ABSTRACT
When using the I²C™ bus protocol, the designer must ensure that the hardware complies with the I²C standard. This application report describes the I²C protocol and provides guidelines on debugging a missing acknowledgment, selecting the pullup resistors, or meeting the maximum capacitance load of an I²C bus. A conflict occurs if devices sharing the I²C bus have the same slave address. This document provides solutions to this conflict by using the devices’ features or external components.

Contents
1 Introduction .................................................................................................................. 2
2 I²C Communication .................................................................................................... 2
2.1 Protocol Description ............................................................................................... 2
2.2 I²C Protocol Screen Shots for CDC(L)949, CDC(L)937, CDC(L)925, CDC(L)913, and CDCE906/907 .................................................................................................................. 2
2.3 Troubleshooting I²C Communication. Reasons for Missing Acknowledgment ................. 4
3 Solving Address Conflict When Sharing I²C Bus ................................................... 5
4 Hardware .................................................................................................................... 8
4.1 Pullup Resistor ....................................................................................................... 8
4.2 Overcoming Maximum Capacitance Load ............................................................. 9
5 References .................................................................................................................. 9

List of Figures
1 I²C Programming Sequence for the CDCE949 ................................................................ 2
2 First Eight Bits of the I²C Pattern: Slave Address and R/W ........................................... 3
3 Second Set of 8 Bits of the I²C Pattern: Command Code ............................................... 4
4 Third Set of 8 Bits of the I²C Pattern Contains Byte to Write/Read (A6xh)......................... 4
5 Slave Address Conflict Solved With I²C Multiplexor .................................................. 8
6 Slave Address Conflict Solved With Bidirectional I²C Buffer ...................................... 8

List of Tables

I²C is a trademark of Koninklijke Philips Electronics N.V.
1 Introduction

Inter-integrated circuit (I\textsuperscript{2}C) (NPX) is a two-wired protocol that can operate at different speeds (standard mode, fast mode, and high-speed mode). Although other protocols are simpler and do not have speed limitation (such as SPI), the scalability offered by I\textsuperscript{2}C made it attractive as an interface for some of TI's CDC products.

I\textsuperscript{2}C prevents data corruption by performing a wired-AND operation as the I\textsuperscript{2}C interface has open-drain or open-collector output. When the SDA line is low that translates in the I\textsuperscript{2}C bus as being busy.

Also, this interface allows the master to check the bus status. By pulling the master SDA line high, it can verify the status of the bus: busy if the line stays low (as some device is pulling the line low), or free if the line is high.

2 I\textsuperscript{2}C Communication

2.1 Protocol Description

The I\textsuperscript{2}C protocol is generated by SDA (bidirectional line: receives I\textsuperscript{2}C pattern and transmits acknowledgment when communication was successful) and SCL (one-direction line).

As the I\textsuperscript{2}C protocol allows multiple slave devices connected to a single I\textsuperscript{2}C bus, each of them is identified with a different address. To address the slave device, seven bits are used.

Transfers are initiated by a start condition (SDA falling edge while SCL is high), and transfers end when a stop condition occurs (rising edge on SDA while SCL is high).

An instruction is sent through the SDA in blocks of one byte, where the first bit is the most-significant bit (MSB). SDA bits must be stable while SCL is high, and they must change while the SCL line is low. If SDA bits are not stable while SCL is high, it is understood as a stop or start condition, a typical cause for a corrupted message or missing acknowledgment.

The number of bytes transmitted is unlimited; each byte must be followed by an acknowledgment bit form the slave confirming the successful reception of the byte. In its absence, the level remains high during the ninth SCL pulse.

![Figure 1. I\textsuperscript{2}C Programming Sequence for the CDCE949](image)

2.2 I\textsuperscript{2}C Protocol Screen Shots for CDC(L)949, CDC(L)937, CDC(L)925, CDC(L)913, and CDCE906/907

The I\textsuperscript{2}C pattern writes in the register’s address A1xh the data A6xh on the CDC(L)949 (slave) whose address is ‘1101100’xb. Screen shots of the protocol can be found in Figure 2, Figure 3, and Figure 4.
The first byte of the I²C protocol contains the device's slave address (1101100xb) and type of operation (0 = write). C2 is SDA and C4 is SCL. The CDCE949 (slave) gives back acknowledgment after the last bit was patterned; only after the SCL last falling edge, the I²C bus is released to handle the next block of communication.

