

3.4 Internal register description

In the next table all the registers of CASA2 are described:

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

3.4.1 SETUP registers

The five 8 bit 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, CASA2 generates interrupt on INT line after successful transmission.

***Rxint*:** active high, CASA2 generates one interrupt on INT line after successful data frame reception.

***Overint*:** active high, CASA2 generates interrupt on INT line if receiver buffer (number n)

has not been cleared(with RxClr) before the next valid message for the same receiver buffer.

Errint**:** active high, CASA2 generates interrupt on INT line if there's an error condition on CAN Controller(bus-off or error-passive).

GensyncRx**:** CASA2 generates a pulse on hasync output signal when a receiving message is detected.

GensyncTx**:** CASA2 generates a pulse on hasync output signal when a sync pulse on transmitting message is detected.

BPR0-BPR1**:** 2 bit to program CAN clock prescaler (4 different system clock of CAN Core). In the next table, it is possible to see the different values of system clock depending by bit 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 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 answer to 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 resynchronisation jump (correct value 1:4). The default value is 2.

Txreq: when this bit is set high, CASA2 sends on CAN bus the message contained in TX

buffer. This bit must be written from MCU to start transmission.

AbortTx: this bit must be set to inform 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 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 all RXOVRn bits on the status register of receiver message buffers(STATUS_RX).

3.4.2 SETUP_RX Register

RxClr3: active high, this bit is set to clear RXOK3 and RXOVR3 bit on the status register of receiver message buffer(STATUS_RX). This bit must be set after the read out operation of 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 RXOK2 and RXOVR2 bit on the status register of receiver message buffers. This bit must be set after the read out operation of 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 RXOK1 and RXOVR1 bit on the status register of receiver message buffers. This bit must be set after the read out operation of received message otherwise at the next reception on the same RX message buffer an overrun condition is raised.

3.4.3 STATUS Registers

STATUS_0 :

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

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

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

TxOK: this bit indicates that transmission has been successfully terminated. This bit is reset automatically when TXREQ bit on 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 (new message received on message buffer 3 withRXOK3 not cleared).

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

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

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

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

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

3.4.4 FILTER Registers

The 2 filter registers are used (together with arbitration register of receiver buffers or arbitration registers of transmit buffer to answer to remote request) to filter out 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 message. 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 arbitration of receiver buffer bits are the corresponding bit on Acceptance Mask only if set to 1.

This filtering function is implemented for every message buffer.

3.4.5 TX Message Buffer

CASA2 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 standard message CASA2 will send on the bus the identifier from ARB28 to ARB18 The others 8 registers contain the data byte of the

message:

TX_ARB0 :bit 28 down to 21 of arbitration of transmitter buffer (TXARB28:TXARB18)

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

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

TX_ARB3 :bit 7 down to 3 of arbitration of transmitter buffer (TXARB4:TXARB0)

TX_Message_0 :first 8 bit data of transmitter buffer

TX_Message_1 : second 8 bit data of transmitter buffer

TX_Message_2 : third 8 bit data of transmitter buffer

TX_Message_3 : fourth 8 bit data of transmitter buffer

TX_Message_4 : fifth 8 bit data of transmitter buffer

TX_Message_5 : sixth 8 bit data of transmitter buffer

TX_Message_6 : seventh 8 bit data of transmitter buffer

TX_Message_7 : eighth 8 bit data of transmitter buffer

3.4.6 RX message buffers

CASA2 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 message object, the message will be stored in the lowest numbered message object. The incoming remote request is not stored on message buffer.

The structure of message buffer is:

RXn_STATUS : Status register of message object n

RXn_ARB0 : bit 28 down to 21 of arbitration of message object n

RXn_ARB1 : bit 20 down to 13 of arbitration of message object n

RXn_ARB2 : bit 12 down to 5 of arbitration of message object n

RXn_ARB3 : bit 7 down to 3 of arbitration of message object n

RXn_Message_0 : first 8 bit data of message object n

RXn_Message_1 : second 8 bit data of message object n

RXn_Message_2 : third 8 bit data of message object n

RXn_Message_3 : fourth 8 bit data of message object n

RXn_Message_4 : fifth 8 bit data of message object n

RXn_Message_5 : sixth 8 bit data of message object n

RXn_Message_6 : seventh 8 bit data of message object n

RXn_Message_7 : eighth 8 bit data of message object n

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

RXn_DLC3:RXn_DLC0 : length of 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.

3.4.7 Error Counters registers

CASA2 has two internal counters for RX error and TX error. The values of these counters are stored in two registers that can be read from MCU.

3.4.8 Trigger Match registers

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

3.5 Bit Timing

A bit period is composed of the following three segments:

- Synchronisation segment
- Timing segment 1
- Timing segment 2

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

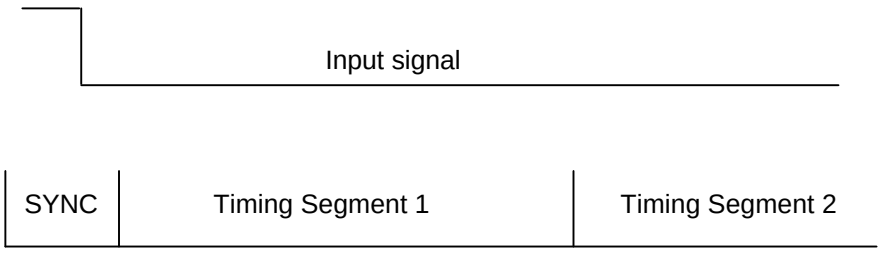


Fig. 2 Bit Time Segments

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

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.

