# *TMS470R1x Class II Serial Interface (C2SI) Reference Guide*

Literature Number: SPNU198B
September 2002

Texas Instruments

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of that third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

**REVISION HISTORY**

| REVISION | DATE | AUTHOR | NOTES |
|---|---|---|---|
| B | 9/02 | B.J. Jackson | Converted to a stand-alone book |
| A | 1/01 | R. Haley | *Pre-dates revision history* |
| * | 10/98 | R. Haley | *Initial version* |
| | | | |
| | | | |
| | | | |

# Contents

# Figures

# Tables

# Class II Serial Interface (C2SI)

The class II serial interface (C2SI) is a communication module used for transmitting and receiving data over a multi-master network. The C2SI module is the interface from the digital logic of the TMS470R1x family of microcontrollers to an external, analog interface chip. Class II communications follow the J1850 Class B protocol established by the Society of Automotive Engineers (SAE).

**Topic** **Page**

# 1 Overview

The class II serial interface (C2SI) is designed to handle all of the class II digital logic functions and operations between the host CPU and the class II bus interface (analog interface).

The C2SI module has the following features:

- ❏ Three external device pins
  - ■ C2SITX (C2SI transmit data output)
  - ■ C2SIRX (C2SI receive data input)
  - ■ C2SILPN (C2SI loop-back enable)
- ❏ Two selectable data rates
  - ■ Normal mode: 10.4 Kbps
  - ■ 4X mode: 41.6 Kbps
- ❏ Five error detection flags
  - ■ Break detect error
  - ■ Overrun error
  - ■ Incomplete byte error
  - ■ Bit timing error causing data to be corrupted
  - ■ CRC error
- ❏ Double-buffered receive and transmit functions
- ❏ Separate transmitter and receiver interrupts that can be interrupt driven or polled through the use of status flags
- ❏ Enable bits for interrupts
- ❏ Automatic CRC generation
- ❏ Sleep mode
- ❏ Automatic calibration with external analog interface via C2SILPN pin

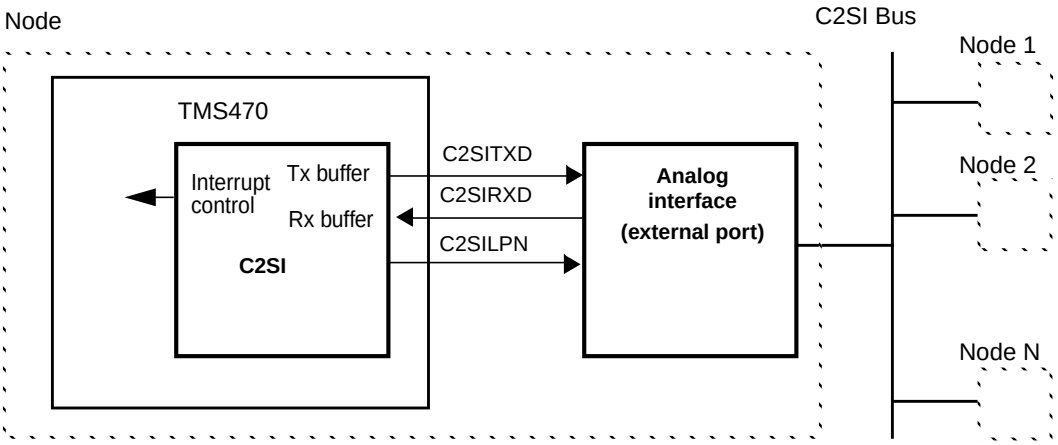
# 2 Functional Description of the C2SI

The C2SI is contained within the TMS470. The C2SI connects externally to an analog interface chip which is outside of the TMS470 as shown in Figure 1 This analog interface chip is required and acts as a buffer between the digital signals of the C2SI and analog signals recognized by the C2SI bus. The analog interface chip can be implemented with either a commercially available chip, such as the Harris HIP7020, or can be designed by the end-user.

The C2SI is a serial interface that supports SAE J1850 class B protocol with selectable data transmissions at either normal or 4X operational mode. The C2SI's receiver and transmitter are double buffered, and each has its own interrupt flags and bits.

Each individual C2SI on the bus is referred to as a node as shown in Figure 1. For information on the flow of data through the C2SI to bus interface pins and the related control pin registers see section 10.10, *C2SI Pin Control Register 1 (C2SIPC1)*, on page 48.

*Figure 1. C2SI Pin Connection Diagram*



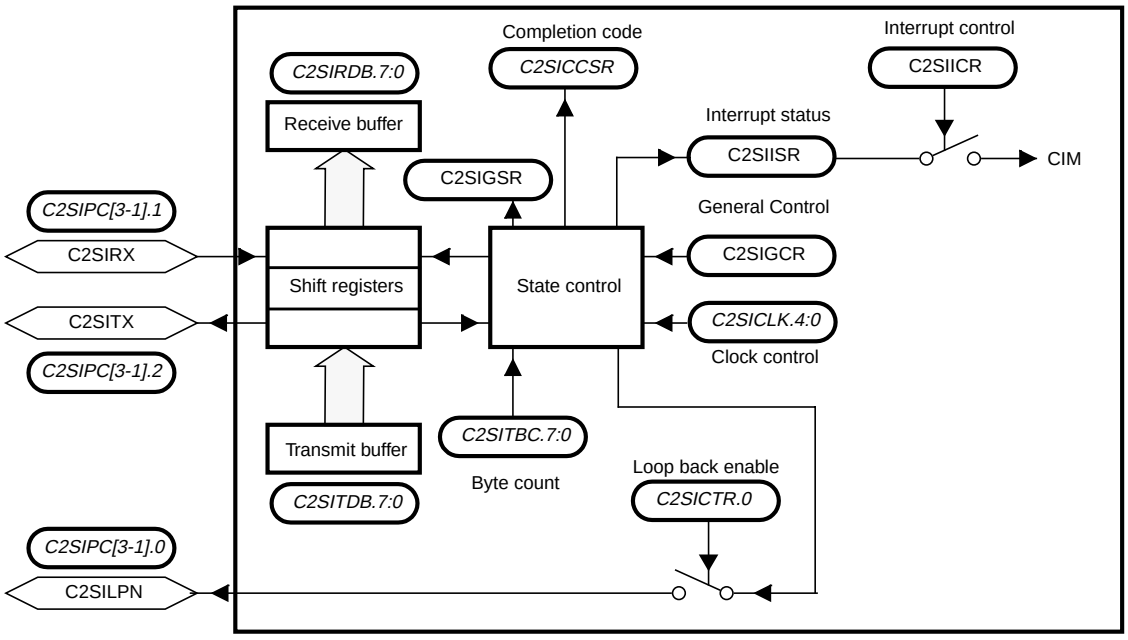

The C2SI bus is a serial data communications link. The nondestructive contention protocol of the Class II bus requires that there be an active (high) voltage and a passive (low) voltage state of the bus. The function of the analog interface is to actively drive the bus to a high voltage when signaled by the C2SI, and passively let an RC network pull the bus down to a low voltage. It also monitors the class II data-bus state for received data that is transferred to the C2SI. The bus is a wired OR arrangement.

The class II bus protocol includes a sleep mode in which any nodes can remain in a low-power standby condition until a node goes active and starts to send a message. The active state is detected by the others, which wake up in time to receive the message if voltage and clock signal are available to the analog interface.

The class II bus is intended to work in a relatively noisy environment. The analog interface (i.e. the Harris HIP7020) is required to filter out the high frequency noise. However, low-frequency noise is generally caused by ground offset between the nodes. The C2SI includes a digital filter to remove the low frequency noise.

For a detailed description of the operation of the C2SI bus, refer to the SAE J1850, Class B Data Communications Network Interface specification.

*Figure 2. C2SI Block Diagram*




## 2.1 C2SI Internal Registers

A general representation of the C2SI internal registers is shown in Table 1 For a more detailed description of the individual bytes, see section 10, *C2SI Internal Registers*, on page 30.

*Table 1. C2SI Internal Registers*

| Address Offset † | Mnemonic | Name | Description | Page |
|---|---|---|---|---|
| 0x00 | C2SIISR | C2SI Interrupt Status Register | Contains transmit/receive interrupt status flags | 30 |
| 0x04 | C2SIICR | C2SI Interrupt Control Register | Contains transmit/receive interrupt enable control bits | 35 |
| 0x08 | C2SIGSR | C2SI Global Status Register | Contains bus status flags | 37 |
| 0x0C | C2SIGCR | C2SI Global Control Register | Contains control bits for initiating and controlling transmissions | 38 |
| 0x10 | C2SITDB | C2SI Transmit Data Buffer | Contains data bits to be transmitted out of the C2SITXD pin | 41 |
| 0x14 | C2SICCSR | C2SI Completion Code Status Register | Contains read-clear transmit/receive completion status flags | 41 |
| 0x18 | C2SICTR | C2SI Control Register | Contains read/write bits for enabling control functions | 45 |
| 0x1C | C2SICLK | C2SI Interface Clock Register | Set to the frequency of the interface clock | 46 |
| 0x20 | C2SITBC | C2SI Transmit Byte Counter | Determines the number of bytes to be transmitted | 47 |
| 0x24 | C2SIPCI | C2SI Pin Control Register 1 | Determines if individual pins are used as general I/O or C2SI function pins | 48 |
| 0x28 | C2SIPC2 | C2SI Pin Control Register 2 | Determines the value of the general I/O output | 50 |
| 0x2C | C2SIPC3 | C2SI Pin Control Register 3 | Reflects the value on the pins | 51 |
| 0x30 | C2SIEMU | C2SI Emulation Buffer Register | Mirror of C2SIRDB, but read does not clear interrupt | 51 |
| 0x34 | C2SIRDB | C2SI Receive Data Buffer | Contains the current data from the receiver shift register | 51 |

† The actual address of these registers is device specific and CPU specific. See the specific device data sheet to verify the C2SI register addresses.

# 3 Data Format

The C2SI receive and transmit data formats, shown in Figure 2 and Figure 3, consist of the components listed below (a detailed description of the individual message components follows after Table 2). For an explanation of the various forms and effects of an in-frame response, refer to SAE J2178/1.