**Figure 2. First Eight Bits of the I²C Pattern: Slave Address and R/W**

The second byte of I²C protocol contains the type of write/read (byte write) and register’s address (1axh) where the operation is performed. C2 is SDA and C4 is SCL. The CDCE949 (slave) gives back acknowledgment after the last bit was patterned; only after the SCL last falling edge, the I²C bus is released to handle the next block of communication.

**Figure 3. Second Set of 8 Bits of the I²C Pattern: Command Code**
Data byte contains byte to write/read (A6xh).
C2 is SDA and C4 is SCL.
The CDCE949 (slave) gives back acknowledgment after the last bit was patterned; only after the SCL last falling edge, the I^2C bus is released to handle the next block of communication.

Figure 4. Third Set of 8 Bits of the I^2C Pattern Contains Byte to Write/Read (A6xh)

2.3 Troubleshooting I^2C Communication. Reasons for Missing Acknowledgment

2.3.1 Timing
Even when using a processor that generates a two-wired interface sequence, timing still must be checked. It is possible that the delay between SCL and SDA is not compliant or too marginal with the defined timing. In those cases, unwanted START or STOP conditions can occur in the middle of the I^2C pattern sent, causing the slave to get a different message or resulting in a missing acknowledgment.

2.3.2 Missing/Unexpected SCL Pulses
An unsuccessful execution occurs when missing or unexpected SCL pulses are within the protocol. If something unexpected happens, it is assured by design that a START condition discards the previous incomplete or erroneous message, and the device is ready to receive the new message.

2.3.3 Incomplete 8-Bit Block
When less than eight bits are sent, the slave waits for the rest of the data. Unless the eighth bit is sent, the slave does not generate an acknowledgment. A way to restart the transmission is starting the new message with a start condition, to send the new I^2C pattern.

2.3.4 Missing Bytes
When a block operation is performed, the amount of bytes to be sent or received may already be fixed in another register. If the number of bytes sent or received is less that the number expected, the state machine waits for the rest of the message. The start condition can be patterned in order to skip the waiting state.
Caveats when "block read" is patterned on CDCE(L)949, CDCE(L)937, CDCE(L)925 and CDCE(L)913:

BCOUNT located in the register's address 08xh bits 7:1 sets the number of bytes that are read when performing a block read. A missing acknowledgment or misunderstood message can occur while performing a block read if the BCOUNT is set to a greater value than the bytes available from the offset set on the command code.

2.3.5 False Slave Address

Each I²C slave device connected to the I²C bus has its own slave address and responds only to that address when an I²C protocol is being patterned.

When the slave device identifies itself with the address being broadcasted in the I²C bus, it ties the SDA signal to ground as can be seen in Figure 2. For that reason, SDA is bidirectional, so that it can handle the acknowledgment received from the slave.

2.3.6 Missing Acknowledgment After Changing the Address

Caveats when programming via I²C the CDCE(L)949, CDCE(L)937, CDCE(L)925, and CDCE(L)913:

During a write instruction (block or byte), each byte change is effective when the last bit of the data byte is patterned.

After changing bits 1:0 from register address 01xh, the slave address changes; therefore, the next I²C message starts with the new slave address. Using the old slave address causes a missing acknowledgment due to a false slave address received.

If during a block write the value of bits1:0 from register address 01xh change, this translates into a different slave address, but this does not impact the communication as the slave address is only checked in the first byte of the block write message.

Caveats when programming via I²C the CDCE906 and CDCE706:

The CDCE906 and CDCE706 slave address is 1101001. When a byte write is performed to set register's address 10xh last 4 bits to 1111, then A0 and A1 control pins overwrite the last two bits of the slave address. When A0 and A1 pins are left floating the pulled up resistors set their value to 1. After that byte write, the slave address is defined by 11010, A0 and A1. If the slave address in the next I²C message is not the new one, then a missing acknowledgment occurs.