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 internal processing after sample point.

resynchronisation Jump Width is used to compensate phase shifts between oscillator frequency of different CAN nodes on 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 resynchronisation. If the falling edge of incoming signal is on TIMING segment 1 then the Bit period is lengthened (the sample point will be at TSEG1 +RSJ). If the falling edge of incoming signal is on Timing segment 2 then the bit period will be shortened (TSEG2 is shortened of RSJ value). The resynchronisation mechanism is shown in fig. 3 and fig. 4.

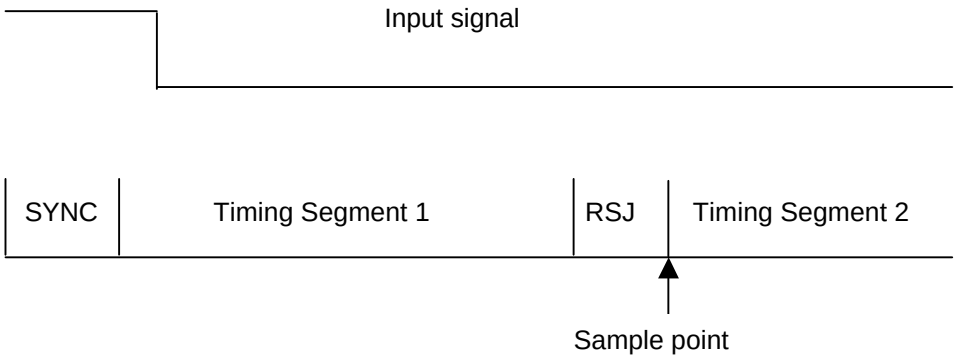


Fig. 3 Lengthening a Bit period

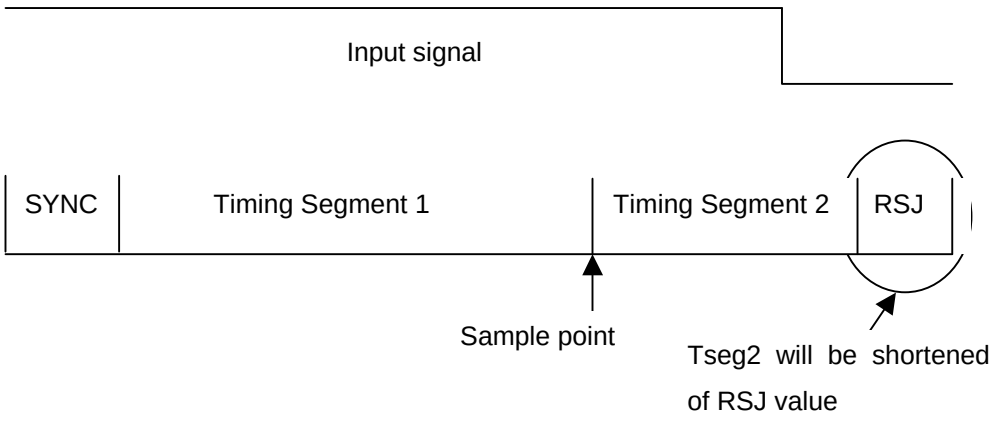


Fig. 4 Shortening a Bit period

The Bit rate of message on bus will be:

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

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

TSEG1 and TSEG2 length must be programmed to respect these conditions:

TSEG1 ≥ **TSEG2 => PS1 +1** ≥ **PS2**

TSEG2 ≥ **RSJ => PS2** ≥ **RSJ**

4 INTERFACE BLOCK DESCRIPTION

CASA2 provides a programmable (with external pin) MCU interface. Two modes can be selected. The first operational mode is interface with 8-bit multiplexed address data bus (mode = 0) and internal register addressable with 16-bit of address. In this operational mode, CASA2 registers are mapped between 8000Hex and 804CHex.

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

The pins for interface block are:

mode	Selection of interface modality. This pin must be **0** to use 8 bit multiplexed data address (lowest) to mapping CASA2 on high address space (8000Hex to 804C Hex) If mode = 1 data and addresses are not multiplexed and the registers are mapped on lowest address space.
Data_addr <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 internal registers access (active high).
WR	Input Write signal to write internal registers (active low).
RD	Input Read signal to readout internal registers (active low).
Addr <7:0>	Input highest address in **mode 0** (used with low address to map CASA2 register between 8000Hex and 804Chex address space) or input lowest address in **mode 1**

4.1 Operational mode 0

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

The bus Data_addr <7:0> must be connected to data/address bus of MCU. The ALE signal is used to latch address. This latched address will be the address of CASA2 internal registers. The bus addr (7:0) will contain the highest address generated by MCU. The internal CASA2 registers will be accessed only if 16-bit address generated by MCU will be between 8000 Hex and 804C Hex (internal Chip select will be generated). Moreover to write in register or to read from register the microcontroller must generate CS signal (active high) WR, RD and ALE signals and must drive Data_addr bus and addr bus. The signal *wr* must be low for at least 3 clock cycle. The data is latched at the rising edge of clock when *cs* = 1, *wr* = 0 and *rd* = 1. In Fig. 5 is reported a write cycle.