- ❏ Start of frame (SOF) period
- ❏ Data
- ❏ Cyclic redundancy check (CRC) byte
- ❏ End of data (EOD)
- ❏ Normalization bit when an in-frame response (IFR) protocol is used
- ❏ IFR bytes initiated by a responder immediately following the normalization bit when an IFR protocol is used.
- ❏ End of data (EOD), end of frame (EOF), and IDLE period

*Figure 3. Typical C2SI Data Frame Format Without In-Frame Response*

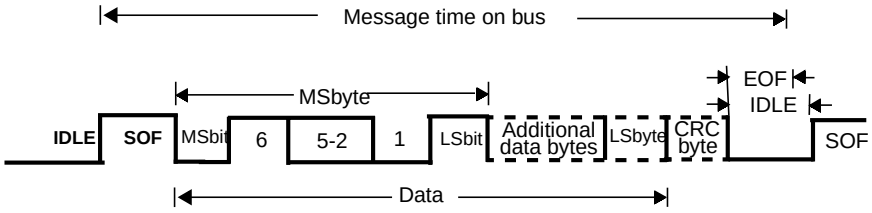



*Figure 4. Typical C2SI Data Frame Format With In-Frame Response*

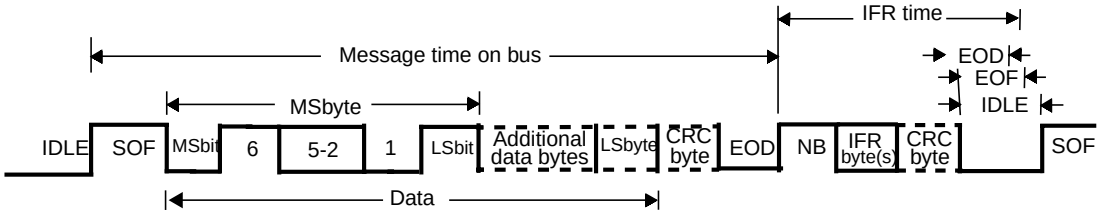


*Table 2. C2SI Message Time Duration*

| Message Components | | Level | Normal Mode | | 4x Mode | |
|---|---|---|---|---|---|---|
| | | | **TX (µs)** | **RX** (µs) | **TX** (µs) | **RX** (µs) |
| **SOF** | | | **192-208** | **163-239** | **48-52** | **41-60** |
| **Data Bits / CRC** | **0** | | **60-68** | **34-96** | **15-17** | **9-24** |
| | **1** | | **122-134** | **97-163** | **31-34** | **24-41** |
| | **0** | | **122-134** | **97-163** | **31-34** | **24-41** |
| | **1** | | **60-68** | **34-96** | **15-17** | **9-24** |
| **EOD** | | | **193-207** | **164-239** | **48-52** | **41-60** |
| **NB†** | | | **122-134** | **97-163** | **31-34** | **24-41** |
| | | | **60-68** | **34-96** | **15-17** | **9-24** |
| **EOF** | | | **271-289** | **240-320** | **68-72** | **60-80** |
| **Break** | | | **>280** | **>239** | **>280** | **>239** |

† There are two different conventions used for the Normalization bit. One type is an active long indicating that the in-frame response contains a CRC, and an active short indicating that it does not contain a CRC. The other type is vice versa: An active short indicating that the in-frame response contains a CRC, and an active long indicating it does not contain a CRC. The NBPOL bit (C2SIGCR.4) is used to determine which type of convention is to be used. See the NBPOL bit description in section 10.4 on page 38.

The C2SI transmits data bits via variable pulse width modulation (VPM) at either the normal mode or the 4x mode (this can be controlled by the 4XMODE bit (C2SIGCR.6)). The following are descriptions of each of the message components:

Start Of Frame (SOF)

The start of every message is initiated when the transmitter drives the bus high for approximately 200 µs (normal mode) or 50 µs (4x mode), which is referred to as the start of frame (SOF).

Data Bits and IFR Bits (when an IFR is used)

Once the SOF duration has been established, the data bits are transmitted on alternating high-low levels. Whether a data bit is a ‘0’ or ‘1’ is designated by the time between two consecutive transitions. A pulse duration of approximately 64 µs (normal mode) or 16 µs (4x mode) represents a data 0 bit if the pulse is low, or a data 1 bit if the pulse is high. Likewise, a pulse duration of

approximately 128 µs (normal mode) or 32 µs (4x mode) represents a 0 bit if the pulse is high, or a 1 bit if the pulse is low. Refer to Table 2, *C2SI Message Time Duration*.

### Cyclic Redundancy Check (CRC)

The CRC is optional. When this option is used, the C2SI will automatically generate a CRC and append it to the end of the data bytes in a message, and to the end of an in-frame response.

There is no CRC for type I and II in-frame responses. Only type III in-frame responses include the CRC. For more information on the types of in-frame responses and CRC value calculations, see the SAE J2178/1 specification.

The generation of CRC in transmitted messages (normal messages and in-frame response messages) and the expectation of CRC in received messages is controlled by the CRCDIS bit (C2SIGCR.7).

### End Of Data (EOD)

Once all data bits including CRC are sent, a falling edge occurs to generate a low level of approximately 200 µs (normal mode) or 50 µs (4x mode). This signifies the end of data (EOD).

An EOD will always appear after the last data byte in a message. Refer to Figure 2 and Figure 3.

### Normalization Bit

When there is an in-frame response from a responding device, the EOD duration ends when the responder sends its Normalization bit prior to the start of the first in-frame response byte (refer to Figure 3).

The normalization bit is always an active high level. The duration of the normalization bit is the same as a high level data 0 or data 1 bit time. When the normalization bit is 1, the in-frame response message ends with a CRC byte. When the normalization bit is 0, the in-frame response message does not end with a CRC byte. The NBPOL bit (C2SIGCR.4) will switch to meaning of 1 and 0 (1 = NO CRC) in order to conform to specific manufacturers conventions.

End Of Frame (EOF)

An end of frame (EOF) signifies the end of a message and appears at the end of all messages. If there is no in-frame response from a responding device, then the low level end of data (EOD) duration at the end of the data bytes will eventually stretch into an end of frame (EOF). Refer to Fig. 1-2. If there is an in-frame response, then the EOF appears after the last in-frame response byte (or CRC byte, if the CRC is used). Refer to Fig. 1-3.

The EOF is a falling edge that lasts approximately 280 µs (normal mode) or 70 µs (4x mode). Once EOF reaches 320 µs (normal mode) or 80 µs (4x mode), the device that transmitted the previous message may begin transmitting a new start of frame (SOF) since no other nodes are trying to access the C2SI bus.

Other C2SI's desiring bus access may try to arbitrate as early as between 280us and 320 us (normal mode). When all other devices that desire bus access detect this rising edge on the bus, they send their start of frame (SOF) almost immediately. If a device loses arbitration (the high voltage level is dominant in arbitration), it removes itself from the bus and its transmission is stopped.

Break

When a break signal is sent onto the bus, all nodes on the bus stop transmission immediately and go back into a reset condition. A break signal is initiated upon a rising edge and has a duration of at least 239 µs. The C2SI transmits a break signal of at least 300 µs.

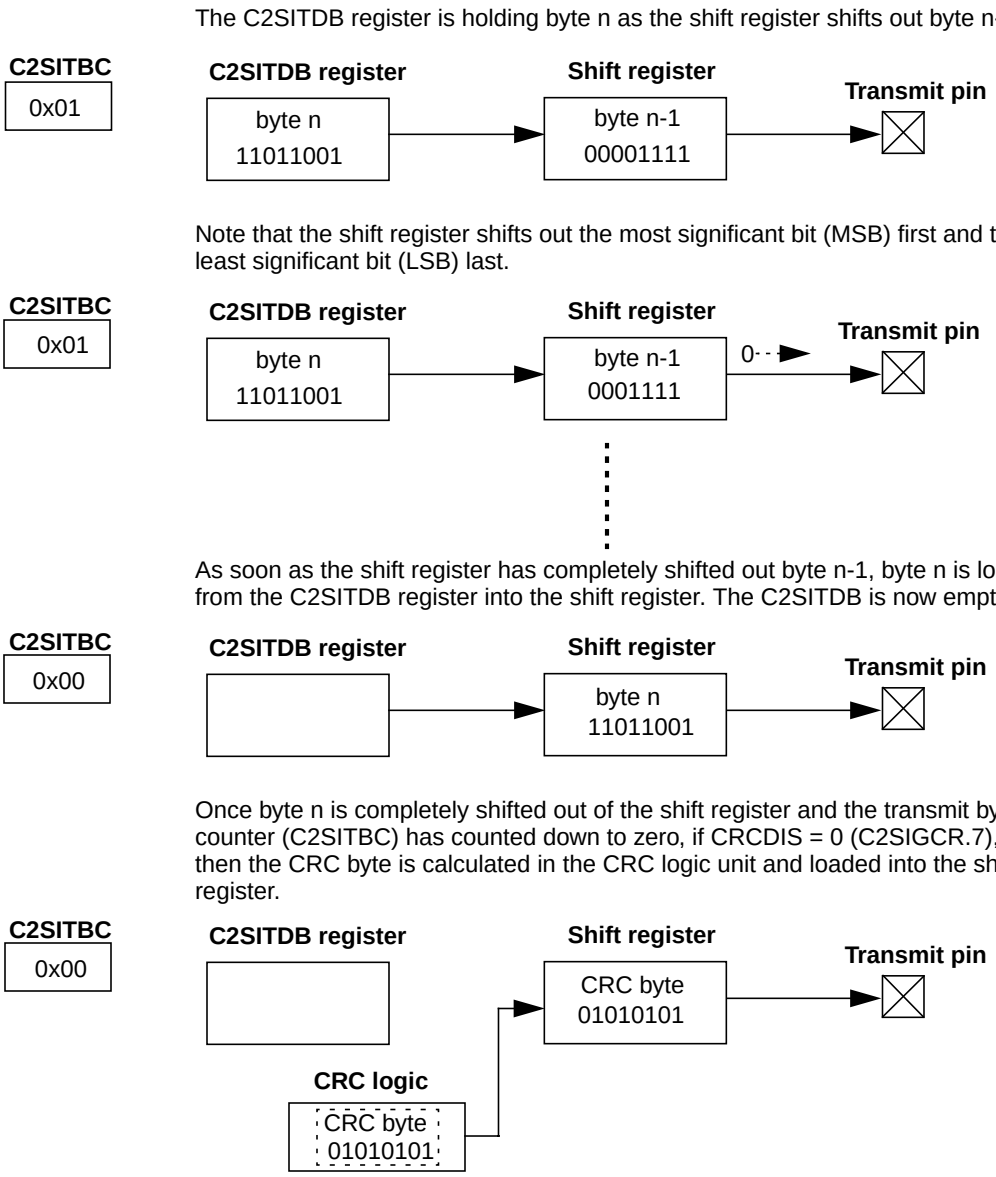
# 4 Transmitting C2SI Messages

Data for transmission on the C2SI bus must be transferred to the transmit data buffer; the data may be transferred by the CPU or by the DMA controller. For DMA based operations, see section 8.1, *DMA Transactions*, on page 24 and the DMA controller specification.

All messages can be transmitted with or without a CRC appended, and is controlled by the CRCDIS bit (C2SIGCR.7). See page 38 for the CRCDIS bit description. If the CRC is not used, the application may instead choose to calculate and include a checksum.

Internal to the C2SI's transmitter is a shift register that holds the contents of a byte as it is physically shifted out MS bit first on the C2SITXD pin. User software continues to supply data to the C2SITDB register (either from the CPU or via transmit DMA) until the entire message has been transferred. Figure 4 shows the transmission of the last 2 bytes of an n byte message plus an appended CRC byte.

*Figure 5. Byte Transmission*




During transmission, the C2SI's receiver and transmitter monitor each bit that the transmitter delivers to the C2SI bus to detect loss of arbitration. Since high voltage is the dominant level, arbitration is lost when the transmitter attempts to deliver a low voltage to the C2SI bus and the receiver detects a high voltage on the data link. This event forces the transmitter to immediately enter

the idle state. It also causes the arbitration lost bit (ARBIF), transmit idle bit (TIDLIF), and transmit buffer empty bit (TBEIF) (C2SIISR.7, 6, 5) to be set to 1 and the transmit DMA enable bit (TXDMAEN) (C2SICTR.3) to be cleared to 0.

