How to Debug I²C Devices Using an Oscilloscope

Vrajesh Davé
Product Development Engineer, LeCroy Corporation

Phil Luu
Technical Product Manager, NXP Semiconductors

Objectives

- ✓ Understand I²C protocol for embedded design
- ✓ Learn how to develop an I²C based system
- ✓ Learn debugging techniques for I²C-bus protocol using an Oscilloscope
- ✓ Understand I²C electrical parameters and debugging them using an Oscilloscope
- ✓ Be able to choose the correct I²C devices
- ✓ Know where to go and get help

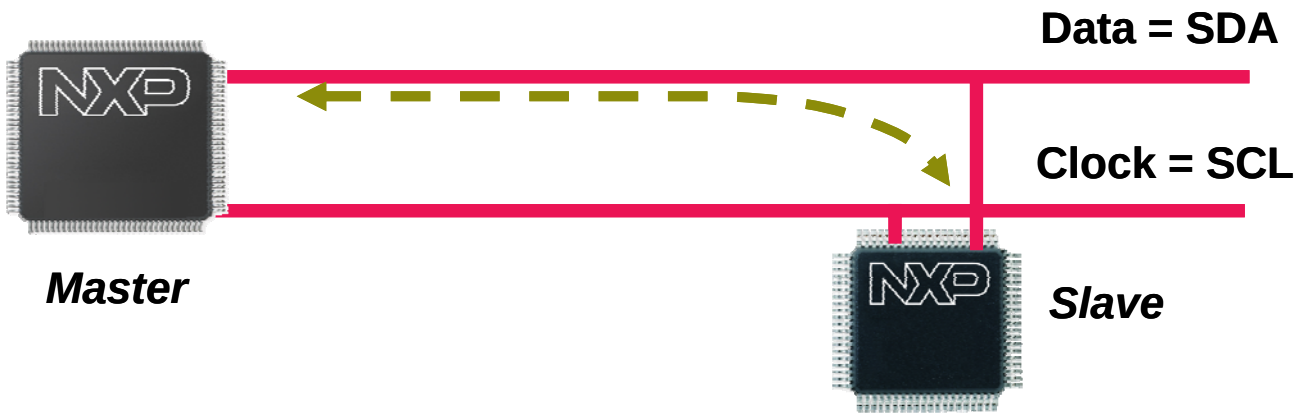
Agenda

- I²C-bus Introduction
- Bus Protocol
- I²C Protocol Decoding Technique – using an Oscilloscope
- Multi-master Synchronization & Arbitration
- Bus recovery
- Fast-mode Plus
- Oscilloscope Scope triggering to debug embedded circuits
- Electrical Characteristics
- Using Oscilloscope to make meaningful measurement
- Application
- Useful links for I²C products and support
- I²C-bus Summary
- Q&A

I²C-bus Introduction

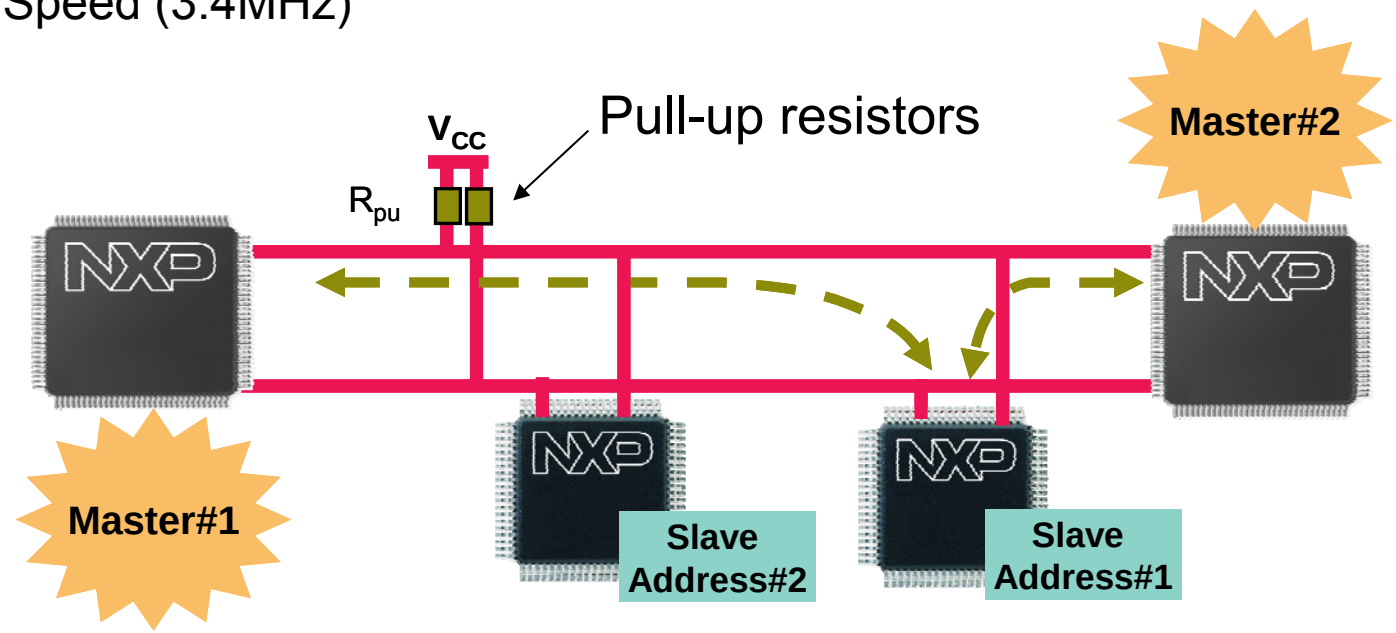
What is I²C-bus

- I²C™ bus = Inter-IC bus, invented by NXP Semiconductors (formerly Philips Semiconductors)
- Bi-directional 2-wire serial bus
  - Serial data (SDA), serial clock (SCL)
- Master-slave communication scheme



What is I²C-bus (cont.)