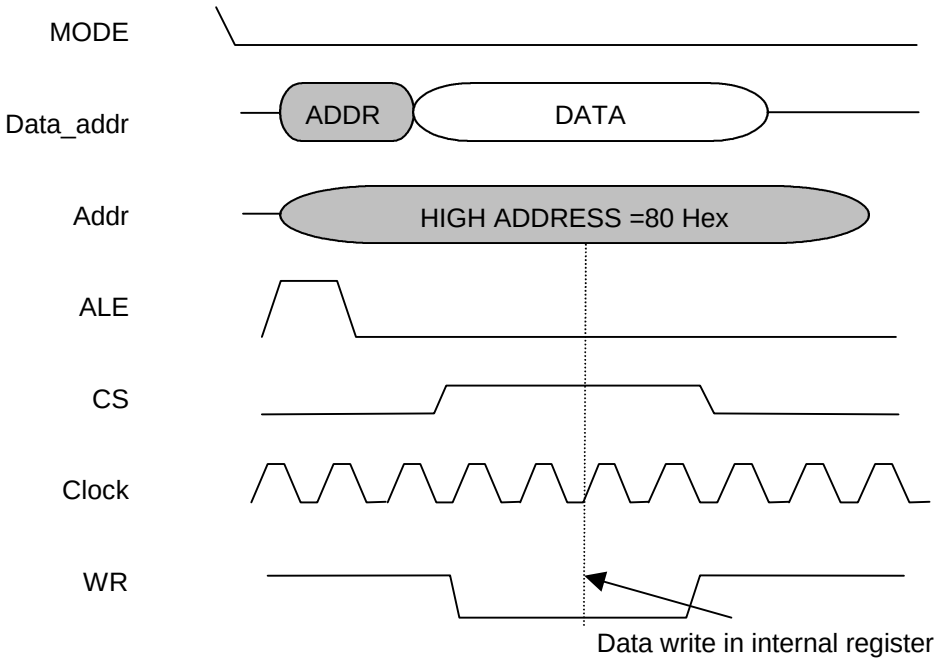


Fig. 5 Mode 0 write cycle

In Fig. 6 are represented the signals to readout internal register of CASA2 in mode 0

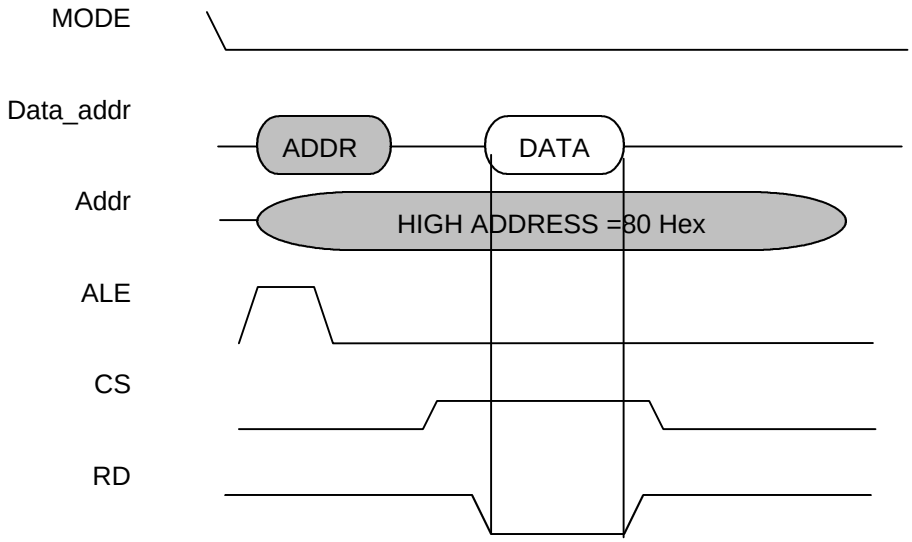


Fig. 6 Mode 0 read cycle

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

4.2 Operational mode 1

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

This interface operational mode is used to write CASA2 registers and to readout from CASA2 with two different 8-bit data and address bus. ALE input pin is not used and the internal registers are mapped between 00Hex and 4C Hex. Addr bus will be used as 8 bit address for internal registers**.** To write data on internal registers the MCU must control the *cs*, *wr* and *rd* signals of CASA2 and must drive addr bus and data bus. The signal *wr* must be low for at least 3 clock cycle. The data is latched at the rising edge of clock when *cs* = 1, *wr* = 0 and *rd* = 1. The fig. 7 shows the write cycle in operational mode 1