A start bit (SOF) is neither a data ‘1’ nor a data ‘0’ and is not subject to arbitration. The C2SI's transmitter will place the start bit on an idle bus or begin following another module's start bit already in progress. If the receive pin detects the beginning of a start bit and the bus is forced low for some unknown reason during the start bit, then the C2SI will enter the idle state. It will get off the bus with transmit idle flag (TIDLIF) set as well as BITERR or CRCERR set. See the bus fault section 4.6, *Bus Error Conditions*, on page 16 for a discussion of what happens if the C2SI cannot begin to put its start bit on the bus.

The complete transmission of a message has occurred when the shift register empties, C2SITDB register is empty, and the Transmit Byte Counter register (C2SITBC) has counted down to 0x00.

**Note: Transmission with Lost Arbitration**

The C2SI module does not attempt to retransmit a message that lost arbitration during a transmission attempt except in the case of a type II in-frame response. See the T2IFR bit (C2SIGCR.5) description on page 1-39.

Detection of lost arbitration is correctly handled by monitoring the ARBIF bit (C2SIISR.7). If the ARBIE bit (C2SIICR.7) = 1, a transmit interrupt request can be generated on this event.

While the ARBIF bit is set, the transmitter does not attempt to communicate on the link. User software must guarantee that ARBIF is cleared before attempting to transmit on the bus.

## 4.1 Transmitting Non-IFR Messages

Non-IFR messages are sent when the TIFR bit (C2SIGCR.1) = 0. Before beginning a transmission, the following steps are recommended:

- C2SITBC register has been set to a non-zero value.
- C2SICCSR register has been read by the CPU.
- C2SIRDB register has been read by the CPU.
- Arbitration lost interrupt bit (ARBIF) is cleared (C2SIISR.7 = 0).
- C2SITDB register has been loaded by either the CPU or by a DMA transfer. This starts the transmission.

The C2SI monitors the C2SI bus until it has become IDLE. Transmission of an SOF sequence prefixes a normal message's packet of data bytes. Following the SOF, the shift register is loaded from the C2SITDB register and the transmission of data bytes continues until the C2SITDB register and the shift register are empty and C2SITBC is 0. A CRC is appended to the end of the packet of data bytes if the CRCDIS bit is 0 (C2SIGCR.7 = 0).

When you start a message, the first byte put into the C2SITDB will quickly move to the empty shift register about a microsecond later and the TBEIF bit will indicate that the C2SITDB is ready for the second byte. Always wait for the TBEIF bit to be set before loading into the C2SITDB.

There are two ways to handle the TBEIF bit when sending a multibyte message. You can clear the TBEIF bit directly or you can write to the C2SITDB to clear the flag. If you use the first method, you will also get a TBEIF interrupt when each completion code bit is filled and this usually means a good message will generate a TBEIF interrupt when setting the XMITOK bit after the message is complete.

If you use the second method and clear the TBEIF bit only by writing to the C2SITDB register, you may want to disable the TX empty interrupt by clearing the TBEIE bit after you write the last byte of your message. This saves you from servicing the TBEIF interrupt when the last byte transmits and when the completion code is filled.

## 4.2 Transmitting In-Frame Response (IFR) Messages

In-frame response messages are sent when TIFR bit (C2SIGCR.1) is 1. Before beginning a transmission, the following steps are recommended:

- C2SITBC register has been set to a non-zero value.
- C2SICCSR register has been read by the CPU.
- Arbitration lost interrupt bit (ARBIF) is cleared (C2SIISR.7 = 0).
- C2SITDB register has been loaded by either the CPU or by a DMA transfer. This starts the transmission.

The responder of the message begins transmitting an in-frame response after the occurrence of an EOD. An NB bit, which reflects whether or not a CRC follows the in-frame response, prefixes the in-frame response packet of data bytes. Following the NB bit, the shift register is loaded by the C2SITDB register and the transmission of data bytes continues until the C2SITDB register and the shift register are empty. A CRC is appended to the end of the packet of the in-frame response bytes if CRCDIS bit is 0 (C2SIGCR.7 = 0).

The TIFR bit (C2SIGCR.1) controls whether or not the C2SI is transmitting an in-frame response or a normal start-of-frame message. The TIFR bit

(C2SIGCR.1) is automatically cleared if the receiver detects any errors during reception. This guarantees that an in-frame response is not transmitted in response to a corrupt message. When a receiver detects any errors during reception, the following conditions will occur:

- TIFR bit (C2SIGCR.1) is cleared
- The transmitter is reset
- TXDMAEN bit (C2SICTR.3) is cleared
- TBEIF (C2SIISR.5) and TIDLIF (C2SIISR.6) are set

The application is responsible for recognizing that the message currently being received is expecting an in-frame response to be transmitted as a response. This may require the application to monitor each incoming byte of a message in order to detect this condition. The application must insure that the in-frame response is ready for transmission prior to the completion of the received packet's EOD sequence.

## 4.3 Transmitting BREAK Messages

The transmission of a BREAK sequence is forced by setting the TBRK bit (C2SIGCR.2) is 1. This results in the current transmit/receive condition of the link being overridden by the BREAK sequence. The duration of the BREAK sequence is independent of the state of the 4XMODE bit (C2SIGCR.6) and is always at least 300 μs long.

Transmitting/receiving a BREAK disables all transmissions/transmitters on the link. It also automatically clears the 4XMODE bit. When the BREAK occurs, all transmitters transmitting a message other than BREAK will be forced off the bus and the C2SI will generate a condition code with at least the break bit set.

You must wait for the break to finish before writing to the transmit data buffer (C2SITDB). See section 5.3, *Receiving BREAK Messages*, on page 19 for an explanation on how the receiving devices react to a BREAK.

## 4.4 Transmission Arbitration

When a message is transmitted by the C2SI module, a copy of the bit stream is redirected back to the received section so it can monitor its arbitration progress. If the C2SI's transmitter applies a data 1 to the data link and the C2SI's receiver returns a data 0, then the transmitter has lost arbitration.

When arbitration is lost, the transmitter removes itself from the data link, and the following events occur:

- The XMITOK bit (C2SICCSR.3) in the Completion Code register is cleared.
- The ARBIF bit (C2SIISR.7) is set.
- The transmitter idles itself by clearing the TXDMAEN bit (C2SICTR.3).
- The internal shift register and the C2SITDB register are marked empty, therefore, TIDLIF and TBEIF bits (C2SIISR.6-5) are set.

**Transmission with Lost Arbitration**

The C2SI module does not attempt to retransmit a message that lost arbitration during a transmission attempt (except during a type II in-frame response). Detection of loss arbitration is correctly handled by monitoring the ARBIF bit (C2SIISR.7). An interrupt request can be generated on this event if ARBIF = 1 and ARBIE bit = 1 (C2SIICR.7).

While the ARBIF bit is set, the transmitter does not attempt to communicate on the link. User software must guarantee that ARBIF is 0 (C2SIISR.7) before attempting to retransmit on the link.

If arbitration is not lost, the transmitter naturally idles itself at the end of a message. The transmitter is idle when the TXDMAEN bit (C2SICTR.3) is cleared and when TIDLIF and TBEIF bits (C2SIISR.6-5) go high. In addition, the C2SI's receiver and transmitter collectively determine whether the XMITOK bit (C2SICCSR.3) should be set. It is set if the transmitter wins arbitration and the receiver has not detected any errors during the reception of the transmitted message.

If the module is ready to transmit a message, just waiting for the bus to go idle and if another device jumps onto the bus after the minimum end of frame (EOF), then the module does **not** send out a start of frame pulse. Instead, it uses the other device's start of frame (SOF) as a synching pulse so that each transmitter will start arbitrating on the bus at the same time.

## 4.5 Byte Boundary Loss of Arbitration when Transmitting

If arbitration is lost on the last bit of a byte being transmitted, the transmitter does not immediately remove itself from the data link. Instead it transmits two additional 1s. If arbitration is lost again on the first 1, the transmitter immediately stops transmitting.

If this loss of arbitration had been due to noise, the extra two 1s are intended to corrupt a potentially acceptable, but erroneously short message generated by the transmitter. If loss of arbitration was due to a higher priority message, then the 1s have no affect on that message.

Also, if a transmit underrun condition occurs within the C2SI module, then arbitration will be lost, and two extra 1s are transmitted in order to corrupt the message for the other receivers.

## 4.6 Bus Error Conditions

Conditions can occur in a system that will temporarily or permanently cause the C2I bus to fail. Three of the more common failure modes are:

- ❏ Short-to-voltage
- ❏ Short-to-ground
- ❏ Bus open - broken wire

The commercially available bus interface devices will handle the physical strain of short to ground and short to power without damaging either device. The interface device however will not directly inform the C2I module of the fault condition. The C2I module must determine the fault through indirect means.

### *4.6.1 Short-to-Voltage*

The following conditions will make the bus appear to be shorted to a voltage source.

- ❏ Bus shorted to voltage source
- ❏ TX pin stuck high
- ❏ RX pin stuck high
- ❏ TX signal between C2I module and interface device shorted to voltage
- ❏ RX signal between C2I module and interface device shorted to voltage

Each of these conditions will cause a BREAK condition on the C2I and set the break flag bit, BRKIF. If the break interrupt enable bit, BRKIE, is also set and C2I interrupts enabled then the BREAK will cause an interrupt. See section 1.6.3 and 1.10.1-bit1 for break handling. The problem can be isolated to a bus problem if the C2I enables the loop back mode of the device. The C2I will still be able to send and receive in loop back mode if the bus shorts to voltage.

### *4.6.2 Short-to-Ground*

The follow conditions will make the bus appear to be shorted to ground.

1) Bus shorted to ground
2) TX pin stuck low (may still receive from other devices)
3) RX pin stuck low
4) TX signal between C2I module and interface device shorted to ground (may still receive from other devices).

5) RX signal between C2I module and interface device shorted to ground

If the C2I transmitter attempts to send under these conditions the TXD pin will be set high and the receive pin will wait indefinitely for the feedback of the start bit. This condition may be ascertained by looking at the IDLE bit after writing to the transmit buffer. The IDLE bit will go to zero 10 us after receiving the beginning of the start bit. Testing the IDLE bit after accounting for the bus delay between TXD high and RXD high and this RXD to IDLE clear time will show if the short to ground symptoms are present.

The problem can be isolated to a bus problem if the C2I enables the loop back mode of the device. The C2I will still be able to send and receive in loop back mode if the bus shorts to voltage. If condition 3 or 5 above occurs then the C2I must be taken off the bus by turning off the transmitter. These conditions cause a BREAK signal to the rest of the devices on the bus.

The C2I receiver can determine a short to ground only by not receiving a message in a timely manner. This is dependent on the software message protocol and should be encompassed in the protocol definitions.

### *4.6.3 Open Bus*

The follow conditions will make the bus appear to be shorted to ground.

- ❏ Bus open or
- ❏ Bus wire broken

In this instance, the transmitter appears to be working if no in frame responses are expected. The data sent out the TXD pin will return through the RXD pin in the normal manner. Lack of an expected IFR will point to this condition as a possible problem. (Problems with the IFR generator is another cause)

The C2I will receive no data from other devices during this time so bus protocols should be set up to identify this type of problem.

### *4.6.4 Bus Error Detection Through Time Outs*

One method to detect many of these bus errors is to set up a software counter. The program would begin the counter when transmitting a byte and disabled the counter after receiving the TX condition code. The software would increment the counter via the real time timer or a main polling loop. If the counter passed a certain time an error would be declared and handled. The type of bus problem could be determined by looking at the TXDIN, RXDIN, IDLE, BRKIF bits and by using the loop back mode.