If the last two bits of the slave address are set by A0 and A1, and the A0 or A1 pins change, it does not impact the communication as long as this change occurred after the first block of the I²C message (that contains the slave address) was patterned.

If the slave address changes during a block write, the communication is still successful as the connection with the device was already established during the first block of the I²C message.

3 Solving Address Conflict When Sharing I²C Bus

A conflict occurs when more than one device with the same slave address is connected to the same I²C bus.

CDCE(L)949, CDCE(L)937, CDCE(L)925, and CDCE(L)913 belong to the same family of products with the same performance but having different outputs. Their slave address has one part fixed/hardwired and one part programmable. CDCE(L)949 and CDCE(L)937 share the same hardwired address. The same situation applies to CDCE(L)925 and CDCE(L)913. For this reason, if the slave address programmable part changes within each of the defined groups, it is possible to reprogram them to have the same slave address. That is, CDCE(L)949 can be programmed to have the same address as that of the CDCE(L)937. But in this situation, when connected to the same I²C bus, a conflict arises.

The CDCE(L)949, CDCE(L)937, CDCE(L)925, and CDCE(L)913 can re-program the last two bits of their default slave address. Therefore, up to four devices of each can be plugged into the I²C bus.

Each of these devices default setups are set to have different slave address. Therefore, with the default setup, one of each of them has to be able to share the same I²C bus with no conflict.
Solving Address Conflict When Sharing I2C Bus

Some devices offer the possibility to fix the slave address with control signals. Thus, the same device can overwrite its slave address with control pins, and it is unnecessary to preprogram the device.

The CDCE906 and CDCE706 offer the possibility to overwrite the last two bits of their slave address with external control pins 1 and 2. As pins 1 (SO/A0/CLK_SEL) and 2 (S1/A1) are shared with other functionalities, these pins act as A0 and A1 when setting the four lower bits of the internal register 10xh to 1111xb. When A1 and A0 are left floating, the effective slave address is 11, as each of them has internal pullups of 150 kΩ.

A programming EVM board can be used in order to change the setup of all these devices: http://focus.ti.com/docs/toolsw/folders/print/cdcei9xxprogevm.html

If the slave device has a fixed address or if the preceding options are not preferred, then an I²C multiplexor or I²C buffers can be used.

I²C multiplexors split the I²C bus into several subbranches and allow the I²C master to select and address one of multiple identical devices, thus resolving address conflict. The multiplexor connects the main I²C bus to the selected slave device and removes electrically the nonselected devices. This device is programmable via I²C protocol, therefore, no additional pins or control logic are required. TI’s PCA9544A can multiplex up to four slave devices with the same slave address, as can be seen in Figure 5.

If an I²C buffer is chosen, then additional lines (and optional control logic) is needed to enable/disable the I²C bus branch corresponding to the addressed/nonaddressed slave device. This solution (see Figure 6) can be achieved by using TI’s PCA9515A I²C buffer or the PCA9517 I²C buffer; the latter one offers additional voltage translation.
Figure 5. Slave Address Conflict Solved With i²C Multiplexor
If the I²C master has enough spare output pins no control logic would be needed. i.e. In this example if two control lines were available, no additional NOT gate would be needed, as each of these control lines can be connected directly to each EN. The I²C master must take care that only one EN is high.

Control logic that enables only one of the PCA9515A. When the PCA9515A is enabled, it habilitates the I²C communication with the associated CDCE949.