- Multi-masters capable
- Each slave device has a unique address
- I²C I/O are open-drain/collector requiring pull-up resistors
- Standard mode (100kHz), Fast mode (400kHz), Fast mode-Plus (1MHz), High Speed (3.4MHz)



I²C-bus – What's New?

- Version 3 spec released, June 2007 includes fast-mode Plus, 1MHz
- More and more major OEMs/Industries are adopting I²C-bus as the "system management bus"
- DDC, PMBus (as well as SMBus) are based on I²C-bus specification

Bus Protocol

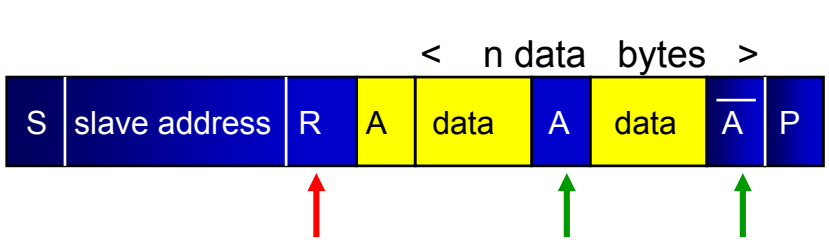
Bus Protocol Basics

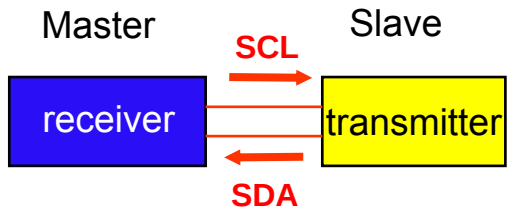
- START bit – sent by master for initiating a communication
- ADDRESS bits – sent by master to talk to the slave device
- R/W bit – sent by master to indicate a read or write operation
- ACK bit (acknowledge) – responded by the receiving device
- DATA (byte wise) – sent by transmitting or receiving device
- NACK (no acknowledge) – sent by master or slave prior to ending communication
- STOP bit – sent by master to end a communication

  or RE-START bit (equivalent to START without a STOP)

Single Read/Write Operation

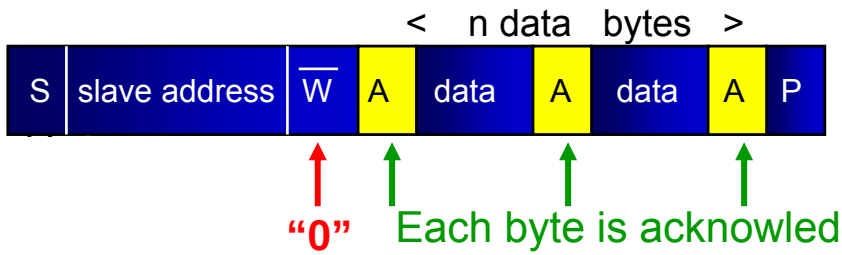
- Read from a Slave device
  - The master is a “MASTER TRANSMITTER then MASTER - RECEIVER”: It sends slave address data and then becomes a receiver

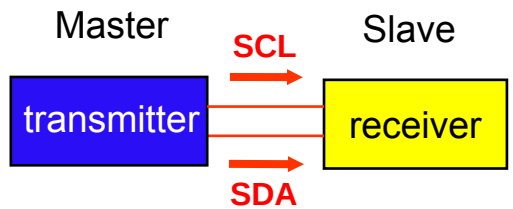




Each byte is acknowledged by the master device (except the last one, just before the STOP condition)