### 4.6.5 Other Bus Errors Not Discussed

Some other the bus errors not discussed:

- ❏ The intermittent error (now you see it, now you don't).
- ❏ Open signals between C2I and interface device. May look like either short to ground, short to power or intermittent.
- ❏ Bus with too much load or capacitance -- slow bus.
- ❏ Combination of several of the above faults.

# 5 Receiving C2SI Messages

## 5.1 Receiving Normal Messages

The reception of a normal message is preceded by an SOF sequence. Data bytes are received through the receiver's shift register, and then transferred into the C2SIRDB register. Reception of a normal message continues until a non-data bit sequence is received (for example, EOD, EOF, BREAK, noise). When this occurs a completion code for the message is updated in the C2SICCSR register.

The state of the CRCDIS bit (C2SIGCR.7) controls the formation of a CRC for transmitted data, and whether the receiver should expect a CRC at the end of the incoming data. If a CRC is expected and there is a CRC error, the CRCERR bit (C2SICCSR.4) in the completion code is set.

## 5.2 Receiving In-Frame Response (IFR) Messages

The reception of an in-frame response (IFR) follows after the C2SI successfully transmitted a normal message which ended with an end of data (EOD). What sets it apart from the reception of a normal message is the preceding normalization bit (NB) rather than the start of frame (SOF) sequence.

Reception of an in-frame response continues until a *non-data bit* sequence is received (for example, EOF, BREAK, noise). When this occurs, a completion code for the message is updated in the C2SICCSR register.

The reception of an in-frame response sets the IFR bit (C2SICCSR.1) in the completion code register. The state of the NB bit is reflected in the completion code's IFRCRC bit (C2SICCSR.0). The IFRCRC bit is set when an in-frame response with a CRC byte appended to the end of it has been received. The CRCERR bit (C2SICCSR.4) in the completion code will be set if there are any errors in this appended CRC byte.

Note that the receiver can receive an IFR without a CRC even if CRCDIS bit = 0 (C2SIGCR.7).

## 5.3 Receiving BREAK Messages

A BREAK on the data link causes any messages on the link to be aborted. The BREAK's corruption of a message in process is detected when a symbol (for example, data bit SOF) has been overridden. The receiver's reaction to a BREAK depends on conditions at the time of the BREAK.

If a message was in process, a symbol could get corrupted and the receiver posts a completion code with errors (BREAK and possibly BTYERR, BITERR or CRCERR). This sets the RCCIF bit (C2SIISR.2) and, if RCCIE bit is 1, a receive interrupt request is generated. Receiving the break symbol also resets internal transmitter state machine and TBEIF is set.

If a message was in process when a BREAK is asserted on the data link, the receiver sets the BRKIF bit (C2SIISR.1) along with the RCCIF bit and clears the 4XMODE bit (C2SIGCR.6). The BRKIF bit is repeatedly set for the first 1us following the BREAK detection. After 1us, the BRKIF bit can be cleared by writing a 0 to the bit. The BREAK interrupt will be generated as long as the BRKIF bit = 1 and the BRKIE bit = 1. See section 6.2, *Break Interrupts*, on page 22.

When the BREAK ends, a completion code with the BREAK bit equal to 1 (C2SICCSR.2) is validated in the completion code register (C2SICCSR). Therefore, a BREAK typically may cause up to three interrupt generating events in the following order:

1) An RCCIF due to an aborted message.
2) A BRKIF condition occurs.
3) Another RCCIF due to the release of BREAK.

A typical way of handling the BREAK is:

1) Disable the BRKIE bit (C2SIICR.1) and clear the BRKIF bit (C2SIISR.1) in the BRKIF interrupt service routine when the BREAK is first detected.
2) Reenable the BRKIE bit (C2SIICR.1) in the RCCIF interrupt service routine when the BREAK has ended.
3) Set a timer outside the C2SI module so you can detect short-to-voltage conditions.

## 5.4 Receiving Digital Filters

A digital filter internal to the C2SI is used to filter noise pulses from the class II receiver driver module that are smaller/shorter than the analog interface chip's filter time constant.

# 6 Interrupts

The C2SI has two main interrupt-related registers--one is an interrupt status register (C2SIISR), and the other is an interrupt control register (C2SIICR). The C2SI module generates one interrupt request back to the CIM. When an interrupt generating event (IGE) occurs, its corresponding interrupt status register (C2SIISR) bit is set (for example, ARBIF = 1). If the corresponding interrupt control register (C2SIICR) bit is enabled (for example, ARBIE = 1), then an interrupt request is sent to the CIM.

If multiple interrupt conditions occur either at the same time or during the time the interrupt service routine is being executed, multiple interrupt status flags will be set but only one interrupt will be generated back to the CIM. User software must read the C2SIISR register to determine which event caused the interrupt and to determine which C2SI interrupt condition has priority; the C2SI has no hardware means of determining priority.

If the interrupt is caused by a wake from sleep mode, however, the WAKE bit (C2SIGSR.0) is set, and will generate an interrupt. The WAKE bit is listed in the C2SIGSR and not the C2SIISR register. As a result, the application software will need to poll both registers to find all occurrences of an interrupt condition.

## 6.1 Proper Handling of IGE

Some interrupt generating events (IGE), such as, transmit buffer empty interrupt, received completion code interrupt, and received buffer full interrupt are persistent and can occur continuously and reassert set their respective interrupt flags unless their cause is handled. An application must examine all interrupt status flag bits in the C2SIISR that are associated with an interrupt request. Since multiple IGEs can cause a request, each interrupt status flag bit should be cleared as it is serviced in order to maintain the ability to examine them and distinguish between those that have been serviced and those that have not.

Interrupts are cleared via two conditions:

- Performing the action requested by the interrupt. For example, if a receive buffer full interrupt is generated (RBFIF is active), then the process of reading the receive buffer will clear the interrupt
- An explicit write of 0 (zero) to the active interrupt status flag bit will clear the interrupt, even if the required service is not taken.

## 6.2 Break Interrupts

This interrupt condition continuously asserts itself for the first 1us. But after the first 1us, the BREAK interrupt can be disabled even if the BREAK condition still exists. Therefore, the typical way of handling the BREAK interrupt is to disable the BREAK interrupt (clear BRKIE) and leave it clear until the completion code for the BREAK is updated in the C2SICCSR register. The interrupt service routine for the RCCIF interrupt can be used to reenable the BREAK interrupt (setting BRKIE = 1).

## 6.3 SHORT Induced Arbitration Loss Interrupts

An interrupt due to bits ARBIF (C2SIISR.7) and ARBIE (C2SIICR.7) when the bus is shorted is unique because it identifies a condition of the link that can last an indefinite period of time. A typical user software response to an arbitration loss event is to reinitialize for a retransmission attempt. Since a shorted bus looks like an idle bus, the transmitter is able to start its attempt with minimum delay. If the short is still present when the transmitter tries to send another message, another arbitration loss occurs and the interrupt is generated. This results in a arbitration loss interrupt every (~280 µsec + transmit_handler + latency).

If the burden of repetitive arbitration loss interrupts due to a short is excessive, user software must either introduce a delay mechanism before a retransmisson attempt or rely on a completion code interrupt to signal that the link has been restored and completely reinitialize transmit interrupts from within a receive interrupt handler.

## 6.4 Completion Code Interrupts

An interrupt due to bits RCCIF (C2SIISR.2) and RCCIE (C2SIICR.2) is unique because it identifies a condition of the link that can last an indefinite period of time.

The setting of the RCCIF bit is a function of the C2SICCSR register being *full* and must not be confused with the C2SICCSR register being *filled*. The C2SICCSR register is filled whenever the completion code is updated and remains full. Therefore, if user software fails to read the completion code once it is updated, or filled, the C2SICCSR register remains full. This results in the RCCIF bit being continuously set = 1. If the RCCIE bit is also set = 1, a continuous series of receive interrupt requests is generated.

# 7 General Purpose I/O

Each of the C2SI pins may be programmed via the C2SI pin control registers (C2SIPC1, C2SIPC2 and C2SIPC3) to be a general-purpose I/O pin.

When the C2SI module is not used, the C2SI pins may be programmed to be either general input or general output pins. This function is controlled in the C2SIPC1 register. Note that each pin can be programmed to be either a C2SI pin or a GPIO pin.

If the C2SI function is to be used, application software must ensure that each pin is configured as a C2SI pin and not a GPIO pin, or else unexpected behavior may result.

# 8 DMA Interface

## 8.1 DMA Transactions

If handling the C2SI message traffic on a byte-by-byte basis requires too much CPU overhead and if the particular device is equipped with the DMA controller, the C2SI may use the DMA controller to receive or transmit data directly to memory. The C2SI module contains two DMA request enable bits: a transmit DMA enable (TXDMAEN) and a receive DMA enable (RXDMAEN), both of which are located in the C2SICTR register.

When a byte is being transmitted or received, the C2SI will signal the DMA via a DMA request signal. The DMA controller will then perform the needed data manipulation.

For DMA-based transmissions, all messages (other than a BREAK) are assembled in RAM, and DMA transfers move the message, byte-by-byte, from RAM into the C2SITDB register. (See the DMA controller specification). All messages transferred via DMA contain only data received via the receive buffer; the contents of the completion code status register (C2SICCSR) are not transferred to RAM. User software must read the completion code status register (C2SICCSR). See section 10.6, *C2SI Completion Code Status Register (C2SICCSR)*, on page 41.

The application is responsible for programming the C2SITBC to the desired number of bytes to be transferred. If the C2SITBC contains a different value than the DMA byte count register, a transmit under-run or overrun condition can occur. For specific DMA features, refer to the DMA controller specification.

## 8.2 Non-DMA Transactions

The C2SI module has been optimized for the use of its DMA facilities during communications on the class II data link. It is possible to perform communication transactions on the data link manually by using the CPU to transfer bytes to/from the C2SI module.

If the application must perform data transfers via the CPU, the appropriate flags, transmit buffer empty interrupt flag (TBEIF) and receive buffer full interrupt flag (RBFIF), are provided. They are both located in the C2SIISR register. When used with their respective interrupt enable bits, TBEIE and RBFIE (C2SIICR.5,0), these flags inform the CPU when it is time to write data to the C2SITDB register and read data from the C2SIRDB register. The application is responsible for guaranteeing that the C2SITDB register is kept full and that the C2SIRDB register is read in accordance with class II data communication rates. Failure to do so results in truncated transmissions or

overrun during reception. The application is also responsible for programming the C2SITBC to the desired number of bytes to be transferred.

# 9 C2SI Modes

## 9.1 4X Mode

The C2SI module has the ability to function in the normal mode or 4X clock mode. In the 4X clock mode, all timing constants for the generation/reception of signals on the data link are effectively divided by four.

Changing the state of the 4XMODE bit (C2SIGCR.6) resets the class II state-machine status, thereby aborting any transmissions or reception in progress. The receipt of a BREAK sequence automatically clears the 4XMODE bit and resets the class II state machine.

If the state of the 4XMODE bit is changed during the reception of a message, the C2SICCSR register is updated to mark the end of the aborted message.

The C2SI must calibrate before transmitting in 4X mode. See section 9.4, *Calibration Mode*, on page 28.

## 9.2 Low Power Mode

The C2SI module has two means to be placed in a low-power mode: a global low-power mode from the system and a local low-power mode via the LPM bit (C2SICTR.0). The net effect on the C2SI is the same, independent of the source.