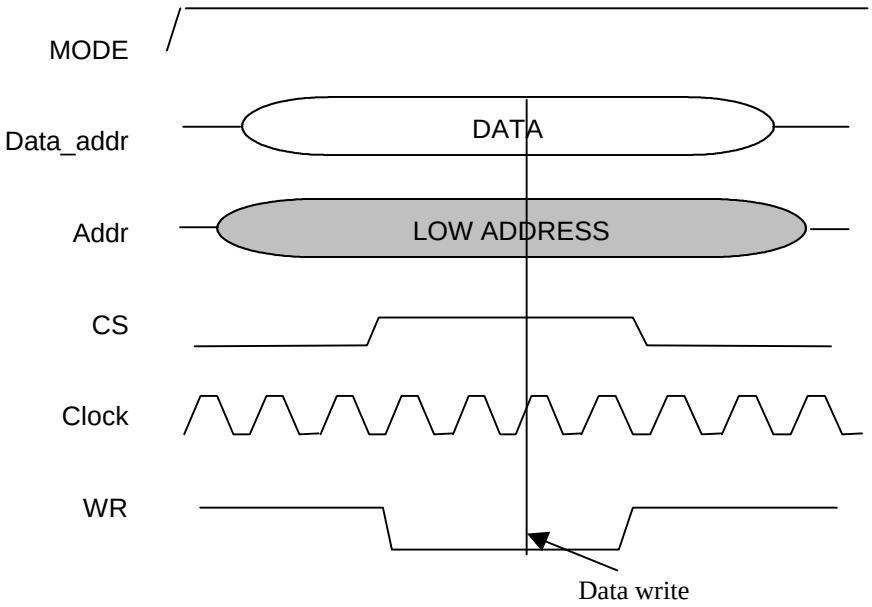


Fig. 7 Mode 1 write cycle

Fig. 8 shows a read cycle. To readout data from CASA2 internal registers, MCU must drives cs=1, rd=0, wr=1.

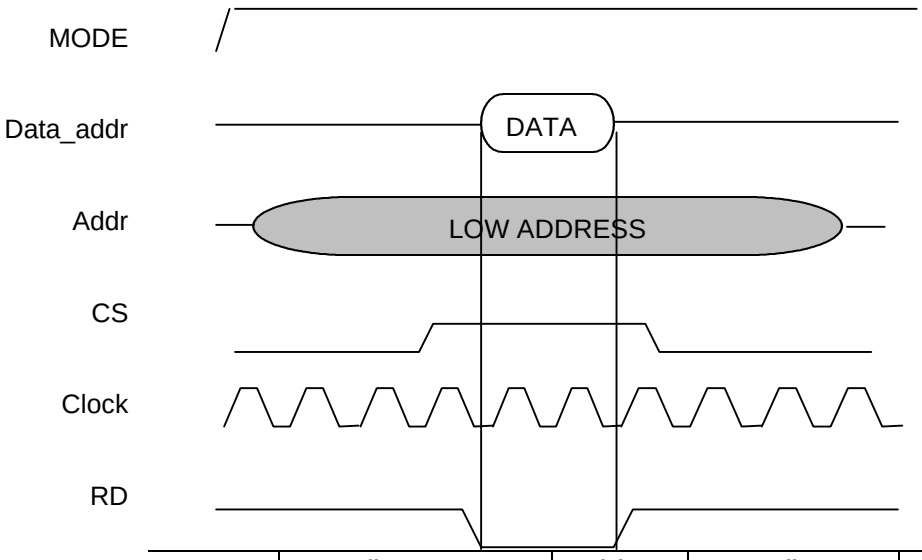


Fig. 8 Mode 1 read cycle

4.3 TIMING

In this paragraph are provided the AC specification for CASA2 interface signals in both operational mode.

4.3.1 AC specification for 8 bit multiplexed mode (mode =0)

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

Symbol	Parameter	Min	Max	Note
t_{AVLL}	Address valid to ALE low	2 ns		Mode 0
t_{LLAX}	Address Hold after ALE low	1 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	0 ns		Mode 0
t_{CHWL}	CS high to WR low	0 ns		Mode 0, mode 1
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_{WHCH}	WR high to CS high	0 ns		Mode 0, mode 1
t_{WS}	WR setup time before clock	5 ns		Mode 0, mode 1
t_{WH}	WR hold time after clock	1 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	18 ns	Mode 0, mode 1