- Write to a Slave device
  - Master transmits both clock and data

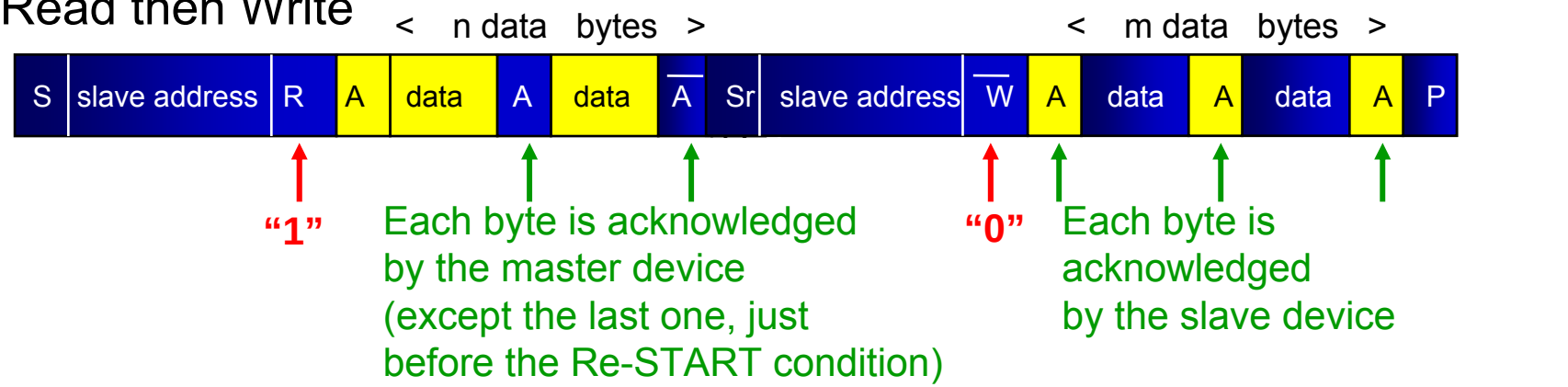




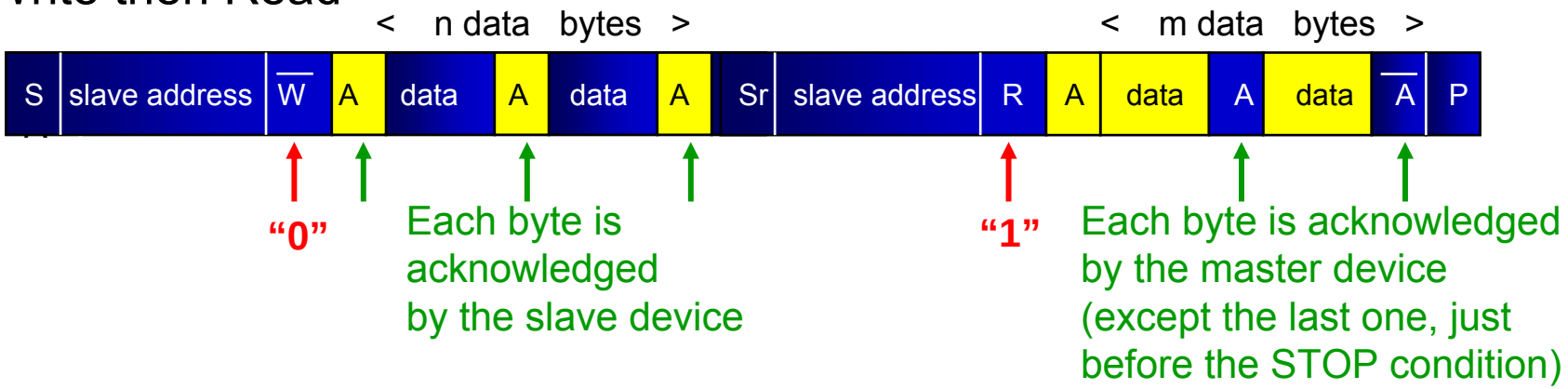
Each byte is acknowledged by the slave device

