Vol 55, No 4—November 2021

I^{2}C Communication Protocol: Understanding I^{2}C Primer, PMBus, and SMBus

Mary Grace Legaspi, Firmware Engineer, and
Eric Peña, Firmware Engineering Manager

Abstract

I^{2}C, or Inter-Integrated Circuit, is a commonly used serial communication protocol in establishing communication between devices, especially for two or more different circuits. I^{2}C Primer is the most commonly used I^{2}C. This article will provide the basic features and standards for I^{2}C Primer, primarily to address proper usage during communication implementation. From the fundamentals of I^{2}C, we'll walk through the availability of its variant subsets, System Management Bus (SMBus) and Power Management Bus (PMBus), and their differences. Each of the three has dedicated functions intended to address different customer requirements.

Why Is It Important?

I^{2}C provides benefits to the designer in establishing simple, 2-way, flexible communication between numerous nodes in a system. I^{2}C reduces complexity by using just two bidirectional wires for transmitting and receiving information. It also allows designers to configure communication between multiple main-node system ICs. I^{2}C also benefits developers in managing the system and power, which gives them the advantage to create a high quality product inside the best possible timeline.

"Communication works for those who work at it."
—John Powell

Communication protocol plays a big role in organizing communication between devices. It is designed in different ways based on system requirements, and these protocols have specific, agreed-upon rules to achieve successful communication.

You've probably used I^{2}C if you've ever built systems with LED displays, sensors, or even accelerometer modules, to name a few. I^{2}C supports the feature of connecting multiple nodes to a single main device and multiple main devices to multiple nodes. This feature is helpful if you want to maximize having one microcontroller logging data to a single memory card or displaying text to a single LCD.

Aside from the most commonly used I^{2}C Primer, there are two additional variations of I^{2}C that focus on the usage for system and power applications. These are called System Management Bus (SMBus) and Power Management Bus (PMBus). → ESTo lo pregunta

By definition, Inter-Integrated Circuit (I^{2}C)—also known as Inter IC—is a hardware communication protocol that uses synchronous communication with a multi-main, multi-node, and serial communication bus. Synchronous communication means two (or more) devices exchanging data share a common clock line. I^{2}C is widely used for connecting lower speed peripheral ICs to processors and microcontrollers. The I^{2}C bus was designed by Philips to allow easy communication between components that reside on the same circuit board.

I^{2}C Primer

Interface

Connections are minimized by using a serial data (SDA) line, a serial clock (SCL) line, and a common ground to carry all communications.

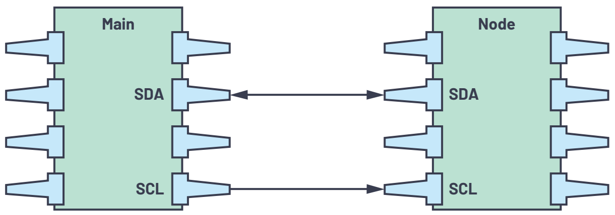


Figure 1. Integrated circuits directly communicate with each other.

There are two wires in each I^{2}C device:

- The SDA is the line for the main device and node to send and receive data.
- The SCL is the line that carries the clock signal. The SCL is always generated by the I^{2}C main. The specification requires minimum periods for the low and high phases of the clock signal.

The I^{2}C bus uses only two bidirectional lines: SDA and SCL for each device for simple inter-IC communication.

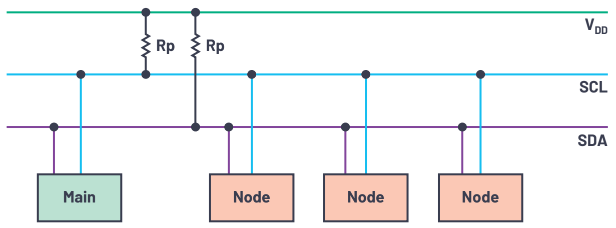


Figure 2. An I²C pull-up resistor connection.

The most important hardware parts are pull-up resistors, which are used by the SDA and SCL lines. I²C compatible devices connect to the bus with open collector or open drain pins, which pull the line low. When there is no transmission of data, the I²C bus lines idle in a high state; the lines are passively pulled high. Transmission occurs by toggling the lines, by pulling low and releasing high. Bits are clocked on falling clock edges.

Open drain outputs require a pull-up resistor (Rp in Figure 2) in order to properly output high. The pull-up resistor is connected between the output pin and the output voltage (V_{DD} in Figure 2) that is desired for a high state.

Four thousand seven hundred ohms (4700 Ω) is the most used value of pull-up resistors with typical values of V_{CC} and V_{DD} (5 V).

~~For reference, shielded 2 AWG twisted pair cables have capacitance in the range of 100 pF/m to 240 pF/m. So, the maximum bus length of an I²C link is about 1 meter at 100 kBaud, or 10 meters at 10 kBaud. Unshielded cable typically has much less capacitance but should only be used within an otherwise shielded enclosure.~~

Table 1 summarizes the key characteristics of I²C.