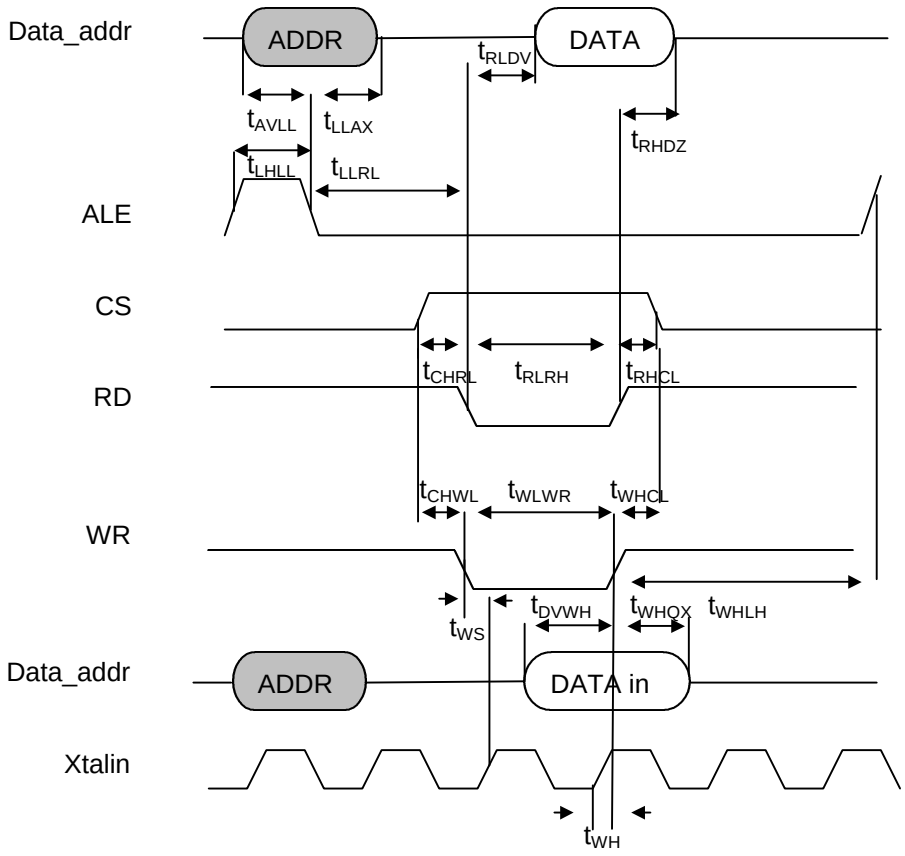
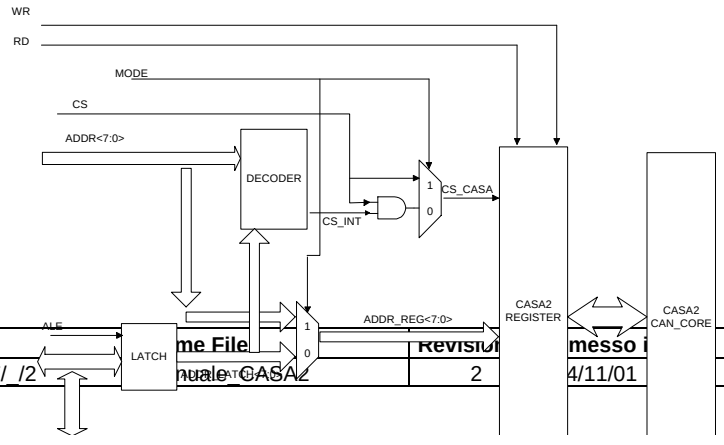


Fig. 9 Timing information for interface signals

4.4 Interface internal structure

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

In the following Figure is possible to see the realisation of the two operational modes and



the access to CASA2 internal registers.

Fig. 10 Interface block scheme

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

The external data bus is connected to internal latch block of CASA2. **ALE** signal latches and extracts the **ADDR_LATCH** (lowest 8-bit address). **ADDR_LATCH** is used to address internal registers in mode 0 and to generate, with **ADDR<7:0>** (highest 8-bit address), internal chip select signal (**CS_INT**). In mode 0 the access to 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 internal register is **ADDR<7:0>**. The two different operational modes are established by **mode** pin that is the selector for the two multiplexer in Figure 5. In Figure 5 are reported moreover the other two blocks of CASA2 device: CASA2 registers and CASA2 CAN_CORE.

4.4.1 Address table

In the next table is possible to see how MCU can address RAM or internal register of CASA2 in operational mode 0

ADDRESS generated by MCU (15 down to 0)	Device
0000 – 7FFF Hex	32 Kword of External RAM
8000 – 800C Hex	CASA2 internal register 00 – 0C Hex (Configuration and status register)
8010 – 801B Hex	CASA2 internal register 10 – 1B Hex Transmitter message buffer
8020 – 802C Hex	CASA2 internal register 20 – 2C Hex First receiver message object
8030 – 803C Hex	CASA2 internal register 30 – 3C Hex Second receiver message object
8040 – 804C Hex	CASA2 internal register 40 – 4C Hex Third receiver message object

5 OPTIONAL FEATURES

5.1 Internal clock frequency

The actual internal system clock could be: external clock divided by 1, 2, 4, and 8. If the bit time length will be (default configuration)16 system clock pulse(see PS1, PS2 programmable bit timing registers) is possible to have four different data rate depending by prescaler programming.

5.2 CASA2 operational mode

CASA2 device could be put on three different operational mode:

Functional Mode

Test mode

Power down mode

5.2.1 Functional mode

This mode is the normal operational mode for CASA2 device. The two input pins **test** and **sena** must be put on logical value 0.