A low-power mode in effect shuts down all the clocks to the module. During a global low-power mode, no registers are visible to the software; nothing can be written to or read from any register. A local low-power mode has the same effect, with the exception that the LPM bit may be written to, and hence able to place the module into a functional mode.

Since entering a low-power mode has the effect of suspending all state-machine activities, care must be taken when entering such modes to insure that a valid state is entered when low-power mode is active. For example, if a low power mode is entered during a transmission on the Class II bus before the message is complete, a completion code will never be sent out, and hence the integrity of the bus is corrupted. As a result, application software must insure that a low power mode is not entered during a transmission.

Low-power mode may be used in conjunction with the ENWAKE bit (C2SIGCR.0) to allow C2SI bus activity to wake the device and exit the low-power mode.

To enter low-power mode and wake up on any C2si bus activity:

❏ Wait for idle bus by checking the IDLE bit (C2SIGSR.3)

- ❏ Clear the ENWAKE bit (C2SIGCR.)
- ❏ Set the LPM bit (C2SICTR.0). This is now local low-power mode.

To enter global low-power mode, set the global LPM bits (CLKCNTL1:0). The global C2SI interrupt does not have to be enabled to wake up the device.

To enter low-power mode and ignore all C2SI bus activity:

- ❏ Wait for idle bus by checking the IDLE bit (C2SIGSR.3)
- ❏ Clear the ENWAKE bit (C2SIGCR.0). Clear C2SIICR.
- ❏ Set the LPM bit (C2SICTR.0). This is now local low-power mode.
- ❏ Unlike some other modules, you do not have to be in a privilege mode to set the LPM bit.

To enter global low-power mode, set the global LPM bits (CLKCNTL1:0).

To exit the low-power mode, you must first clear the LPM bit. No other C2SI bit is writable until the LPM bit is cleared. The internal C2I counters and state machine will not start until the LPM bit is cleared. This means the time from the LPM cleared to the end of the OSF starting bit must meet the minimum SOF time in order to receive a good message. In many instances the wake-up message will be lost because this time could not be met.

The global peripheral power-down override bit (PPWNOVR< CLKCNTL.7) has no effect on the C2I.

## 9.3 Emulation Mode

The C2SI module may be placed in a suspend mode by the TMS470 system. This is usually when the TMS470 is being used as an emulator or being debugged via the test access port (TAP). When being used by a monitor program, the receive data buffer (C2SIRDB) has a mirror register called C2SIEMU. This register contains the same contents as the C2SIRDB, but a read of this register will not cause the receive buffer full interrupt flag (RBFIF) to clear. This allows the user to keep a memory window open for the receive buffer, without having the monitor program clear the interrupt automatically.

The software has the choice via the ESPEN bit (C2SICTR.1) as to the state-machine action taken during a suspend mode. If the ESPEN bit (C2SICTR.1) is active, the C2SI will immediately suspend its activity. Once again, the user's software must ensure that suspending the transmission or reception of data will not corrupt the class II bus. If ESPEN is inactive, the C2SI will continue operating normally.

## 9.4 Calibration Mode

Calibration allows the C2SI module to know the expected time delay between sending a bit out of the TXD pin and receiving the same bit back into the RXD pin. Due to the variations among different analog designs, the C2SI module needs to be calibrated to the actual bus load. A counter in the C2SI will measure the delay and adjust the transmit timings automatically according to the results of the calibration cycle.

Since another device trying to send simultaneously with a calibration cycle would interfere with the calibration, it is best if no other modules are transmitting on the bus during a calibration cycle.

If you must calibrate on a busy bus, the it is recommended that you first calibrate using the loop-back mode followed by a calibration message on the loaded bus. It is also possible to calibrate using ONLY the loop-back mode on some external transceivers, but this will need to be determined on a system-by-system basis.

If you calibrate on a busy bus, then you may lose arbitration to another device. If this happens you must repeat the loop-back cycle before attempting another calibration cycle on the bus. Keep trying until the C2SI returns a SMITOK status after a bus calibration.

Calibrate the loop-back sequence by performing the following:

1) Prepare for a standard transmission.
    a) Read the completion code register.
    b) Clear the necessary bits in the C2SIISR such as; ARBIF, BRKIF, etc.
    c) Read the receive buffer if necessary.
    d) Set the transmit byte counter, C2SITBC to 1.
2) Set the calibration and loop-back enable bits, CALEN, LPEN, (C2SICTR.6:5).
3) Turn off the CRC by setting the CRCDIS. If you are calibrating for 4X mode, then set the 4XMODE bit (C2SIGCR.6).
4) Write any value to the TX data buffer (C2SITBD). The value 0x55 will finish in the shortest time.
5) Wait for transmit idle flag (TIDLIF) to go high to indicate the end of the loop-back transmission.

Calibrate the bus by performing the following:

1) Keep the calibration set, but turn off the loop-back enable bits (CALEN = 1 and LPEN = 0).
2) Transmit a normal message. The C2SI will send this byte out on the bus and set the calibration constant using the time it takes it to return on the RXD pin. The final calibration constant will get set using the last bit of the message.
3) You have successfully calibrated if you finish normally with the XMITOK bit set. Then turn off the CALEN bit.
4) If you receive arbitration of another error, then repeat the procedure starting from step 1 of the loop-back procedure.

Although the C2SI uses a default calibration constant upon reset, it is recommended that the device perform a calibration cycle to get the full bus performance.

You MUST do a calibration cycle before you can transmit using the 4X mode of operation since most bus interface devices have different delays for the normal and 4X modes.

The calibration constants do not affect the C2SIs reception of data from other devices.

If other transmitters must be on the bus, then keep performing a calibration cycle until the completion code returns a transmit OK with no arbitration errors. If possible, transmit a zero since this will win all arbitrations. If you can’t use zero, then try to use the lowest value with zeros in all the even bit positions.

The calibration constants do not affect the C2SI’s reception of data from other devices.

# 10 C2SI Internal Registers

*Table 3. C2SI Control Register File Used With the TMS470 CPU*

| Address Offset† | Mnemonic | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Register Name |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | C2SIISR | ARBIF | TIDLIF | TBEIF | TXUOIF | RXOIF | RCCIF | BRKIF | RBFIF | Interrupt status register |
| 0x04 | C2SIICR | ARBIE | TIDLIE | TBEIE | TXUOIE | RXOIE | RCCIE | BRKIE | RBFIE | Interrupt control register |
| 0x08 | C2SIGSR | Reserved | | | | IDLE | Rsrvd | NOISE | WAKE | Global status register |
| 0x0C | C2SIGCR | CRCDIS | 4XMODE | T2IFR | NBPOL | Rsrvd | TBRK | TIFR | ENWAKE | Global control register |
| 0x10 | C2SITDB | TDDATA.7:0 | | | | | | | | Transmit data buffer register |
| 0x14 | C2SICCSR | ROVR | BITERR | BYTE RR | CRCERR | XMITOK | BREAK | IFR | IFRCRC | Completion code status register |
| 0x18 | C2SICTR | Rsrvd | CALEN | LPEN | Rsrvd | TXDMA EN | RXDMA EN | ESPEN | LPM | Peripheral control register |
| 0x1C | C2SICLK | Reserved | | | ICLKFR | | | | | Interface clock register |
| 0x20 | C2SITBC | TBCOUNT.7:0 | | | | | | | | Transmit byte counter |
| 0x24 | C2SIPC1 | TXPOL | RXPOL | TXFUN | RXFUN | LPFUN | TXDIR | RXDIR | LPDIR | Pin control register 1 |
| 0x28 | C2SIPC2 | Reserved | | | | | TXDOUT | RXDOUT | LPDOUT | Pin control register 2 |
| 0x2C | C2SIPC3 | Reserved | | | | | TXDIN | RXDIN | LPDIN | Pin control register 3 |
| 0x30 | C2SIEMU | REDATA.7:0 | | | | | | | | Receive emulation register |
| 0x34 | C2SIRDB | RDDATA.7:0 | | | | | | | | Receive data buffer register |

† The actual address of these registers is device specific and CPU specific. See the specific device data sheet to verify the C2SI register addresses.

## 10.1 C2SI Interrupt Status Register (C2SIISR)

The C2SIISR register consists of interrupt flags. Each interrupt flag represents the occurrence of a different interrupt generating event; the individual flags are set when their respective interrupt generating event occurs. Interrupt flags in the C2SIISR register are purely status flags and do not initiate interrupt requests alone. Only when used in conjunction with the interrupt enable bits in the interrupt control register (C2SIICR) will any

requests be made to handle the interrupt. A write of '0' to a bit in this register will clear the bit.

| hex | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x00 | ARBIF | TIDLIF | TBEIF | TXUOIF | RXOIF | RCCIF | BRKIF | RBFIF |
| | RC-0 | RC-1 | RC-1 | RC-0 | RC-0 | RC-0 | RC-0 | RC-0 |

RC = Read/Clear, *–n* = Value after reset

**Bit 7** **ARBIF** Transmit Arbitration Lost Interrupt Flag.

This bit is set when the C2SI determines that it has lost its transmit arbitration attempt for the data link. This can occur when a message of higher priority is being transmitted by another device. Also, a BREAK asserted on the data link could cause arbitration to be lost in certain cases.

When arbitration is lost and ARBIE = 1 (C2SIICR.7), the C2SI module generates a transmit interrupt request.

The ARBIF bit and the type II in-frame response control bit, T2IFR (C2SIGCR.5), need to be monitored together in order to determine whether a transmission will be resent or not when a transmit arbitration lost interrupt occurs. See also T2IFR, page 1-39. The following two conditions can happen when the C2SI loses arbitration while transmitting:

- If ARBIF = 1 and the T2IFR = 0 (meaning a type II in-frame response is not in progress):
  - The TXDMAEN bit (C2SICTR.3) is automatically cleared
  - The TIDLIF and TBEIF bits (C2SIISR.6,5) are set.

  The transmitter is disabled from continuing its transmission and does not attempt to get on the data link. ARBIF bit must be cleared before any transmission can take place.
- If ARBIF = 1 and T2IFR = 1, the transmitter remains active and automatically resends the contents of the shift register.

For more details on the type II in-frame response see the SAE J1850 specification.

**Note: Enabling ARBIE**

ARBIE must be enabled for any transmission except for type II in-frame responses. Otherwise, it is possible to lose arbitration in the first byte of a message and the receiver will not recognize the loss in a completion code since the completion code starts updating only after the receipt of the first good byte. Unless the setting of ARBIF identifies the loss, the transmit software may assume that the transmission has succeeded.

If transmit retry is required for the next frame, the latency requirement for interrupt response is:

- ❏ Normal operation:~790 µsec (a CRC byte shift time and an EOF in normal mode)
- ❏ 4X mode:~70 µsec (an EOF in 4X mode)

0 = Arbitration not lost.
1 = Transmitter lost during data link arbitration attempt.

**Bit 6** **TIDLIF** Transmit Idle Interrupt Flag.

Active high indicates that the transmitter is idle and available for reloading. This occurs whenever the following events occurs:

- ❏ The transmitter loses arbitration.
- ❏ Break
- ❏ The transmit shift register became empty after emptying the C2SITDB register while TXDMAEN = 0 (C2SICTR.3 = 0).

When this occurs and TIDLIE = 1, C2SI generates a transmit interrupt request.