Table 1. I²C Summary

Features	Specs
Wires	2
Maximum Speed	Standard mode = 100 kbps Fast mode = 400 kbps High speed mode = 3.4 Mbps Ultrafast mode = 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Maximum Number of Mains	Unlimited
Maximum Number of Nodes	1008

Theoretically, the maximum number of nodes is 2^7 or 2^{10} for addressing mode; however, 16 addresses are reserved for special purposes.

I²C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the main and the node. The clock signal is always controlled by the main. El main lleva la batuta

Reserved I²C Node Address

There are 16 reserved I²C addresses. These addresses correspond to one of two patterns: 0000 XXX or 1111 XXX. Table 2 shows I²C addresses reserved for special purposes.

Table 2. I²C Reserved Addresses

I²C Node Address	R/W	Bit Description
0000 000	0	General call address
0000 000	1	Start byte
0000 001	X	CBUS address
0000 010	X	Reserved for different bus format
0000 011	X	Reserved for future purposes
0000 1XX	X	High speed-mode main code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit node address

How Does I²C Work?

I²C data is transferred in messages, which are broken up into frames of data. The read and write protocol contains the address frame with the binary address of the node and another data frame that contains the data being transmitted, start and stop conditions, repeated start bits, read/write bits, and acknowledge/not acknowledge bits between each data frame.

Timing Specification Table

The I²C timing table is also important as it allows engineers to design ICs that are compatible with the bus requirement. Each data rate has its own timing specification to which the main and node must adhere for correct data transfer.

Table 3 shows the symbols and parameters available on a timing specification table.

Table 3. Sample I²C Timing Specification Table

Symbol	Parameter	Units
f_{SCL}	SCL clock frequency	kHz
$t_{HD(STA)}$	Hold time (repeated) start condition	µs
t_{LOW}	Low period of the SCL pin	µs
t_{HIGH}	High period of the SCL pin	µs
$t_{SU(STA)}$	Set-up time for a repeated start condition	µs
$t_{HD(DAT)}$	Data hold time	µs
$t_{SU(DAT)}$	Data set-up time	ns
t_r	Rise time for SDA signals	ns
t_f	Fall time for SDA signals	ns
$t_{SU(STO)}$	Set-up time for stop condition	µs

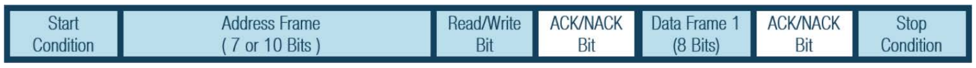


Figure 3. An I²C message.

I^{2}C Transmission Sub-Protocols

Transmission over the bus is either a read or write operation. The reading and writing protocols build upon a series of sub-protocols such as start and stop conditions, repeated start bits, address bytes, data transfer bits, and acknowledge/not acknowledge bits.

Start Condition

As the name suggests, a start condition always occurs at the start of a transmission and is initiated by the main device. This is done to wake the idling node devices on the bus. The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low. See Figure 4.

Repeated Start Condition

Without issuing a stop condition, a start condition can be repeated during a transmission. This is a special case, called the repeated start, and is used for changing data transmission direction, repeating transmission attempts, synchronizing several ICs, or even controlling serial memory. See Figure 5.

Address Frame

The address frame contains a 7-bit or 10-bit sequence, depending on the availability (refer to the data sheet). See Figure 6.

I^{2}C doesn't have node select lines like SPI, so it needs another way to let the node know that data is being sent to it, and not another node. It does this by addressing. The address frame is always the first frame after the start bit in a new message.

The main sends the address of the node it wants to communicate with to every node connected to it. Each node then compares the address sent from the main to its own address. If the address matches, it sends a low voltage ACK bit back to the main. If the address doesn't match, the node does nothing, and the SDA line remains high.

Read/Write Bit

The address frame includes a single bit at the end that informs the node whether the main wants to write data to it or receive data from it. If the main wants to send data to the node, the read/write bit is at a low voltage level. If the main is requesting data from the node, the bit is at a high voltage level. See Figure 7.

ACK/NACK Bit

Each frame in a message is followed by an acknowledge/not acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

Legend: For the following figures, the white box represents the node while the blue box is for the main. See Figure 8.

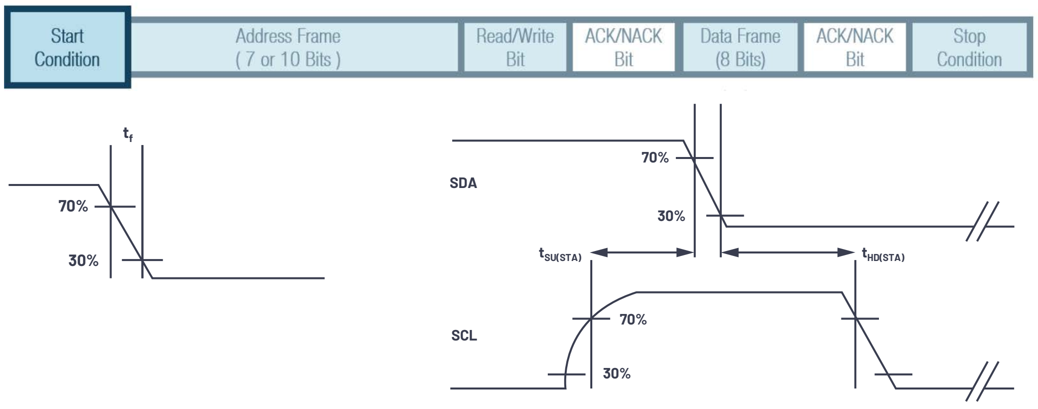


Figure 4. Start condition.