5.2.2 Test mode

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

5.2.3 Power done mode

This mode could be used to put CASA2 device on sleep mode (internal clock is off). To put CASA2 in this mode **test** input pin must be 0 and **sena** input pin must be 1

6 APPLICATION NOTE

6.1 Registers value after reset

In the next table are reported the registers value after reset of CASA2 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							
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**							
RX3_MESSAGE_7	**4B**	**00000000**							
RX3_STATUS	**3C**	reserved	reserved	Reserved	**RX3_extfr**	**Rx3_DLC3**	**Rx3_DLC2**	**Rx3_DLC1**	**Rx3_DLC0**
		X	X	X	**0**	**0**	**0**	**0**	**0**

6.2 Configuration flow

The following flow diagram is intended to understand the action that MCU must perform to program CASA2 CAN controller to send or to receive CAN message.

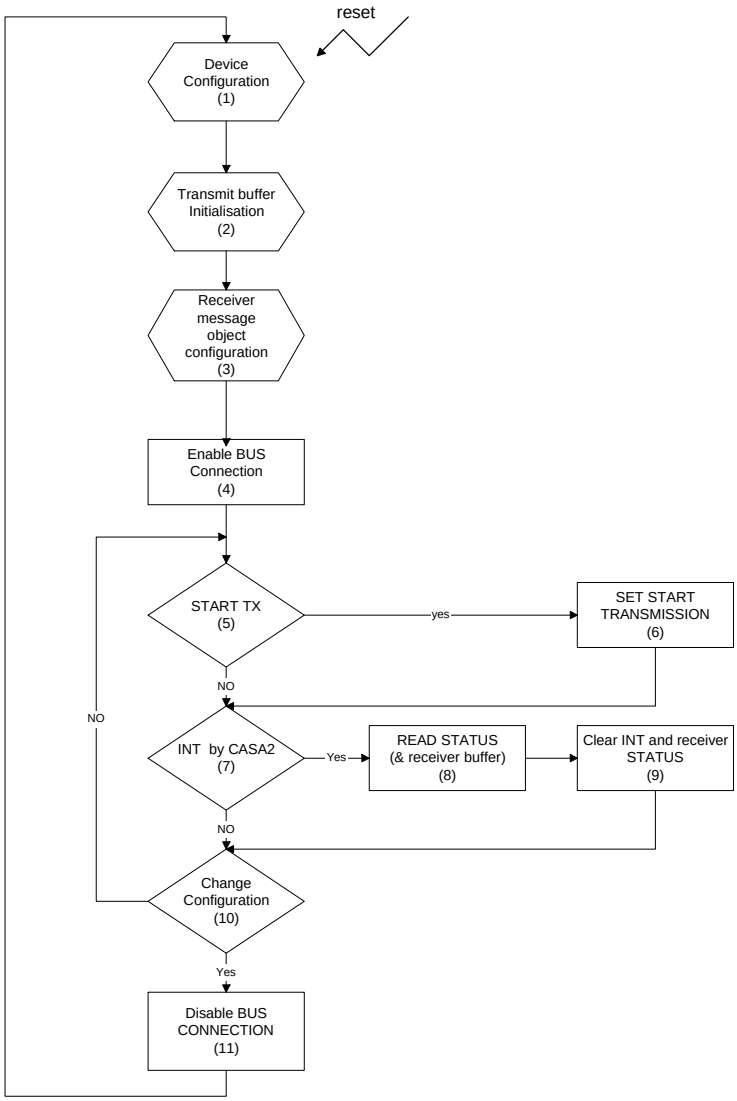


Fig. 11 operating flow diagram

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

6.2.1 Initialisation and transmission of DATA FRAME with INT generation and behaviour of MCU after interrupt received

Configuration of all setup registers after RESET signal.

1

write reg 00 H, data =01Hex

(SETUP_0 register) system clock = external clock, enable transmission 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	0	1

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)

After SETUP registers is necessary to configure TX message object:

2

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

write reg 14 H, data =00Hex

TX_MESSAGE_0	00000000

⇨

TX_MESSAGE_0	00000000

write reg 15 H, data =FF Hex

TX_MESSAGE_1	00000000

TX_MESSAGE_1	11111111

write reg 16 H, data =0FHex

TX_MESSAGE_2	00000000

⇨

TX_MESSAGE_2	00001111

2

write reg 17 H, data =F0Hex

TX_MESSAGE_3	00000000	⇨	TX_MESSAGE_3	11110000

write reg 18 H, data =AA Hex

TX_MESSAGE_4	00000000	⇨	TX_MESSAGE_4	10101010

write reg 19 H, data =55Hex

TX_MESSAGE_5	00000000	⇨	TX_MESSAGE_5	01010101

write reg 1A H, data =66Hex

TX_MESSAGE_6	00000000	⇨	TX_MESSAGE_6	01100110

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

After TX message object initialisation MCU must connect CASA2 to CAN bus.