TIDLIF remains set as long as both the C2SITDB register and the transmit shift register are empty. It can be cleared by a CPU write to the C2SITDB register or a DMA transfer from system RAM to the C2SITDB register.

0 = Transmitter is in use.
1 = Transmitter is idle.

**Bit 5** **TBEIF** Transmit Buffer Empty Interrupt Flag.

This bit is set whenever the C2SITDB register is emptied by the transfer of its contents into the internal shift register. TBEIF is set when the C2SITDB register is empty or when bits are posted to the completion code register (C2SICCSR). It can be cleared by a CPU write to the C2SITDB register or a DMA transfer from system RAM to the C2SITDB register

When TBEIF is set and TBEIE = 1, C2SI generates a transmit interrupt request. If the C2SITDB register is empty and TBEIE = 1, continuous interrupt requests are possible.

0 = Transmit data buffer contains a character.
1 = Transmit data buffer register is empty.

**Bit 4** **TXUOIF** Transmit Under-run, Over-run Interrupt Flag.

A transmit under-run condition occurs when the transmitter has finished transmitting a byte and is ready to receive the next byte, but the DMA or CPU has not provided the next byte to be transmitted. However, if C2SITBC = 0, then the transmit under-run flag is not set, as that indicates the end of transmission condition.

A transmit over-run condition can occur if the CPU or DMA sends a byte to be transmitted when C2SITBC = 0.

An active TXUOIF indicates a severely disturbed system, either by hardware or software. This condition results in the C2SI sending out an extra data bit, causing an error in the frame. This will cause all systems receiving the frame to receive an erroneous message.

0 = Transmit under-run, over-run has not occurred.
1 = Transmit under-run, over-run has occurred.

**Bit 3** **RXOIF** Receive Over-run Interrupt Flag.

A receive over-run condition occurs when the receiver has received the next byte, transferred it to the receive buffer (C2SIRDB), but the previous buffer contents have not yet been read, either by the CPU or the DMA. The previous byte is overwritten (and hence lost), and RXOIF is set.

0 = Receive over-run has not occurred.
1 = Receive over-run has occurred.

**Bit 2** **RCCIF** Receiver Completion Code Interrupt Flag.

The RCCIF bit is set after an end of frame (EOF) time at the end of a normal and in-frame response message, and at the end of a BREAK. This bit is continuously set until the C2SICCSR register is read by the CPU. Reading the C2SICCSR will clear this bit.

When this bit is set and RCCIE bit (C2SIISR.2) = 1, C2SI generates a receive interrupt request. If the C2SICCSR register has not been read and RCCIE = 1, continuous receive interrupt requests are possible.

The latency requirement for interrupt response:

- ❏ Normal operation:~500 μsec (one byte shift times in normal mode)
- ❏ 4X mode:~125 μsec (one byte shift time in 4X mode)

0 = All bits in C2SICCSR register hold an intermediate completion code.
1 = Completion code is done updating in C2SICCSR register.

**Bit 1** **BRKIF** Received Break Interrupt Flag.

This bit is set = 1 as soon as the receiver detects a BREAK sequence on the data link. When this occurs and BRKIE = 1, the C2SI generates a receive interrupt request.

BRKIF bit is repeatedly set for the first 1 us following the BREAK detection. After 1 us, the BRKIF bit can be cleared by writing a 0 to the bit. The BREAK interrupt will be generated as long as the BRKIF is 1 and BRKIE is 1.

The latency requirement for interrupt response:

- ❏ Normal operation:~1200 μsec (a SOF and two byte shift times in normal mode)
- ❏ 4X mode:~300 μsec (a SOF and two byte shift times in 4X mode)

0 = Break condition has not been received.
1 = Break condition has been received.

**Bit 0** **RBFIF** Receiver Buffer Full Interrupt Flag.

This bit is set when the receiver posts a received data byte in the C2SIRDB register. When this occurs and RBFIE = 1, the C2SI generates a receive interrupt request.

RBFIF bit remains set as long as the C2SIRDB register is full and can be cleared by a CPU read of the C2SIRDB register or by a DMA transfer from the C2SIRDB register to system RAM.

The latency requirement for interrupt response:

- ❏ Normal operation:~500 μsec (one byte shift time in normal mode)
- ❏ 4X mode:~125 μsec (one byte shift time in 4X mode)

0 = No new data bytes have been received.
1 = A data byte has been received.

## 10.2 C2SI Interrupt Control Register (C2SIICR)

The C2SIICR register consists of interrupt enable control bits. Interrupt flags in the interrupt status register (C2SIISR) are purely status flags and do not initiate interrupt requests alone. Each interrupt flag in the C2SIISR register has a corresponding interrupt enable bit in the C2SIICR register. Any interrupt generating events that cause the interrupt flags to be set, occurring while the interrupt enable is 0 are ignored and lost. Only events that occur after an interrupt enable is 1 initiate an interrupt request.

| hex | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x04 | ARBIE | TIDLIE | TBEIE | TXUOIE | RXOIE | RCCIE | BRKIE | RBFIE |
| | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 |

RW = Read/Write, –*n* = Value after reset

**Bit 7** **ARBIE** Transmit Arbitration Lost Interrupt Enable.

When set to 1, the event (s) which causes ARBIF to be set to 1 also generates a transmit interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of ARBIF; but this does not prevent the ARBIF flag from being set.

0 = ARBIF interrupt disabled.
1 = ARBIF interrupt enabled.

**Bit 6** **TIDLIE** Transmitter Idle Interrupt Enable.

When set to 1, the event(s) which causes TIDLIF to be set to 1 also generates a transmit interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of TIDLIF; but this does not prevent the TIDLIF flag from being set.

0 = TIDLIF interrupt disabled.
1 = TIDLIF interrupt enabled.

**Bit 5** **TBEIE** Transmit Buffer Empty Interrupt Enable.

When set to 1, the event which causes TBEIF to be set to 1 also generates a transmit interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of TBEIF; but this does not prevent the TBEIF flag from being set.

0 = TBEIF interrupt disabled.
1 = TBEIF interrupt enabled.

**Bit 4** **TXUOIE** Transmit Under-run, Over-run Interrupt Enable

When set to 1, the event which causes TXUOIF to be set to 1 also generates a transmit interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of TXUOIF; but this does not prevent the TXUOIF flag from being set.

0 = TXUOIF interrupt disabled.
1 = TXUOIF interrupt enabled.

**Bit 3** **RXOIE** Receive Over-run Interrupt Enable

When set to 1, the event which causes RXOIF to be set to 1 also generates a receive interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of RXOIF; but this does not prevent the RXOIF flag from being set.

0 = RXOIF interrupt disabled.
1 = RXOIF interrupt enabled.

**Bit 2** **RCCIE** Receiver Completion Code Interrupt Enable.

When set to 1, the event which causes RCCIF to be set to 1 also generates a receive interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of RCCIF; but this does not prevent the RCCIF flag from being set.

0 = RCCIF interrupt disabled.
1 = RCCIF interrupt enabled.

**Bit 1** **BRKIE** Received Break Interrupt Enable.

When set to 1, the event which causes BRKIF to be set to 1 also generates a receive interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of BRKIF; but this does not prevent the BRKIF flag from being set.

0 = BRKIF interrupt disabled.
1 = BRKIF interrupt enabled.

**Bit 0** **RBFIE** Receive Buffer Full Interrupt Enable.

When set to 1, the event which causes RBFIF to be set to 1 also generates a receive interrupt request.

When cleared to 0, no interrupt request is possible due to the setting of RBFIF; but this does not prevent the RBFIF flag from being set.

0 = RBFIF interrupt disabled.
1 = RBFIF interrupt enabled.

## 10.3 C2SI Global Status Register (C2SIGSR)


RC = Read/clear, *–x* = Value after reset is indeterminate

**Bits 7-4** **Reserved** Reads are undefined and writes have no effect.

**Bit 3** **IDLE** Data Link Idle Flag.

An idle data link has a lack of activity for more than 280 µsec. This bit reflects the current status of the data link and is set according to current activity.

0 = C2SI data link is busy.
1 = C2SI data link is idle.

**Bit 2** **Reserved** Reads are undefined and writes have no effect.

**Bit 1** **NOISE** Noise detected on C2SI Data Link Flag.

Noise is a pulse of duration less than a normal bit time. This bit is set and remains set after the detection of noise and therefore records the detection of past noise. The digital noise filter will eliminate some of the shorter noise pulses before triggering this bit.

0 = Noise has not been detected.
1 = Noise has been detected by the receiver on the data link.

**Bit 0** **WAKE** Wake up from Low Power Mode Status Flag.

This bit tracks the state of the C2SI wake up output. If ENWAKE = 1 (C2SIGCR.0 = 1) and activity was detected on the data link, WAKE is set. A read of this register clears both the WAKE and the ENWAKE bits. See also the bit description for ENWAKE (C2SIGCR.0)

If the C2SI module had been sleeping, the TMS470 device wakes up when data link activity is detected. When a wake condition occurs, an interrupt is generated back to the CPU, and WAKE is set.

0 = No data on bus or wake up mode not enabled.
1 = Data detected on class II data bus and the wake up mode is enabled.

## 10.4 C2SI Global Control Register (C2SIGCR)

| hex | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x0C | CRCDIS | 4XMODE | T2IFR | NBPOL | Reserved | TBRK | TIFR | ENWAKE |
| | RW-0 | RW-0 | RW-0 | RW-0 | | RW-0 | RW-0 | RW-0 |

RW = Read/Write, *–n* = Value after reset

**Bit 7** **CRCDIS** Generation Disabled Control Bit.

When set, this bit prevents the C2SI module from appending a CRC to the end of transmitted messages including IFRs. It also disables CRC checking during the receipt of a message. Therefore if this bit is set and the incoming message has a CRC appended to it, the C2SI will see the CRC byte at the end of the message as a data byte.

When this bit is cleared, CRC transmission/reception is enabled. A CRC will be appended to the end of messages transmitted, and CRC checking is enabled for the reception of messages. If there is a CRC error during the receipt of a message, then the CRCERR bit (C2SICCSR.4) will be set.

0 = All data packets and transmitted IFRs include a CRC.
1 = No CRC is appended/expected with messages.

**Bit 6** **4XMODE** 4X Mode Control Bit.

When set, this bit puts the C2SI in 4X mode which quadruples the transmit/receive bit rate. This bit is reset whenever a BREAK message is received. Toggling this bit causes the C2SI internal state machine to be reset and forces TIFR (C2SIGCR.1) and TXDMAEN (C2SICTR.3) to be cleared. Also, if this bit is toggled during the reception of a message, the C2SICCSR register is updated and RCCIF is set. A calibration cycle must be performed after setting this bit. For additional information, see section 9.1, *4X Mode*, on page 26.

0 = Operate in normal mode.
1 = Operate in 4X mode.

**Bit 5** **T2IFR** Type 2 In-Frame Response Control Bit.

This bit puts the C2SI into a special type 2 in-frame response mode whereby one-byte In-Frame Responses may be arbitrated. When the T2IFR bit is set, the C2SI automatically requests the one-byte in-frame response previously

loaded and tries to transmit it again at the next byte boundary. Once the transmitter wins arbitration, the byte is gone and the T2IFR bit is automatically cleared by the C2SI. The CRC check needs to be disabled, i.e., CRCDIS = 1 when transmitting Type II IFRs.