Figure 6. Slave Address Conflict Solved With Bidirectional I²C Buffer

4 Hardware

4.1 Pullup Resistor

The interval of the valid pullup resistors for SDA and SCL is down-limited by the static load specifications and up-limited by the rising and falling edge specification. The minimum resistor value is determined by the maximum current load the output transistor can handle. The I²C specification limit of 3 mA (for standard and fast mode) or 20 mA (for fast mode plus).

\[
R = \frac{(V_{dd_{\text{max}}} - V_{ol_{\max}})}{0.003 \text{ A}}
\]

when \( V_{dd} = 1.8 \text{ V} \) and \( V_{ol_{\text{max}}} = 0.4 \text{ V} \) \( R_{\text{min}} = 466 \Omega \)

The maximum value is calculated from the rise time specifications of the I²C bus:

\[
V(t1) = 0.3 \times V_{dd} = V_{dd} \times \left(1 - e^{\frac{t1}{RC}}\right)
\]

\( t1 = 0.3566749RC \)

\[
V(t2) = 0.7 \times V_{dd} = V_{dd} \times \left(1 - e^{\frac{t2}{RC}}\right)
\]

\( t2 = 1.2039729RC \)

\( t = t2 - t1 = 0.8472979RC \)
For standard-mode I\textsuperscript{2}C-bus: $t = \text{rise time} = 1000 \text{ ns (1 \mu s)}$, so $RC = 1180.2 \text{ ns}$

Example: at a maximum I\textsuperscript{2}C bus load of 400 pF: $R_{\text{max}} = 2.95 \text{ k\Omega}$. For fast-mode: I\textsuperscript{2}C-bus rise time = 300 ns at 400 pF: $R_{\text{max}} = 885 \text{ \Omega}$

### 4.2 Overcoming Maximum Capacitance Load

Adding more I\textsuperscript{2}C and SMBus devices on the bus may exceed the 400-pF limitation. I\textsuperscript{2}C multiplexors, I\textsuperscript{2}C switches, and I\textsuperscript{2}C buffers and repeaters can isolate slave devices that are not currently needed to reduce the overall system loading, and then meet the maximum load capacitance specification.

The I\textsuperscript{2}C multiplexor splits the I\textsuperscript{2}C bus in I\textsuperscript{2}C bus subbranches. I\textsuperscript{2}C multiplexors allow the I\textsuperscript{2}C master to communicate with only one of the multiple devices connected. Although I\textsuperscript{2}C multiplexors are used to solve address conflicts, they can also be used to reduce the load capacitance. They isolate devices that are not needed to reduce the overall system loading. When one of the connected devices is selected, the I\textsuperscript{2}C multiplexor acts as a wire. The cumulative capacitive loading of the main I\textsuperscript{2}C bus and the other active I\textsuperscript{2}C subbranches must be considered; so, care must be taken that each branch does not exceed the 400 pF specified (beyond this, the rising and falling times specifications would be violated).

I\textsuperscript{2}C switches are like I\textsuperscript{2}C multiplexors but more than one device can be selected simultaneously.

I\textsuperscript{2}C buffers/repeaters provide capacitive isolation from the I\textsuperscript{2}C bus, so they are used to go beyond the maximum capacitive load allowed in the bus 400 pF (specified for standard and fast I\textsuperscript{2}C mode)

### 5 References

1. *Interface Guide* (SSZT009)
3. CDCE906, Programmable 3-PLL Clock Synthesizer/Multiplier/Divider data sheet (SCAS814)
4. CDCE949, CDCEL949, Programmable 4-PLL VCXO Clock Synthesizer With 1.8V, 2.5V and 3.3V LVCMOS Outputs data sheet (SCAS844)
7. *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (SPRU131)
8. *TMS320c54x/LC54x/VC54x, Fixed-Point Digital Signal Processors* data sheet (SPRS039)
**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 test 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 the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in medical/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or “enhanced plastic.” Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer’s risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplifiers</td>
<td>Audio</td>
</tr>
<tr>
<td>Data Converters</td>
<td>Automotive</td>
</tr>
<tr>
<td>DLP® Products</td>
<td>Broadband</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Control</td>
</tr>
<tr>
<td>Clocks and Timers</td>
<td>Medical</td>
</tr>
<tr>
<td>Interface</td>
<td>Military</td>
</tr>
<tr>
<td>Logic</td>
<td>Optical Networking</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>Security</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>Telephony</td>
</tr>
<tr>
<td>RFID</td>
<td>Wireless</td>
</tr>
<tr>
<td>RF/IF and ZigBee® Solutions</td>
<td></td>
</tr>
</tbody>
</table>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2009, Texas Instruments Incorporated