4

write reg 01 H, data =08Hex

(SETUP_1 register) CASA2 connected to BUS, data frame(no remote frame), 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	0	0	0	1	0	0	0

Setting of transmission request

5
6

write reg 03 H, data =0Ahex[1]

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

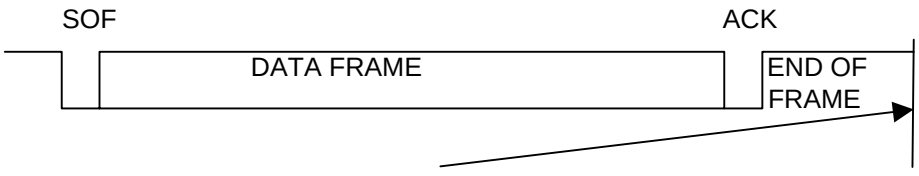
SETUP_3	RxClr	Reset	IntClr	AbortTx	Txreq	RSJ2	RSJ1	RSJ0
	0	0	0	0	0	0	1	0

⇩

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

At this point CASA2 start to transmit the message until an acknowledge come from another CAN node.

In the next figure is reported the standard CAN format on CAN BUS:



[1] The five Gray bit are not updated after setting (in this case Txreq). These bit are not written inside the registers.

INT generation

Fig. 12 DATA frame

STATUS register before TX request:

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

STATUS register after start of transmission:

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

STATUS register after end of completed correctly transmission:

7

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

At this point CASA2 will send interrupt generation ;INT signal (Active low).

The interrupt generation will be cleared by MCU setting IntClr bit on SETUP_3 register.

Now we suppose that MCU receives INT signal and it wants program CASA2 to send the same data frame on CAN bus.

8

read reg 05 H

(MCU read status register after INT received)

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

9
10
5
6

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	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 INT signal become high(interrupt request disabled), CASA2 start new transmission and STATUS register will be automatically updated as following:

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

⇩

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

6.2.2 Initialisation of receiver message objects and filtering function.

In this paragraph is reported an example about configuration of CASA2 to receive message with filtering function. In this example MCU, after configuration of CASA2, wait for Interrupt (from CASA2)and then MCU read status registers, RX message object and it clear interrupt request and receiver status.

Configuration of CASA2:

1

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	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	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

3

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

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

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

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

4

write reg 01 H, data =00Hex

(SETUP_1 register)CASA2 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 CASA2 is ready to receive message from other CAN nodes and to generate interrupt if the message has been received.

When CASA2 receive a message, (for example with Identifier = 11111111111) it stores message in the receiver message object that satisfy filtering function(in this case RX3 message object). In the following tables are reported the STATUS registers before and after message receiving(Rxbuf0 and Rxbuf1 bit, after received message, contain the indication about the receiver message object that stored the message incoming).

STATUS register before receiving.

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

⇩

STATUS register after received message

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

5
7

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

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 RX3 message object will store the arbitration of received message and the data registers of RX3 message object will store the data of received message. At the end of frame the INT will be generated and the action that MCU must perform could be the followings (These operation are only read operation and the internal registers of CASA2 are not updated):

8

read reg 06 H

(STATUS_RX register)readout of CASA2 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 read out of received message, the Microcontroller must clear receiver message object status(principally Rxok bit) and INT signal setting RXCLRn bit in SETUP_RX register and IntClr bit in SETUP_3 register(see how to clear interrupt in the precedent paragraph).

5
6

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

9

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 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 Microcontroller don't clear receiver message object status(principally Rxok bit) and CASA2 receive a new message on the same message object (for example 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

6.2.3 Setting to answer to remote frame request

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

1

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 SETUP registers is necessary to configure TX message object:

2

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

write reg 14 H, data =00H

TX_MESSAGE_0	00000000

⇨

TX_MESSAGE_0	00000000

2

write reg 15 H, data =FF Hex

TX_MESSAGE_1	00000000	⇨	TX_MESSAGE_1	11111111

write reg 16 H, data =0FHex

TX_MESSAGE_2	00000000	⇨	TX_MESSAGE_2	00001111

write reg 17 H, data =F0Hex

TX_MESSAGE_3	00000000	⇨	TX_MESSAGE_3	11110000

write reg 18 H, data =AA Hex

TX_MESSAGE_4	00000000	⇨	TX_MESSAGE_4	10101010

write reg 19 H, data =55 Hex

TX_MESSAGE_5	00000000	⇨	TX_MESSAGE_5	01010101

write reg 1A H, data =66Hex

TX_MESSAGE_6	00000000	⇨	TX_MESSAGE_6	01100110

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.

4

write reg 01 H, data =48Hex

(SETUP_1 register) CASA2 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 remote frame request MCU must set Txreq bit on SETUP_3 register. After Txreq setting, CASA2 wait for remote frame request that match (see filtering functionality) the identifier stored on TX_ARB registers.

5
6

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 CASA2 wait for remote frame request and STATUS register will be not updated.

If the request arrives, CASA2 will send acknowledge and then start to transmit data frame as answer to remote frame request. 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 transmission CASA2 will send INT to MCU that can read out the STATUS register to check that all is OK.

6.2.4 Setting to send remote frame request

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

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

Configuration of all setup registers after RESET signal.

1

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 SETUP registers is necessary to configure TX message object:

2

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

3

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_0	11111111
FILTER_AM_1	00000000		FILTER_AM_1	11111111

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_0	01010101
RX1_ARB_1	00000000		RX1_ARB_1	01010101

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_0	10101010
RX2_ARB_1	00000000		RX2_ARB_1	10101010

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_0	11111111
RX3_ARB_1	00000000		RX3_ARB_1	11111111

4

write reg 01 H, data =10Hex

(SETUP_1 register) CASA2 connected to BUS, remote frame request, standard message, length of data can be any value .