Combined Read/Write Operation

- Read then Write

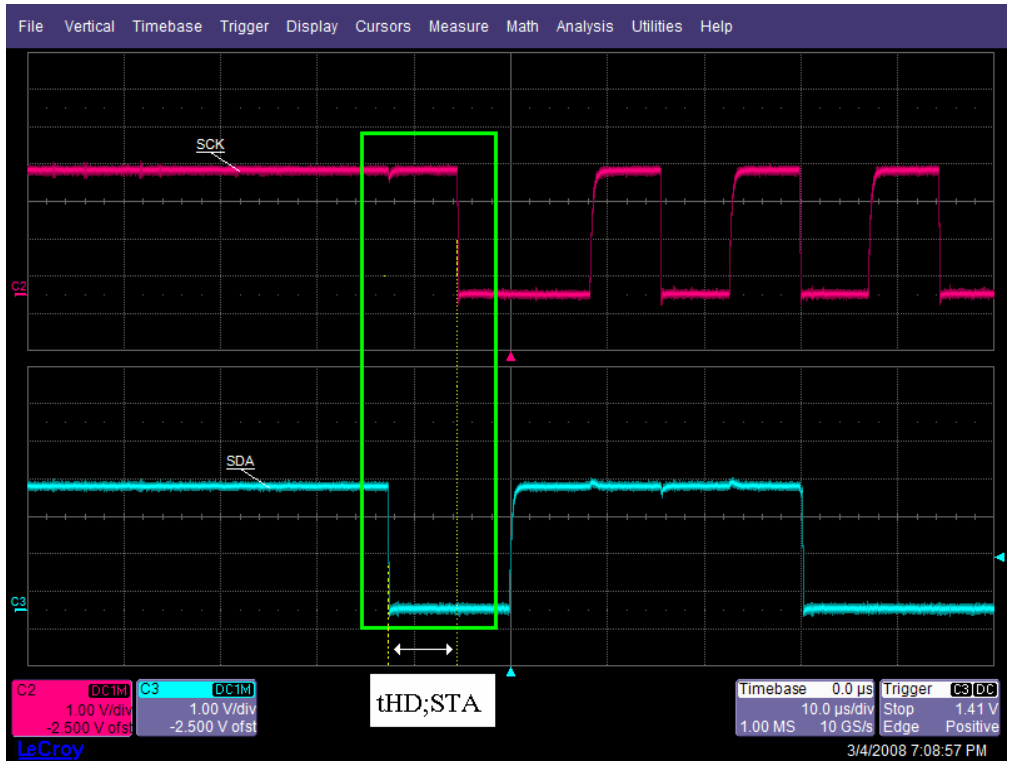


- Write then Read



START Timing Diagram

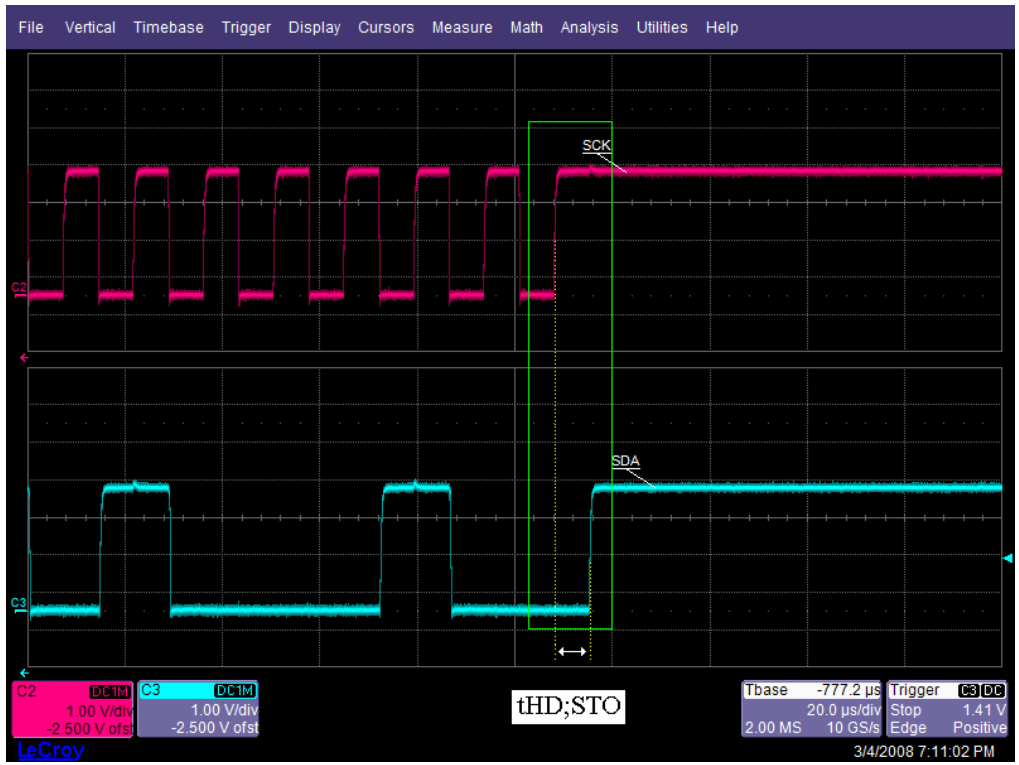
- START condition is a high to low transition on the SDA line while SCL is high



START	Slave Address	R/$\overline{W}$	ACK	DATA	ACK	… DATA	NACK	RE-START/ STOP

STOP Timing Diagram

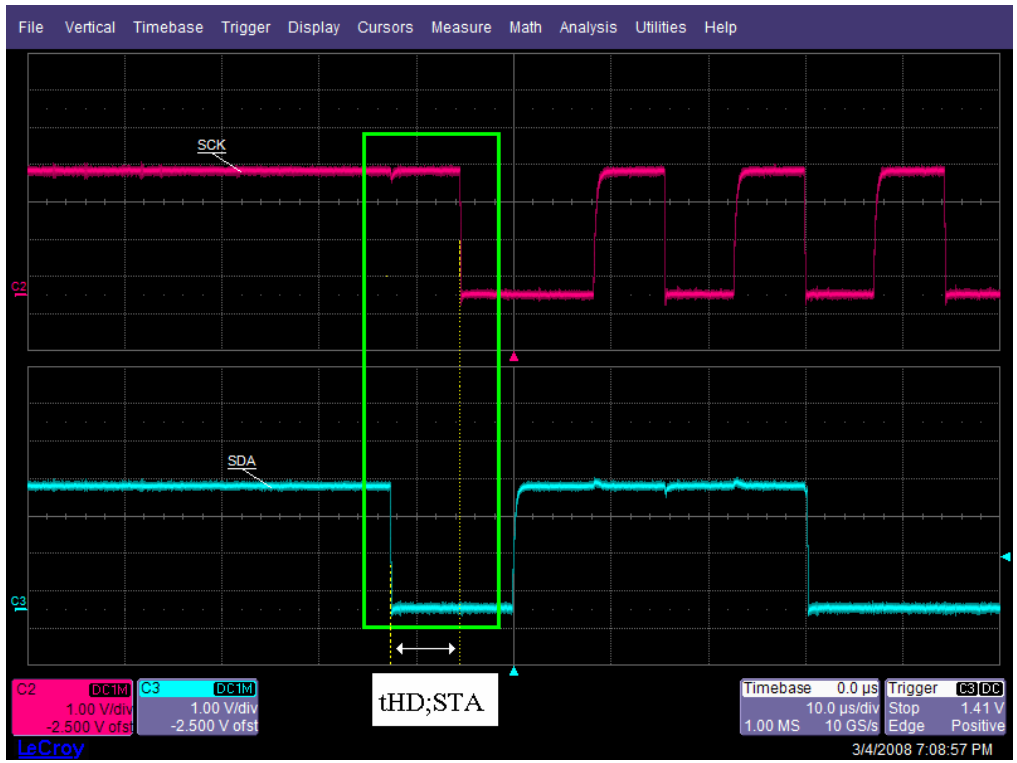
- STOP condition is a low to high transition on the SDA line while SCL is high