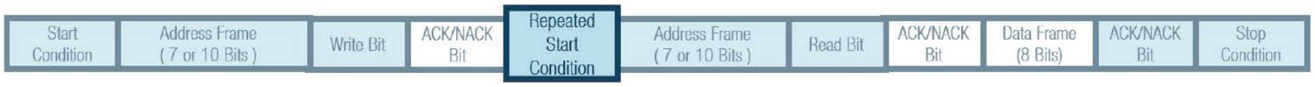


Figure 5. Repeated start condition.

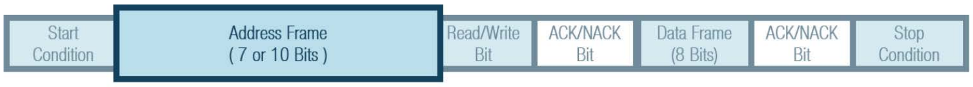


Figure 6. Address frame.

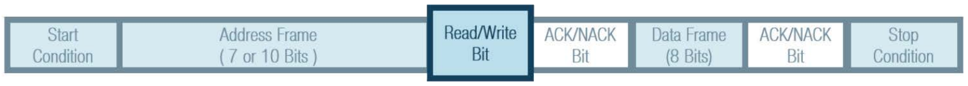


Figure 7. Read/write bit.

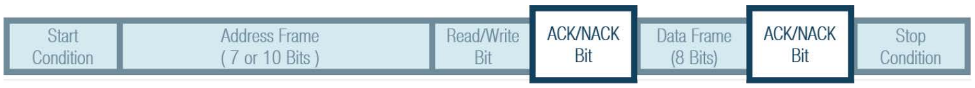


Figure 8. Acknowledge/not acknowledge bit.

Data Frame

After the main detects the ACK bit from the node, the first data frame is ready to be sent. The data frame is always 8 bits long and sent with the most significant bit first. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully. The ACK bit must be received by either the main or the node (depending on who is sending the data) before the next data frame can be sent. See Figure 9.

Stop Condition

After all the data frames have been sent, the main can send a stop condition to the node to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low-to-high transition on the SCL line, with the SCL line remaining high.

The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high. See Figure 10.

Steps of I²C Transmission: Write

For an example of I²C transmission of a write single location, see Figure 11.

Step 1

The main sends the start condition to every connected node by switching the SDA line from a high voltage level to a low voltage level *before* switching the SCL line from high to low.

Step 2

The main sends each node the 7-bit or 10-bit address of the node it wants to communicate with, along with the write bit.

For example, a 7-bit address is 0x2D. Adding a write bit equivalent to 0, it will be 0x5A.

Step 3

Each node compares the address sent from the main to its own address. If the address matches, the node returns an ACK bit by pulling the SDA line low for one bit. If the address from the main does not match the node's own address, the node leaves the SDA line high.

ACK is possible by bringing the SDA line low during the ninth pulse of the SCL and floating high for NACK.

Step 4

The main sends or receives the data frame.

Step 5

After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.

Step 6

To stop the data transmission, the main sends a stop condition to the node by switching SCL high before switching SDA high.

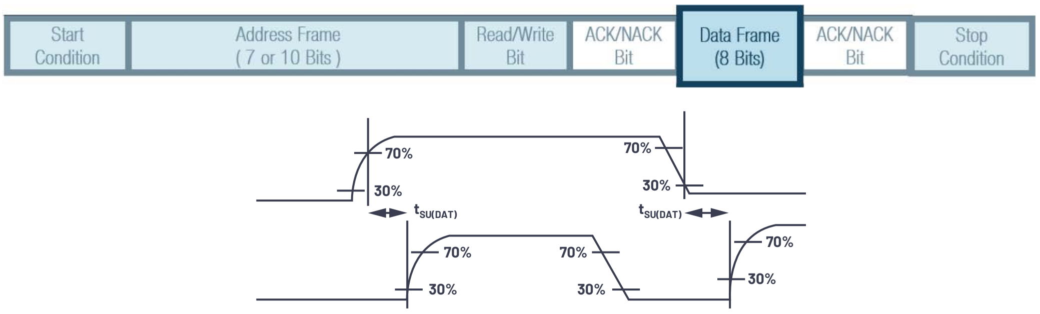


Figure 9. Data frame.