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

Transmission request

5
6

write reg 03 H, data =0AHex

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

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

At this point CASA2 start to transmit remote frame request until an acknowledge come from another CAN node.

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

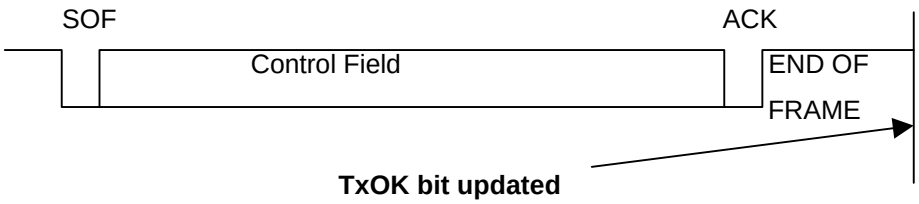


Fig. 13 Remote frame

STATUS register before TX request:

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

STATUS register after start of transmission:

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

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 CASA2 wait for remote frame answer. If remote frame answer arrives the behaviour of CASA2 from all point of view is the same of normal receiving message (see paragraph 6.2.3).

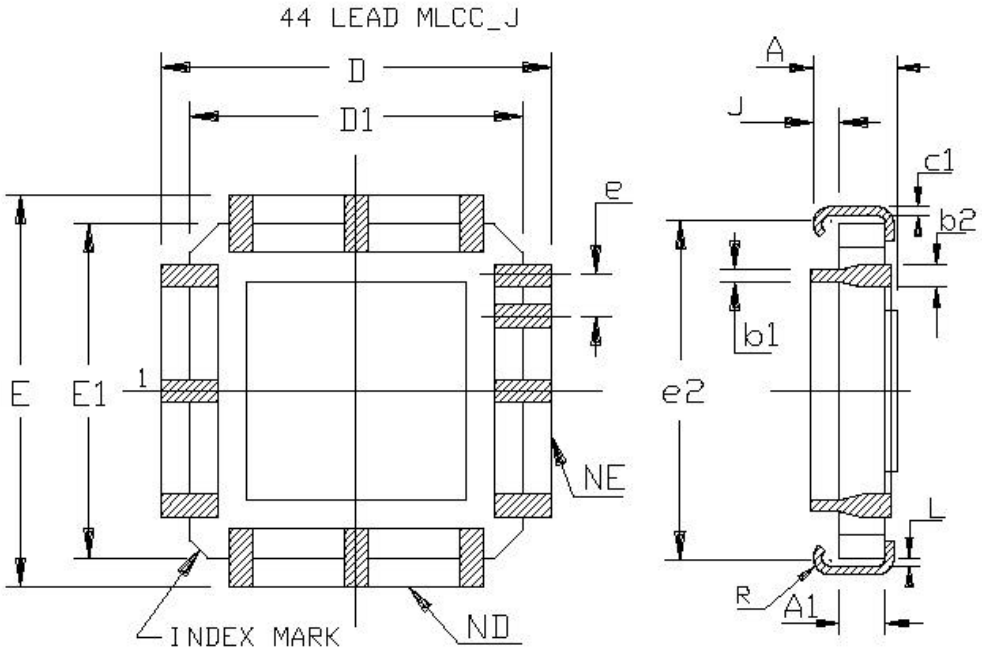
7 PACKAGE

7.1 Mechanical characteristics

44 Pin Multilayer Ceramic Package (MLCC)

Dimension: 17.14 x 17.14 mm

Pin to Pin Spacing : 1.27 mm



	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	----

7.2 Thermal Characteristics

Thermal resistance (P=1-2Watt, Ta =25°C)

Still air	:	Rθja = 59.5 °C/W
Forced air	:	Rθja = 43.7 °C/W

http://www.aurelia-micro.it mailto:contactus@aurelia-micro.it Tel. +39.050.754260 Fax +39.050.754261

8 CASA2 PINOUT

8.1 Pin Assignment

In the following table there are the PIN assignments for CASA2.

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]	
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]	

VCCA1 = VCCA2 = VCCB1 = VCCB2 = VCCB3 = VCCB4 = 5 V

VSSA1 = VSSA2 = VSSB1 = VSSB2 = VSSB3 = VSSB4 = 0V

8.2 Bonding Diagram

In Fig. 14 is reported the CASA2 device bonding diagram.

Fig. 14 CASA2 Bonding Diagram

http://www.aurelia-micro.it mailto:contactus@aurelia-micro.it Tel. +39.050.754260 Fax +39.050.754261

CONTACT US

Aurelia Microelettronica – CAEN Microelettronica

operative office

Via Giuntini 13 - 56023 Navacchio PI - ITALY

Tel. +39.050.754260 - Fax +39.050.754261

E-mail: info@aurelia-micro.it URL: http://www.caen.it/micro

CAEN headquarters

Via Vetraia 11 - 55049 Viareggio LU - ITALY

Tel. +39.0584.388431 - Fax +39.0584.388959