START	Slave Address	R/$\overline{W}$	ACK	DATA	ACK	… DATA	NACK	RE-START/ STOP

RE-START Timing Diagram

- RESTART condition is a high to low transition on the SDA line while SCL is high, exactly the same as the START



START	Slave Address	R/$\overline{W}$	ACK	DATA	ACK	… DATA	NACK	RE-START/ STOP

I²C Slave Address

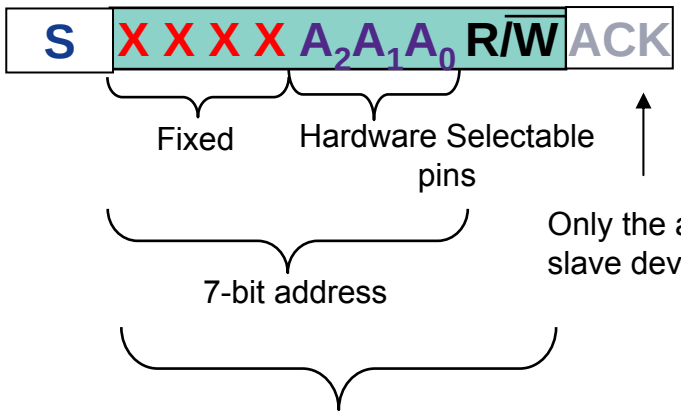
- Two formats of I²C slave address: 7-bit or 10-bit address
- 7-bit Slave address is most popular, allows up 111(2^7 = 128 devices →128 – 17 reserved)
- 10-bit slave address may accommodate up to 2^{10} = 1024 devices on the same bus
- Devices that supports either 7-bit or 10-bit address, may co-exist on the same bus

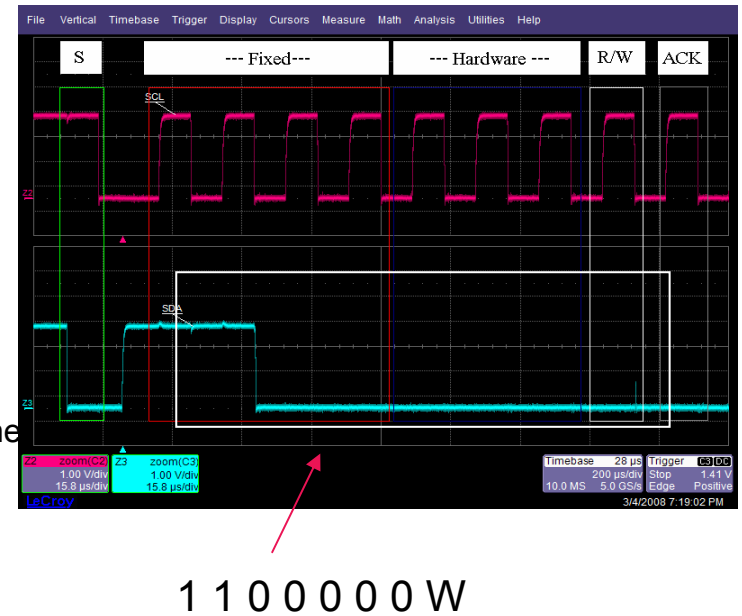
Reserved Addresses

Slave address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	CBUS address[3]
0000 010	X	reserved for different bus format[4]
0000 011	X	reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	reserved for future purposes
1111 0XX	X	10-bit slave addressing

I²C Slave Address – 7bit Format

- ✓ Consist of fixed address and hardware selectable address (optional), a total of 7-bit
- ✓ R/W bit is sometimes included as part of the address, and is called read or write address)