The program must set the T2IFR before the response byte is loaded. This bit can only be active when TIFR = 1, as this indicates a response to an in-frame message.

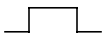
The CRCDIS bit must also be set when T2IFR = 1.

0 = Normal mode.
1 = Enter type II IFR mode. The next transmission by this node will be a type II IFR. The CRCDIS and TIFR bits should be set at the same time or before this bit.

**Bit 4** **NBPOL** Normalization/Bit Polarity Control Bit.

This bit determines which required convention for the normalization bit is to be used.

The NBPOL bit may be changed at any time, but is intended to be initialized once along with any other setup tasks at the beginning of the user program.

| NBPOL | NB | | CRC appended to end of IFR? |
|---|---|---|---|
| 0 |  | 0 | NO |
| 0 |  | 1 | YES |
| 1 |  | 0 | YES |
| 1 |  | 1 | NO |

When NBPOL = 0, the NB bit is decoded as follows:

0 = No CRC follows the in-frame response.
1 = CRC follows the in-frame response.

When NBPOL = 1, the NB bit is decoded as follows:

0 = CRC follows the in-frame response.
1 = No CRC follows the in-frame response.

**Bit 3** **Reserved.** Reads are undefined and writes have no effect.

**Bit 2** **TBRK** Transmit BREAK Sequence Control Bit.

When set, this bit forces the transmitter to send a BREAK sequence. Once sent, this bit automatically clears itself and causes TIFR, 4XMODE (C2SIGCR.1,6) and TXDMAEN (C2SICTR.3) to be cleared also. After the BREAK sequence is transmitted, a completion code is formed in the C2SICCSR register and the RCCIF and BRKIF bits (C2SIISR.2,1) are set. You must wait for the break to finish before writing to the transmit data buffer register (C2SITDB).

0 = No BREAK sent.
1 = Send a BREAK now.

**Bit 1** **TIFR** Transmit In-Frame Response Control Bit.

When set, this bit informs the transmitter to begin transmitting the in-frame response after an EOD. When clear, the transmitter waits for an EOF or an idle data link to transmit a normal message. The transmitter must be suitably enabled by having placed data into the internal shift register (either by DMA or CPU writes).

This bit is automatically cleared if the receiver detects any error, receives a BREAK, or the 4XMODE bit (C2SIGCR.6) is manually reset. The program must clear this bit before the next regular transmitted message.

0 = Start transmission after an EOF or on an IDLE data link.
1 = Start transmission after an EOD.

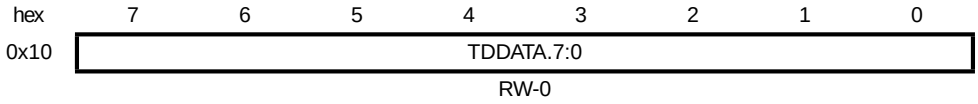
**Bit 0** **ENWAKE** Wake up Enable.

When set, this bit enables the C2SI module to generate a wake up interrupt whenever activity is detected on the data link. (This bit must be cleared to prevent the activation of the wake up signal)

Once a wakeup condition has occurred, ENWAKE is cleared.

0 = Operate in normal mode.
1 = Enable the wake up circuit and enable relaxed SOF timing.

## 10.5 C2SI Transmit Data Buffer Register (C2SITDB)




RW = Read/Write, –*n* = Value after reset

**Bits 7-0** **TDDATA** Transmit Data Buffer Register.

After reset, the C2SITDB register is considered *empty* since nothing has been written to it. After a write to C2SITDB, it is considered *full*. The register remains full until the transmit shift register becomes empty. When this occurs, the contents of a full C2SITDB are transferred to the shift register and are shifted out under the control of the transmitter. The C2SITDB is now empty. Write to the C2SITDB register only when TBEIF = 1.

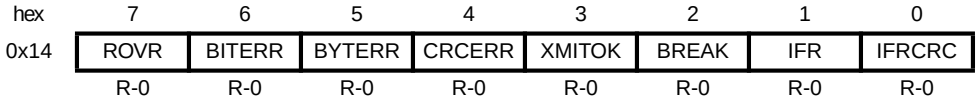
The emptying of C2SITDB causes two events to occur:

- ❏ The setting of TBEIF = 1.
- ❏ The generation of a DMA request by the transmitter.

A write to the C2SITDB register either by the CPU or transmit DMA causes the TBEIF and TIDLIF bits (C2SIISR.5,6) to be cleared.

## 10.6 C2SI Completion Code Status Register (C2SICCSR)

The C2SICCSR register reflects the status of a message being received. Reading this register will not alter the contents of the register. The new completion codes are described in Table 4 on page 43. This registers content cannot be altered by software. It is hardware set and cleared. It is set, by hardware, when the conditions are met and is cleared after receiving the next good byte.




R = Read, –*n* = Value after reset

**Bit 7** **ROVR** Receive Completion Code Over-Run Flag.

This bit is set when the receiver posts two completion codes to the C2SICCSR register that were not separated by a C2SICCSR read.

0 = No over-run
1 = The receiver overran the C2SICCSR register during the last message; completion code data was lost.

**Bit 6** **BITERR** Received an Improperly Timed Bit Error Flag.

0 = No bit timing errors in received message
1 = The received message is corrupted because of a bit timing error.

**Bit 5** **BYTERR** Received an Incomplete Byte Error Flag.

This bit is set when the receiver detects that each byte does not contain exactly 8 bits.

0 = No byte errors in received message
1 = The received message is corrupted because of an incomplete byte error.

**Bit 4** **CRCERR** Received Message with a CRC error Flag.

This bit is set when the CRC, sent by the transmitter along with the data, does not correspond to the data read by the receiver.

0 = CRC correct for received message.
1 = The received message is corrupted as indicated by a CRC error.

**Bit 3** **XMITOK** Received Transmitted Message and Transmit was OK Flag.

This bit is set after receiving a message and the following conditions are satisfied:

- ❏ This C2SI module was the transmitter of the message.
- ❏ The message won arbitration.
- ❏ No reception errors occurred.

This bit is always cleared when a BREAK is received.

0 = Not contending for data link, arbitration was lost, or transmission errors occurred.
1 = Transmitted message was sent successfully.

**Bit 2** **BREAK** Received a BREAK Sequence Flag.

0 = No BREAK received.
1 = Received a BREAK sequence.

**Bit 1** **IFR** Received an In-Frame Response Flag.

This bit is set after the first good byte of an IFR message is received. Since the NB bit indicates that an in-frame response follows, the setting of this bit is an indication that the incoming message is an in-frame response.

0 = An in-frame response has not been received.
1 = An in-frame response has been received.

**Bit 0** **IFRCRC** Received an In-Frame Response with a CRC Flag.

This bit is set when an in-frame response message with a CRC appended to it has been received.

0 = An in-frame response with a CRC has not been received.
1 = An in-frame response with a CRC has been received.

*Table 4. Completion Code Descriptions*

| Completion Code | Description |
|---|---|
| 00h | Received at least 1 byte of a normal message. |
| 02h | Received at least 1 byte of an in-frame response with no CRC transmission. |
| 03h | Received at least 1 byte of an in-frame response with CRC and no transmission. |
| 08h | Received first byte of a normal message that is being transmitted. |
| 0Ah | Received first byte of an in-frame response message that is being transmitted. |
| 0Bh | Received first byte of an in-frame response with CRC that is being transmitted. |
| 04h | Received a BREAK message. |

All bytes transmitted by the C2SI's transmitter are re-directed back into the receive data buffer register (C2SIRDB). Therefore, the C2SICCSR register is updated to reflect the status of not only receiving a message from another C2si, but also transmitting a message to another C2SI, since bytes that are transmitted are re-directed back and received in the C2SI's own C2SIRDB register.

The completion code will update after each receipt of a byte according to the status of receiving that byte. For instance, if there is a bit timing error in the byte being received, then the BITERR bit (C2SICCSR.6) in the C2SICCSR register will be set once that full byte is received.

The received completion code interrupt flag (RCCIF) in the interrupt status register (C2SIISR) is set after an end of frame (EOF) time at the end of a

normal and in-frame response message. The completion code in the C2SICCSR register should be read at these times to ensure that it is not overrun by the next completion code and to clear the C2SICCSR register and RCCIF bit (C2SIISR.2) so that an RCCIF interrupt is not continuously generated.

The resulting completion code at the end of a normal or in-frame response message will remain unchanged until the first byte of the next message (normal message or in-frame response message) is received and this byte is error free. If this first byte is error free then the completion code register resets and updates as conditions to set the bits are detected. If this first byte is not error free, the C2SICCSR register does not become reset and start over, the completion code register will still keep on updating but the RCCIF bit will not get set, and a DMA transfer will not occur.

When the first byte of a message is received without errors and the C2SICCSR register has not been read yet, the completion code of the previous message will be destroyed since the C2SICCSR register is updated to reflect the new message. Since the completion code was not written, no boundary exists to separate the two messages, therefore, the second message is also corrupted.

When a break is detected on the bus, the BREAK bit in the completion code is set, and, depending on the situation, the BYTERR bit, BITERR bit, CRCERR, etc., may be set also. If a transmission of a message was in process when a break is detected, the RCCIF bit is set soon after the detection of the break because a message was aborted. When detecting release of break, the BRKIF will be set again, as well as RCCIF. For additional information, see section 5.3, *Receiving BREAK Messages*, on page 19.

If the C2SI module is transmitting and its receiver overruns the C2SIRDB register, the transmitter continues unaffected. If the C2SI module was waiting to transmit when its receiver overran and the C2SICCSR is not read, the transmitter is held off until the C2SICCSR register is either read by the CPU or DMA.

The RXDMAEN bit (C2SICTR.2) is cleared each time the RCCIF flag is set. Therefore, the receive DMA bit needs to be re-enabled (RXDMAEN = 1) or the C2SIRDB register needs to be read by the CPU before the next incoming byte is received in order to prevent overrun. Overrun occurs when a data byte is received and placed in the C2SIRDB register while the register is full. This destroys the first byte since it was over written by the second.

The contents of the C2SICCSR register are only valid and meaningful when read after the setting of either RBFIF (receive buffer full interrupt) or RCCIF (receive completion code interrupt) (C2SIISR.2, C2SIISR.0).

## 10.7 C2SI Peripheral Control Register (C2SICTR)

| hex | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x18 | Reserved | CALEN | LPEN | Reserved | TXD-MAEN | RXD-MAEN | ESPEN | LPM |
| | | RW-0 | RW-0 | | RW-0 | RW-0 | RW-0 | RW-0 |

RW = Read/Write, –*n* = Value after reset

**Bit 7** **Reserved.** Reads are undefined and writes have no effect.

**Bit 6** **CALEN** Calibration Enable.

Allows the C2SI to calibrate itself with drift from the analog section. When set, the internal calibration logic will adjust itself to compensate for variations within the analog section. This should be used only when the C2SI bus is idle. When calibrated, CALEN will automatically clear. You must calibrate before transmitting in the 4X mode. For additional information, see section 9.4, *Calibration Mode*, on page 28.

0 = C2SI is calibrated.
1 = C2SI is still calibrating itself.

**Bit 5** **LPEN** Loopback Enable.

This bit enables the C2SILPN output to an active low. This places the external analog circuit into a loopback mode for diagnostics. However, if calibration is required, the CALEN bit also needs to be set. Note that this provides an automatic way of using loopback enable. The same functionality can be provided with software control of a general-purpose I/O pin (GPIO)

0 = C2SI is in normal operating mode.
1 = The C2SI directs the external transceiver to tie TX directly to RX.