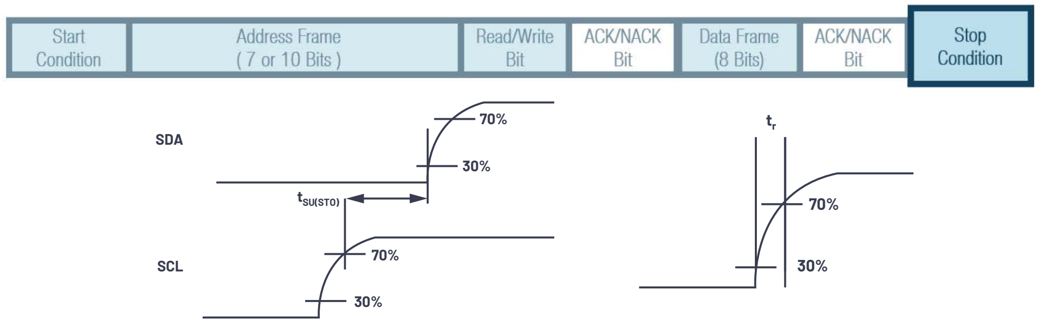
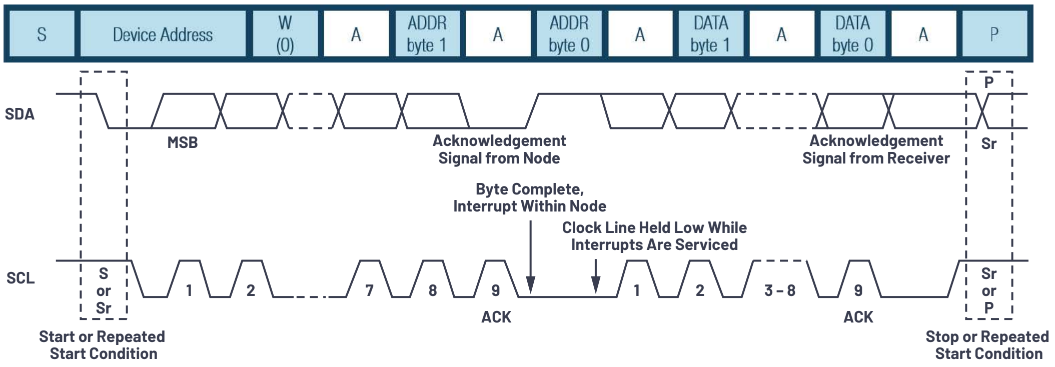


Figure 10. Stop condition.



S
Device Address
W (0)
A
ADDR byte 1
A
ADDR byte 0
A
DATA byte 1
A
DATA byte 0
A
P
SDA
MSB
Acknowledgement Signal from Node
Acknowledgement Signal from Receiver
P
Sr
Byte Complete, Interrupt Within Node
Clock Line Held Low While Interrupts Are Serviced
SCL
S or Sr
1
2
7
8
9
ACK
1
2
3 - 8
9
ACK
Sr or P
Start or Repeated Start Condition
Stop or Repeated Start Condition

Figure 11. Data sheet example for I²C transmission of write single location.

S	Device Address	W (0)	A	ADDR byte 1	A	ADDR byte 0	A	S	Device Address	R (1)	A	DATA byte 0	A	DATA byte 1	A	P		

Figure 12. Data sheet example for I²C transmission of read single location.

Steps of I²C Data Transmission: Read

Step 1

The main sends the start condition to every connected node by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low.

Step 2

The main sends each node the 7-bit or 10-bit address of the node it wants to communicate with, along with the write bit.

For example, a 7-bit address is 0x2D. Adding a write bit equivalent to 0, it will be 0x5A.

Step 3

Each node compares the address sent from the main to its own address. If the address matches, the node returns an ACK bit by pulling the SDA line low for one bit. If the address from the main does not match the node's own address, the node leaves the SDA line high.

Step 4

After the initial start, addressing, and acknowledge, since the main already knows its node and the address to point to, some devices have a repeated start condition to clean the transaction.

Note: For reading purposes only!

Step 5

The main sends each node the 7-bit or 10-bit address of the node it wants to communicate with, along with the read bit.

For example, a 7-bit address is 0x2D. Adding a read bit equivalent to 1, it will be 0x5B.

Step 6

Each node compares the address sent from the main to its own address. If the address matches, the node returns an ACK bit by pulling the SDA line low for one bit. If the address from the main does not match the node's own address, the node leaves the SDA line high.

Step 7

After the ACK bit, the main receives the data frame from the node.

Step 8

After each data frame has been transferred, the main returns another ACK bit to the sender to acknowledge successful receipt of the frame, or the main returns a NACK if the read request is already done.

Step 9

To stop the data transmission, the main sends a stop condition to the node by switching SCL high before switching SDA high.

Single Main with Multiple Nodes

Because I²C uses addressing, multiple nodes can be controlled from a single main. With a 7-bit address, 128 (2^7) unique addresses are available. Using 10-bit addresses is uncommon but provides 1024 (2^{10}) unique addresses. To connect multiple nodes to a single main, wire them with 4.7 kΩ pull-up resistors connecting the SDA and SCL lines to V_{CC}.

Multiple Mains with Multiple Nodes

Multiple main devices can be connected to a single node or multiple nodes. The problem with multiple main devices in the same system comes when two mains try to send or receive data at the same time over the SDA line.

To solve this problem, each main needs to detect if the SDA line is low or high before transmitting a message.