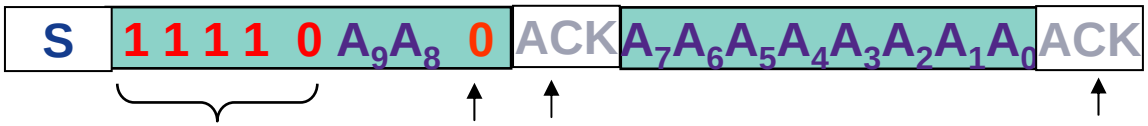




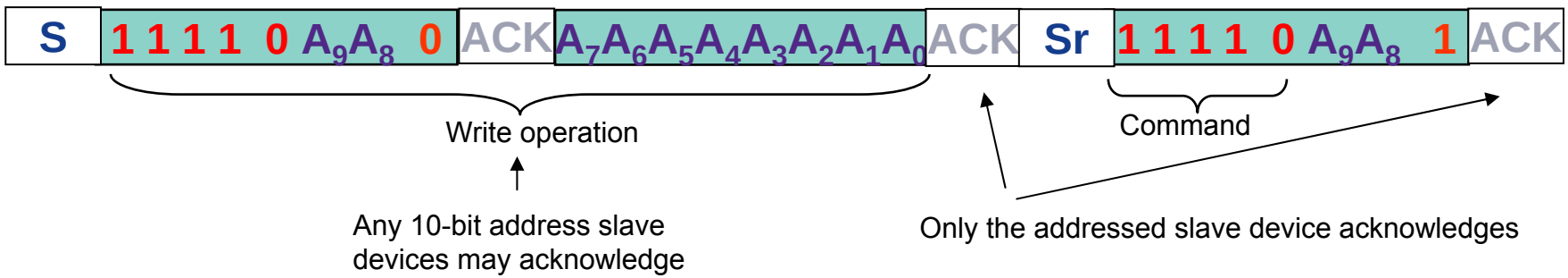
START	Slave Address	R/W	ACK	DATA	ACK	… DATA	ACK/ NACK	RE-START/ STOP

I²C Slave Address – 10bit Format

- ✓ Consist of fixed command, 11110 and 10-bit address (fixed or hardware selectable)
- ✓A **write operation** takes up two bytes

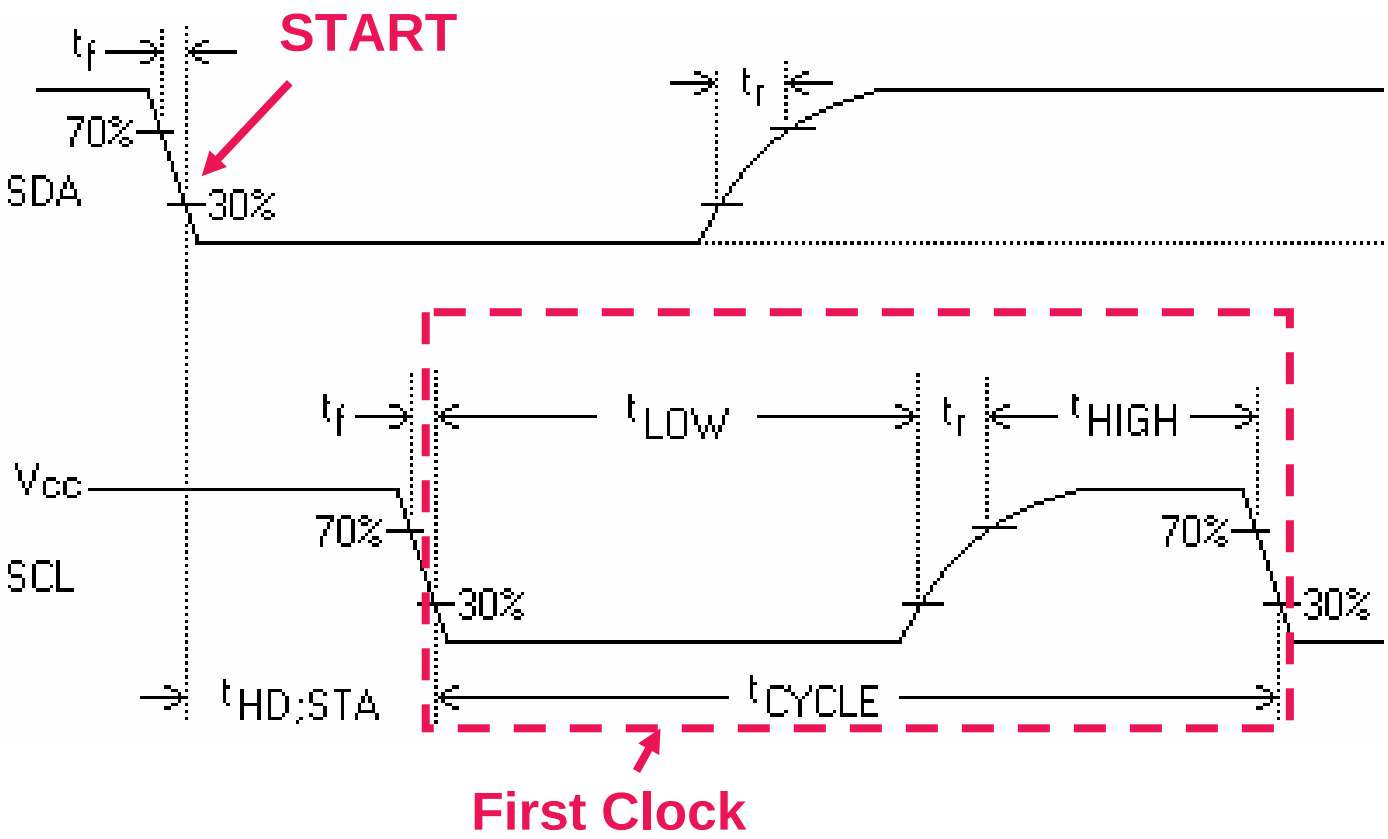


- ✓ A **read operation** requires a two byte for write, followed by a 1 byte read