**Bit 4** **Reserved.** Reads are undefined and writes have no effect.

**Bit 3** **TXDMAEN** Transmit DMA Enable.

Enables the transmit DMA request signal to be generated. For additional information, see section 8, *DMA Interface*, on page 24.

0 = Transmit DMA is not used.
1 = Transmit DMA is used.

**Bit 2** **RXDMAEN** Receive DMA Enable.

Enables the receive DMA request signal to be generated. For additional information, see section 8, *DMA Interface*, on page 24.

0 = Receive DMA is not used.
1 = Receive DMA is used.

**Bit 1** **ESPEN** Emulator Suspend.

Suspends the transmission and reception of data if the CPU is in debug mode. When inactive, the C2SI will continue normal operation during emulation mode. For additional information, see section 9.3, *Emulation Mode*, on page 27.

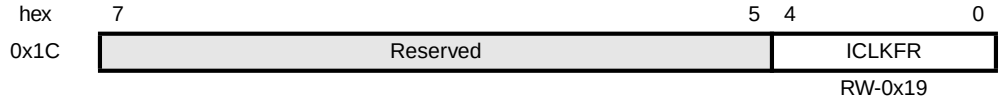
0 = Data is not suspended during emulation mode.
1 = Data is suspended immediately upon entering emulation mode.

**Bit 0** **LPM** Low Power Mode.

When active, the C2SI enters a power down state and disables the C2SIs internal clocks. This bit is the last C2SI bit set going into low-power mode and the first bit cleared coming out of low-power mode. For additional information, see section 9.2, *Low Power Mode*, on page 26.

0 = C2SI is not in low power mode.
1 = C2SI is in low power mode.

## 10.8 Interface Clock Register (C2SICLK)




RW = Read/Write, –*n* = Value after reset

**Bits 7-5** **Reserved.** Reads are undefined and writes have no effect.

**Bits 4-0** **ICLKFR** Interface Clock Frequency.

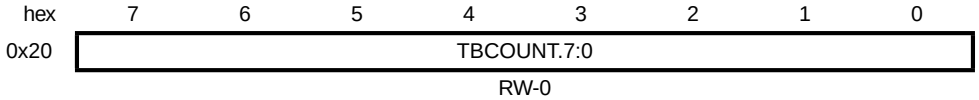
These bits must be set equal to the current C2SI system clock frequency (ICLK). The following table shows the clock frequencies and their corresponding values for this register.

| ICLK (MHz) ICLKFR (4:0) | | ICLK (MHz) ICLKFR (4:0) |
|---|---|---|
| 250x19 | | 170x11 |
| 240x18 | | 160x10 |
| 230x17 | | 150x0F |
| 220x16 | | 140x0E |
| 210x15 | | 130x0D |
| 200x14 | | 120x0C |
| 190x13 | | 110x0B |
| 180x12 | | 100x0A |
| | | 90x09 |

**Note: Note: Non-Specified ICLK Values**

If an illegal combination is used (such as 0x1F), ICLK is assumed to be 25 MHz.

## 10.9 C2SI Transmit Byte Counter (C2SITBC)




RW = Read/Write, –*n* = Value after reset

**Bits 7-0** **TBCOUNT** Transmit Byte Counter.

This register contains the number of data bytes to be transmitted. The CRC byte is not included in this number. The transmit byte counter decrements

each time the transmit data buffer register (C2SITDB) is loaded by CPU or DMA.

When the C2SITBC attempts to decrement past 0 (either from DMA or user software), the C2SI module will automatically append the required CRC byte, if CRC is used (CRCDIS = 0), to the end of the message.

Since C2SITBC is an 8-bit register it allows, a maximum of 255 bytes to be sent out in a single message. See the SAE J1850, Class B Data Communications Network Interface specification for details of the J1850 message protocol and maximum J1850 message length.

## 10.10 C2SI Pin Control Register 1 (C2SIPC1)

| hex | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x24 | TXPOL | RXPOL | TXFUN | RXFUN | LPFUN | TXDIR | RXDIR | LPDIR |
| | RW-1 | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 | RW-0 |

RW = Read/Write, *–n* = Value after reset

A block diagram of the I/O pins controlled by these registers is shown in Figure 2, *C2SI Block Diagram*, page 4.

**Bit 7 TXPOL Transmit Pin Polarity**

Determines the polarity of the transmit pin when the transmit pin is configured as a C2SI pin. An active low pin will place 0 volts to represent the SOF, while an active high pin will place Vcc volts to represent SOF.

0 = Transmit pin is active low
1 = Transmit pin is active high

**Bit 6 RXPOL Receive Pin Polarity**

Determines the polarity of the receive pin when the receive pin is configured as a C2SI pin. An active low pin will expect 0 volts to represent a data 1, while an active high pin will expect Vcc volts to represent a SOF.

0 = Receive pin is active low
1 = Receive pin is active high

**Bit 5** **TXFUN Transmit Pin Function**

Determines whether the transmit pin is to be used as a general-purpose I/O pin or as a C2SI transmit pin. When this pin is inactive, the transmit pin (C2SITXD pin) may be used as an input or output pin, depending on the value of the TXDIR bit. If in C2SI mode, the pin characteristics are determined by the C2SI function bits.

0 = Transmit pin is I/O
1 = Transmit pin is C2SI pin

**Bit 4** **RXFUN** Receive Pin Function

Determines whether the receive pin is to be used as a general-purpose I/O pin or as a C2SI receive pin. When this pin is inactive, the receive pin (C2SIRXD pin) may be used as an input or output pin, depending on the value of the RXDIR bit. If in C2SI mode, the pin characteristics are determined by the C2SI function bits.

0 = Receive pin is I/O
1 = Receive pin is C2SI pin

**Bit 3** **LPFUN** Loop-back Pin Function

Determines whether the loop-back pin is to be used as a general-purpose I/O pin or as a C2SI loop-back pin. When this pin is inactive, the loop-back pin (C2SILPN pin) may be used as an input or output pin, depending on the value of the LPDIR bit. If in C2SI mode, the pin characteristics are determined by the C2SI function bits.

0 = Loop-back pin is I/O
1 = Loop-back pin is C2SI pin

**Bit 2** **TXDIR** Transmit Direction

Controls the direction of the transmit pin when it is used as a general-purpose I/O pin (TXFUN = 0). If the transmit pin is used as an C2SI pin (TXFUN = 1), the TXDIR bit has no effect.

0 = Input
1 = Output

**Bit 1** **RXDIR** Receive Direction

Controls the direction of the receive pin when it is used as a general-purpose I/O pin (RXFUN = 0). If the receive pin is used as an C2SI pin (RXFUN = 1), the RXDIR bit has no effect.

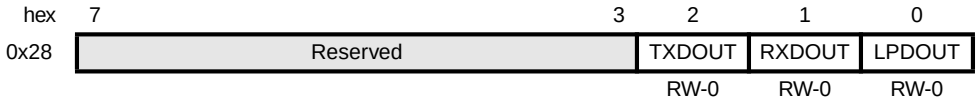
0 = Input
1 = Output

**Bit 0** **LPDIR** Loop-back Direction

Controls the direction of the loop-back pin when it is used as a general-purpose I/O pin (LPFUN = 0). If the loop-back pin is used as an C2SI pin (LPFUN = 1), the LPDIR bit has no effect.

0 = Input
1 = Output

## 10.11 C2SI Pin Control Register 2 (C2SIPC2)




RW = Read/Write, –*n* = Value after reset

**Bits 7-3** **Reserved.** Reads are undefined and writes have no effect.

**Bit 2** **TXDOUT** Transmit Data Out.

Only active if the transmit pin is configured as a general-purpose I/O pin (TXFUN = 0) and configured as an output (TXDIR = 1). The value of this bit indicates the value sent to the transmit pin (C2SITXD pin).

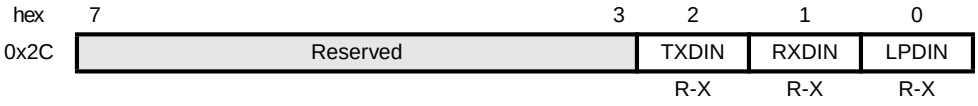
**Bit 1** **RXDOUT** Receive Data Out.

Only active if the receive pin is configured as a general-purpose I/O pin (RXFUN = 0) and configured as an output (RXDIR = 1). The value of this bit indicates the value sent to the receive pin (C2SIRXD pin).

**Bit 0** **LPDOUT** Loop-back Data Out.

Only active if the loop-back pin is configured as a general-purpose I/O pin (LPFUN = 0) and configured as an output (LPDIR = 1). The value of this bit indicates the value sent to the loop-back pin (C2SILPN pin).

## 10.12 C2SI Pin Control Register 3 (C2SIPC3)




R = Read only, –*X* = Value is indeterminate

**Bits 7-3** **Reserved.** Reads are undefined and writes have no effect.

**Bit 2** **TXDIN** Transmit Data In.

The value of this bit reflects the value on the transmit pin (C2SITXD pin).

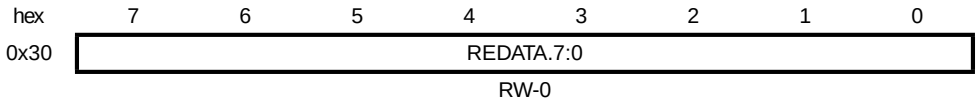
**Bit 1** **RXDIN** Receive Data In.

The value of this bit reflects the value on the receive pin (C2SIRXD pin).

**Bit 0** **LPDIN** Loop-back Data In.

The value of this bit reflects the value on the loop-back pin (C2SILPN pin).

## 10.13 C2SI Receive Data Emulation Buffer Register(C2SIEMU)

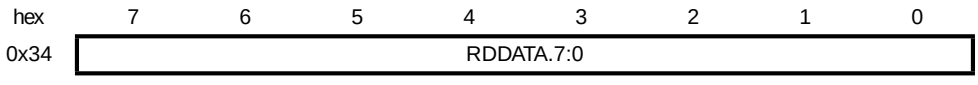



RW = Read/Write, –*n* = Value after reset

**Bits 7-0** **REDATA** Receive Data Emulation Buffer Register.

This register is a mirror image of the receive data buffer register (C2SIRDB). Both the C2SIEMU and the C2SIRDB contain identical values. The only difference between these two registers is that a read from C2SIEMU will not automatically clear the RBFIF interrupt.

## 10.14 C2SI Receive Data Buffer Register (C2SIRDB)




RW = Read/Write, –*n* = Value after reset

**Bits 7-0** **RDDATA** Receive Data Buffer Register.

This register is a buffer following the receiver's shift register. Once the shift register has been filled from receiving a byte, its contents are transferred into the C2SIRDB register and the RBFIF flag is set each time the C2SIRDB register is filled.

All data bytes transmitted out of the transmit data buffer register (C2SITDB) will also be redirected back into the receive data buffer register (C2SIRDB) and will therefore set the RBFIF flag. This may cause confusion, and user software must determine whether the data read from this register is from a transmit or a receive.

Receive DMA transfers use this register or the C2SICCSR register as their source.

A receive interrupt request will be generated if enabled by setting the RBFIE bit = 1. A read of this register by the CPU will clear RBFIF.