If the SDA line is low, this means that another main has control of the bus, and the main should wait to send the message. If the SDA line is high, then it's safe to transmit the message. To connect multiple mains to multiple nodes, use the diagram shown in Figure 13, with 4.7 kΩ pull-up resistors connecting the SDA and SCL lines to V_{CC}.

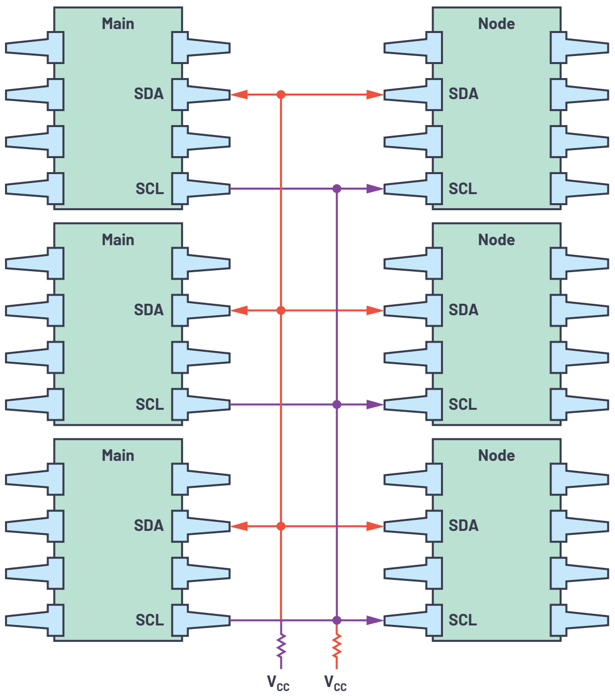


Figure 13. Multiple mains with multiple nodes connected.

Arbitration

Several I²C multi-main devices can be connected to the same I²C bus and operate concurrently. By constantly monitoring the SDA and SCL for start and stop conditions, they can determine whether the bus is currently idle or not. If the bus is busy, the main device delays pending I²C transfers until a stop condition indicates that the bus is free again.

However, it may happen that two main devices start a transfer at the same time. During the transfer, the mains constantly monitor the SDA and SCL. If one of them detects that SDA is low when it should be high, it assumes that another main is active and immediately stops its transfer. This process is called *arbitration*. Both mains generate a start bit and proceed with their transmissions.

If the mains happen to choose the same logic levels, nothing happens.

As soon as the mains attempt to impose different logic levels, the main driving the signal low is proclaimed the winner; the loser detects the logic mismatch and abandons its transmission.

Take a moment to appreciate the simplicity and efficacy of this arrangement:

- The winner continues its transmission without interruption—no corrupted data, no driver contention, no need to restart the transaction.
- Theoretically the loser could monitor the node address during the arbitration process and make a proper response if it happens to be the addressed node.
- If the competing mains are both requesting data from the same node, the arbitration process does not unnecessarily interrupt either transaction—no mismatch will be detected, and the node will output its data to the bus such that multiple mains can receive it.

Clock Stretching

This is also referred to as clock synchronization.

Note: The I²C specification does not specify any timeout conditions for clock stretching—that is, any device can hold down SCL if it likes.

In I²C communication protocol, the clock speed and signals are always generated by the main device. The signal generated by the I²C main device provides synchronization between the main and node connection.

There are instances where a node or subnode is not fully working and needs to slow down prior to receiving the generated clock from the main. This is accomplished through a mechanism known as clock stretching.

During the clock stretching period, in order to reduce the bus speed, the node is allowed to hold down the clock. While on the main side, it is necessary that, after its high state, the clock signal must be read back. Then it must wait until the line has reached the high state.

Bandwidth

While clock stretching is a common practice, there is an effect on the bandwidth side. In using clock stretching, the total bandwidth among the shared bus might be significantly decreased. Bus performance must still be reliable and fast even using this technique. It is necessary to cover the estimated effects of using clock stretching, especially on I²C buses shared by multiple devices.

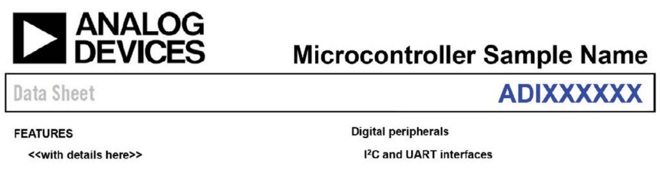

ANALOG DEVICES
Microcontroller Sample Name
Data Sheet ADIXXXXXX
FEATURES
<<with details here>>
Digital peripherals
I²C and UART interfaces

Figure 14. Microcontroller data sheet.

Clock stretching allows an I²C node device to force the main device into a wait state. A node device may perform clock stretching when it needs more time to manage data—for example, to store received data or prepare to transmit another byte of data. This typically occurs after the node device has received and acknowledged a byte of data.

Which I²C Node Devices Need Clock Stretching?

Whether or not clock stretching is needed depends on the functions of the node device. Here are two examples:

- A processing device, such as a microprocessor or a microcontroller, may need additional time to process an interrupt, receive and manage data, and perform the appropriate function(s).
- A simpler device, such as an EEPROM, does not process data internally, so it does not need clock stretching to perform any of its functions.