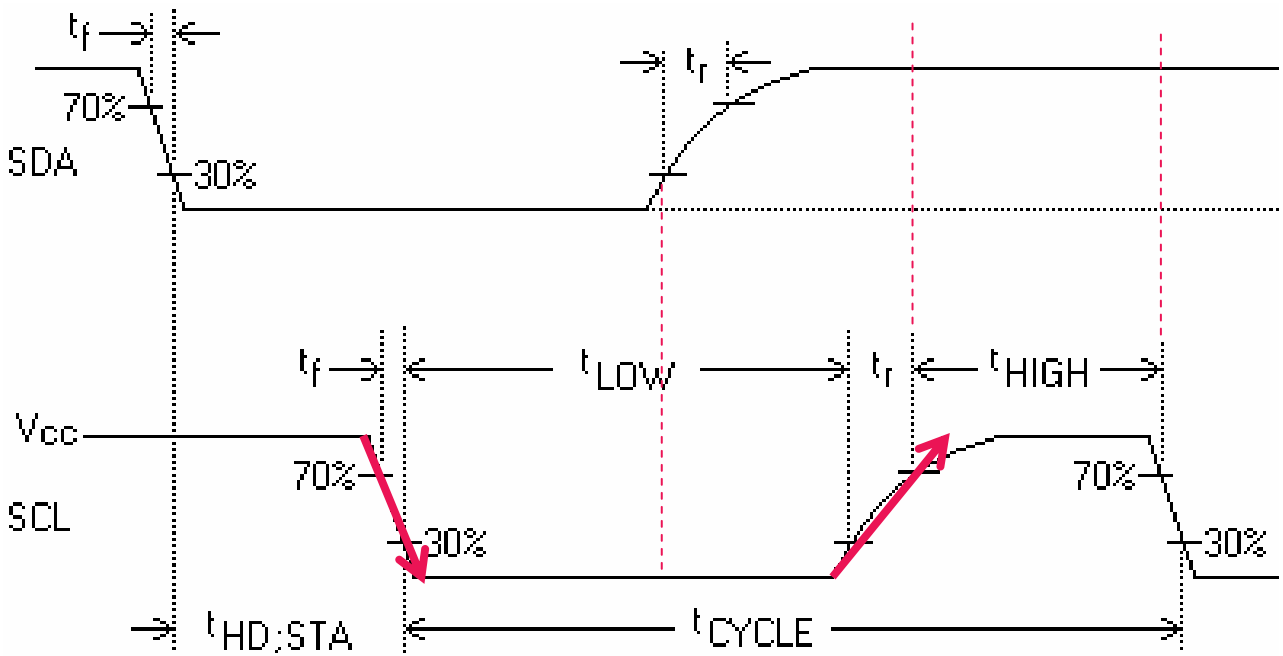
1st Clock Timing Diagram

- First clock cycle is clock cycle after the START bit



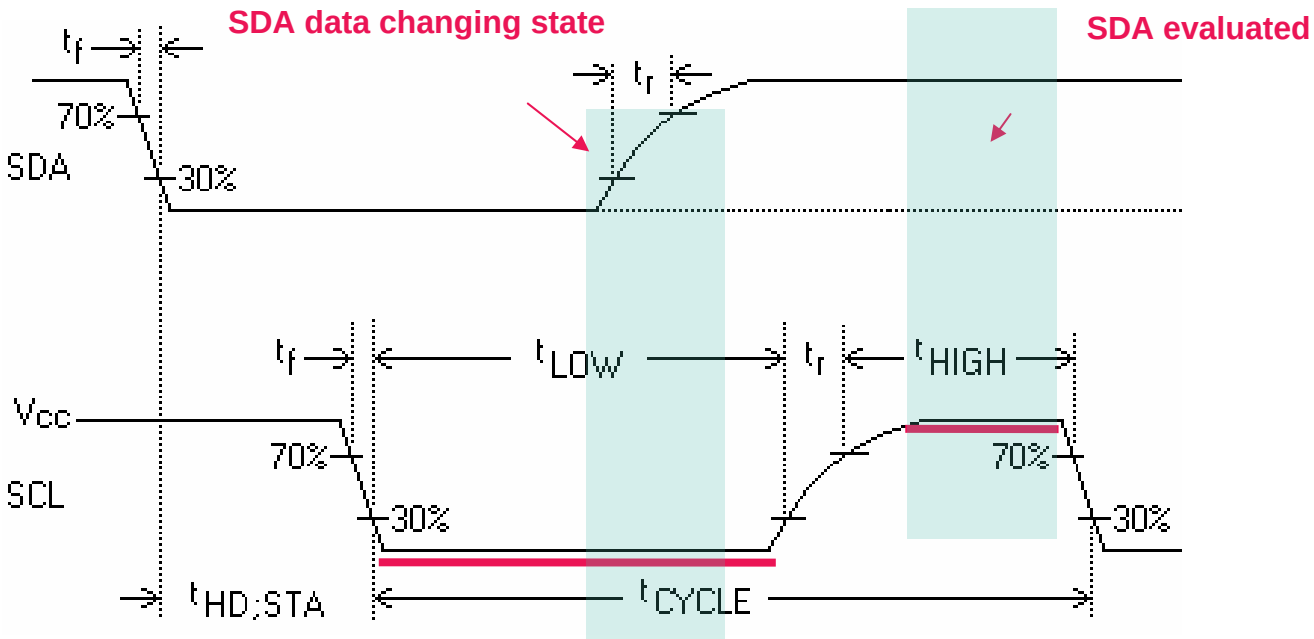
SCL, Serial Clock Timing Diagram

- **Falling edge** of SCL signals the requirement to provide data on SDA
- **Rising edge** of SCL signals the need to hold that data bit
- **Clock period** (T) = $t_{CYCLE} = 1/f_{SCL} = t_{LOW} + t_{HIGH} + t_f + t_r$