I²C Data Sheet Sample Overview

There are different approaches in terms of creating a data sheet depending on different companies and manufacturers. As a starter, Figure 13 shows a sample data sheet and basic I²C details including registers and electronic specifications.

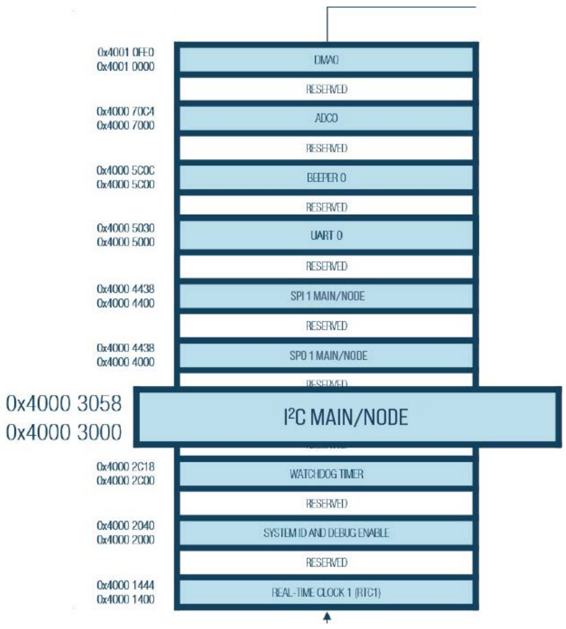


Figure 15. Microcontroller memory map.

Table 4 shows the most used I²C registers. Names and descriptions may vary depending on the data sheet, but the functionality and usage is common.

Table 4. I²C Register Description

Name	Description
I2C_ADDR1	Main address byte 1
I2C_ADDR2	Main address byte 2
I2C_BYT	Start byte
I2C_ID	Node address device ID
I2C_MCTL	Main control
I2C_MRX	Main receive data
I2C_SCTL	Node control
I2C_SRX	Node receive
I2C_STAT	Main and node FIFO status

The creation of I²C may vary depending on usage. Table 5 shows a sample of basic I²C driver API requirements.

Table 5. I²C Driver Development

Main	Node
Initialization	
Tx Handler	Tx Handler
Rx Handler	Rx Handler
Event Interrupt	
Error Interrupt	

SMBus → Apareció en los exámenes

SMBus is known to be used in applications that require critical monitoring of parameters. It is most commonly used in computer motherboards and embedded system applications. It has additional monitoring specifications for temperature, supply voltage, fan monitor, and/or control integrated chips.

The SMBus is a 2-wire bus that is similar to the I²C bus that was developed by Philips in the 1980s. The two main signals are the clock, or SMBCLK, and data, or SMBDAT. The I²C primer and SMBus are compatible with each other, but there are notable differences such as:

- The SMBus logic level thresholds are fixed and not proportional to a device's supply voltage. This allows devices with different supply voltages to operate on the same primer. For example, one SMBus might have devices powered from 1.8 V, 3.3 V, and 5 V.
- They both operate on the same speed up to 100 kHz, but the I²C primer has 400 kHz and 2 MHz versions.
- The SMBus provides for a minimum clock speed and limits the amount the clock may be stretched in one transaction. A violation of the timeout limits causes all SMBus devices to reset their I/O logic to allow the bus to restart. This enhances the robustness of the bus.
- The timeout for both is also different. The I²C primer has no timeout while the SMBus has a timeout—consider 35 ms for a 10 kHz minimum clock speed.
- Packet error checking (PEC) was originally defined for SMBus. A packet error code byte is added at the end of each transaction.
- Some of the remaining differences involve transfer types, alert line, suspend line, and power down or up.

It is an explicit requirement that an SMBus device must acknowledge (ACK) its own address every time it is received, regardless of what else the device may be doing. This assures that a main device can accurately determine what devices are active on the bus.

All SMBus transactions are carried out through one of the specified SMBus protocols.

The SMBus includes an optional signal, SMBALERT#, that node devices can use to quickly notify the main or system host that it has information for the main, such as reporting a fault condition. → me imagino q' de acá viene el nombre

SMBus: otra versión del I2C.

System → management

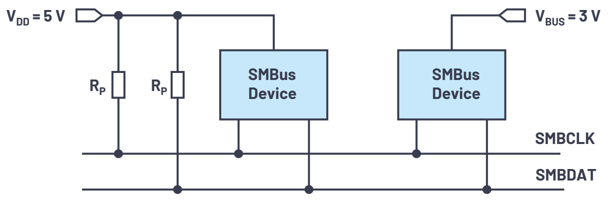


Figure 16. SMBus topology.

SMBus Pull-Up Circuitry

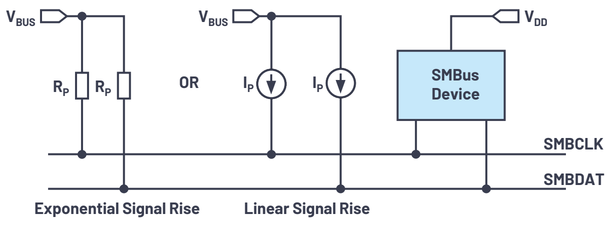


Figure 17. SMBus pull-up circuitry.

SMBus Address

SMBus addresses are 7 binary bits long and are conventionally expressed as 4 bits followed by 3 bits followed by the letter b—for example, 0001 110b. These addresses occupy the high 7 bits of an 8-bit field on the bus. The low bit of this field, however, has another semantic meaning that is not part of an SMBus address.

Node Address (7)

Figure 18. Node address.

A 7-bit destination address is sent from the main to one or more devices (such as with the general call address) on the bus.

Note the start and stop conditions are transitions, not bits, and are shown without a bit count number above the symbol. When shown in a transaction diagram, the repeated start is also a transition, not a bit, and is shown without a bit count above the symbol.

S	Node Address (7)	W (0)	A (1)	DATA BYTE (8)	A (1)	P

Figure 19. An SMBus message.

SMBus Timing Measurements

Table 6. SMBus Parameters

Symbol	Parameters	Units
f_{SMB}	SMBus operating frequency	kHz
t_{BUF}	Bus free time between stop and start condition	µs
$T_{HD\text{-}STA}$	Hold time after (repeated) start condition	µs
$T_{SU\text{-}STA}$	Repeated start condition set-up time	µs
$t_{SU(STO)}$	Stop condition set-up time	µs
$t_{HD(DAT)}$	Data hold time	ns
$t_{SU(DAT)}$	Data set-up time	ns
$t_{TIMEOUT}$	Detect clock low timeout	ms
t_{LOW}	Clock low period	µs
t_{HIGH}	Clock high period	µs

PMBus: Power Management Redefined

In addition to SMBus comes a variant, the PMBus, which is an open standard power management protocol. This flexible and highly versatile standard allows for communication between devices based on both analog and digital technologies, and provides true interoperability, which will reduce design complexity and shorten time to market for power system designers.

PMBus is used in digital management of power supplies with power control and management components. It has commands and structures to support the requirement for power management. This means both I²C primer and PMBus are compatible and interoperable on the electrical requirements and command semantics.

One of the essential parameters in power management is monitoring of overvoltage level, and PMBus provides a command for setting and reading it. By adding to the available features of I²C primer and SMBus, PMBus acts as a protocol layer on top of the existing standards, especially the SMBus.

The I²C specification only describes the physical layer, timing, and flow control of a 2-wire bus. The I²C specification does not describe the format of messages (like the SMBus protocols) and does not describe the content of the messages.

The PMBus specification is a complete power management protocol. It includes how to get the bits and bytes from one device to another (that is, transport); it also describes a command language that gives meaning to those bits and bytes.

Addressing

For redundant systems there are up to three signals to set the address location of the power supply once it is installed in the system: Address2, Address1, and Address0. For non-redundant systems, the power supply device address location should be B0h.

Hardware

The device in the power supply shall be compatible with SMBus 2.0 high power specifications for I²C V_{DD}-based power and drive (for V_{DD} = 3.3 V). This bus shall operate at 3.3 V.

Power Sourcing

The circuits inside the power supply shall derive their power from the standby output. For redundant power supplies, the device(s) shall be powered from the system side of the OR'ing device. The PMBus device shall be on whenever AC power is applied to the power supply or a parallel redundant power supply in the system.

Pull-Ups

Only weak pull-up resistors shall be on the SCL or SDA lines inside the power supply. The main pull-up resistors are provided by the system and may be connected to 3.3 V or 5 V. For the system design, the main pull-ups shall be located external to the power supply and derive their power from the standby rail.

Data Speed

The PMBus device in the power supply shall operate at the full 100 kbps SMBus speed and avoid using clock stretching as much as possible, as it can slow down the bus.

Summary

Table 8 provides a general view and summary in terms of specifications: signaling, timing, and electrical among I²C Primer, SMBus (both for high and low power), and PMBus.

How Are I²C Primer, SMBus, and PMBus Related?

Originally developed to facilitate battery management systems, SMBus uses I²C hardware but adds second-level software, which ultimately allows devices to be hot swapped without restarting the system. PMBus extends SMBus by defining a set of device commands specifically designed to manage power converters, exposing device attributes such as measured voltage, current, temperature, and more. In general, I²C primer, SMBus, and PMBus devices can share a bus without any major issues.

I²C, SMB, PMB Advantages

- Only uses two wires
- With ACK/NACK bit
- Well-known protocol
- Supports multiple main devices and multiple nodes
- Hardware is less complicated than UART
- Widely used method

Disadvantages

- Slower data transfer rate than SPI
- Size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI

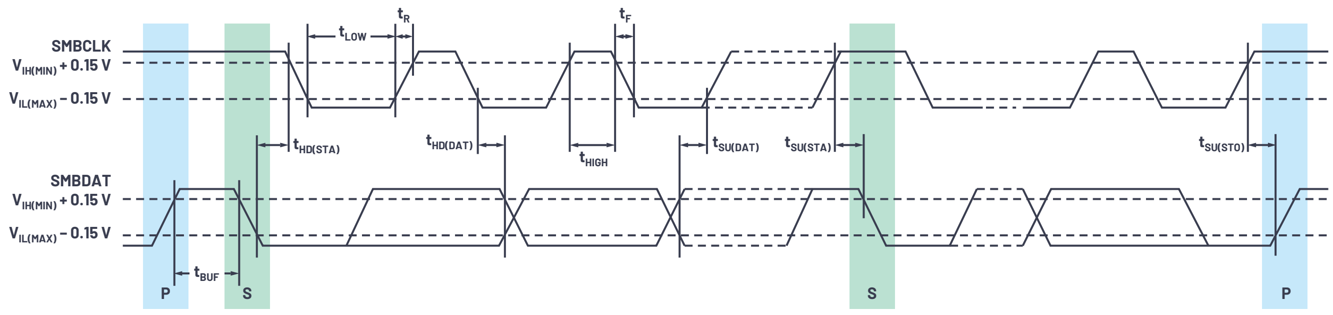


Figure 20. SMBus timing measurements.

Table 7. PMBus Addressing

Addresses Used	Main Addressing Used for Most Server Power Supplies with Two Addressing Pins				Additional Addresses If Three Addressing Pins Are Provided on the Power Supplies			
System Addressing Address2/Address1/Address0	0/0/0	0/0/1	0/1/1	0/1/1	1/0/0	1/0/1	1/1/0	1/1/1
PMBus Device Read Addresses	B0h/B1h	B2h/B3h	B4h/B5h	B6h/B7h	B8h/B9h	BAh/BBh	BCh/BDh	BEh/BFh

Use Cases

- Sensor reading
- Sensor writing
- EEPROM, temperature sensor, touch screen, proximity
- Transmitting and controlling user directed actions
- Communicating with multiple microcontrollers
- Consumer electronics
- System management
- Power management
- Debugging

References

"Advantages and Limitations of I²C Communication." Total Phase, August 2016.

Afzal, Sal. "I²C Primer: What Is I²C? (Part 1)." Analog Devices, Inc.

Afzal, Sal. "I²C Timing: Definition and Specification Guide (Part 2)." Analog Devices, Inc.

Campbell, Scott. "Basics of the I²C Communication Protocol." Circuit Basics.

I²C Quick Guide. Analog Devices, Inc.

"I²C—What's That?" I²C Bus.

Table 8. Summary Specifications for I²C Primer, SMBus, and PMBus

Specification		I²C Primer	SMBus High Power	SMBus Low Power	PMBus
Signaling	Packet Error Checking (Optional)	—	—	—	—
	SMBALERT (Optional)	—	—	—	—
	Block Size Limit	—	32 bytes	32 bytes	255 bytes
Timing	Data Rate:	100 kbps	100 kbps	100 kbps	100 kbps
	• Standard Mode	400 kbps	—	—	400 kbps
	• Fast Mode	1 Mbps	—	—	—
	• Fast Mode Plus	3.4 Mbps	—	—	—
	• High Speed Mode	0 Hz to 3.4 MHz	10 kHz to 100 kHz	10 kHz to 100 kHz	10 kHz to 400 kHz
	Clock Speed		25 ms to 35 ms	25 ms to 35 ms	25 ms to 35 ms
	Bus Timeout	—	50 μs	50 μs	50 μs
	Bus Main Request Delay (Min)	—	—	—	—
	SCL Hold Time (Max)	—	2 ms	2 ms	2 ms
	Data Hold Time (Min)	—	300 ns	300 ns	300 ns
Electrical	Capacitance Load per Bus Segment (Max)	400 pF	400 pF	—	400 pF
	Rise Time (Max)	1 μs at 100 kHz, 300 ns at 400 kHz	1 μs	1 μs	1 μs at 100 kHz, 300 ns at 400 kHz
	Pull-Up Current at 0.4 V (Max)	3 mA (standard and fast mode)	4 mA	350 μA	4 mA
	Leakage Current per Device (Max)	±10 μA	±10 μA	±5 μA	±10 μA
	VIL Input Logic Low Threshold (Max)	0.3 V_{DD} or 1.5 V	0.8 V	0.8 V	0.8 V
	VIH Input Logic High Threshold (Min)	0.7 V_{CC} or 3 V	2.1 V	2.1 V	2.1 V
	VOL Output Logic Low Threshold (Max)	0.4 V	2.4 V	0.4 V	0.4 V

About the Author

Mary Grace Legaspi is a firmware engineer and part of the Design and Layout Team working with the Consumer Software Engineering Group at Analog Devices. She joined ADI in Cavite, Philippines in September 2018. She graduated from Tarlac State University with a bachelor's degree in electronics engineering. She is currently studying toward a Master of Management at the University of the Philippines. She can be reached at mary.legaspi@analog.com.

About the Author

Eric Peña is a firmware engineering manager and part of the Design and Layout Team working with the Industrial Platform and Networking Group at Analog Devices. He joined ADI in Cavite, Philippines in April 2019. He graduated from Adamson University in Manila with a bachelor's degree in computer engineering. Eric previously worked at Technology Enabler Designer as a firmware engineer and also as a systems engineer at Fujitsu Ten Solutions. He can be reached at eric.pena@analog.com.