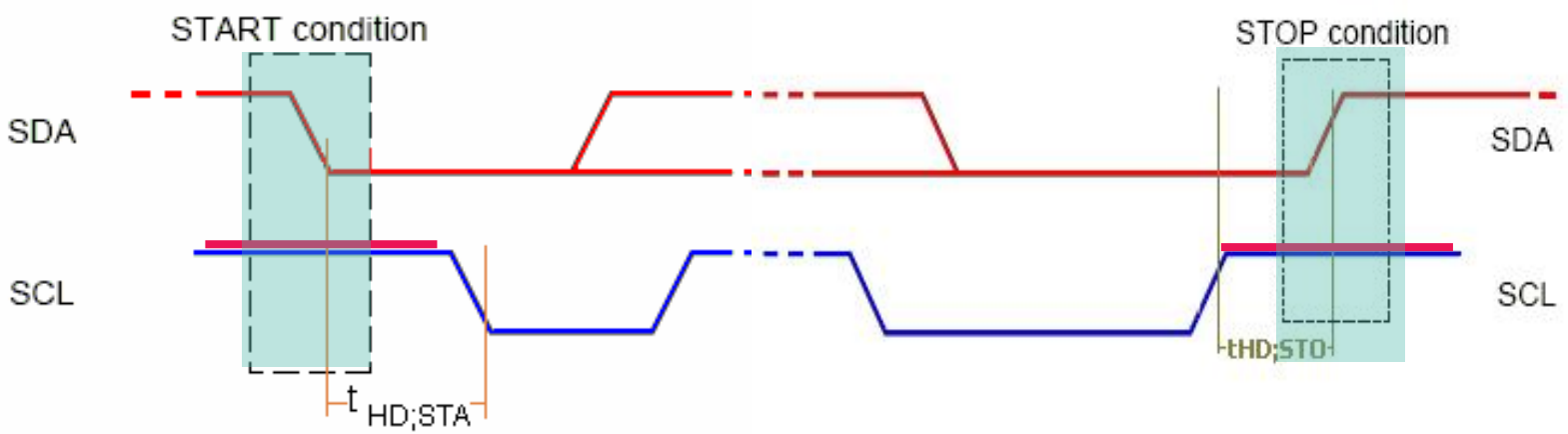
SDA, Serial Data Timing Diagram

- Normal data transition occurs when SCL is low
- Data is evaluated when SCL is high



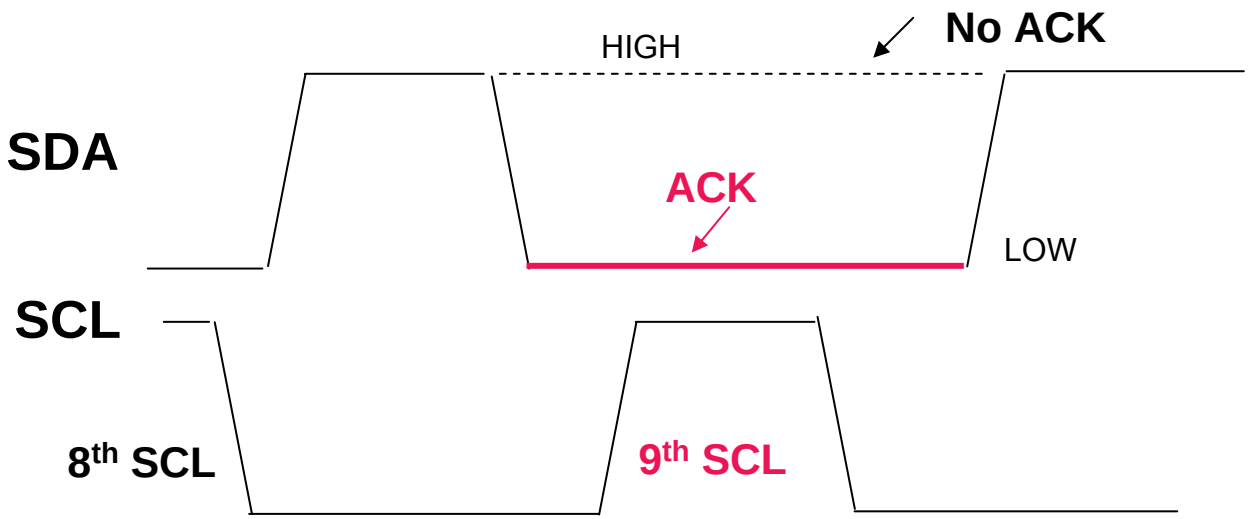
SDA, Serial Data Timing Diagram

- Data transitions when SCL is high, signifies special conditions such that
  - High-to-Low transition is a START or Re-START condition
  - Low-to-High transition is a STOP condition



Acknowledge (ACK)

- Acknowledge bit occurs on the 9th SCL clock pulse
- The transmitter and receiver behave as follows:
  1. Transmitter releases SDA line after the 8th clock pulse
  2. Receiver **acknowledges** by pulling SDA low on the 9th clock pulse
  3. Transfer is aborted if SDA does not go low (no ACK)

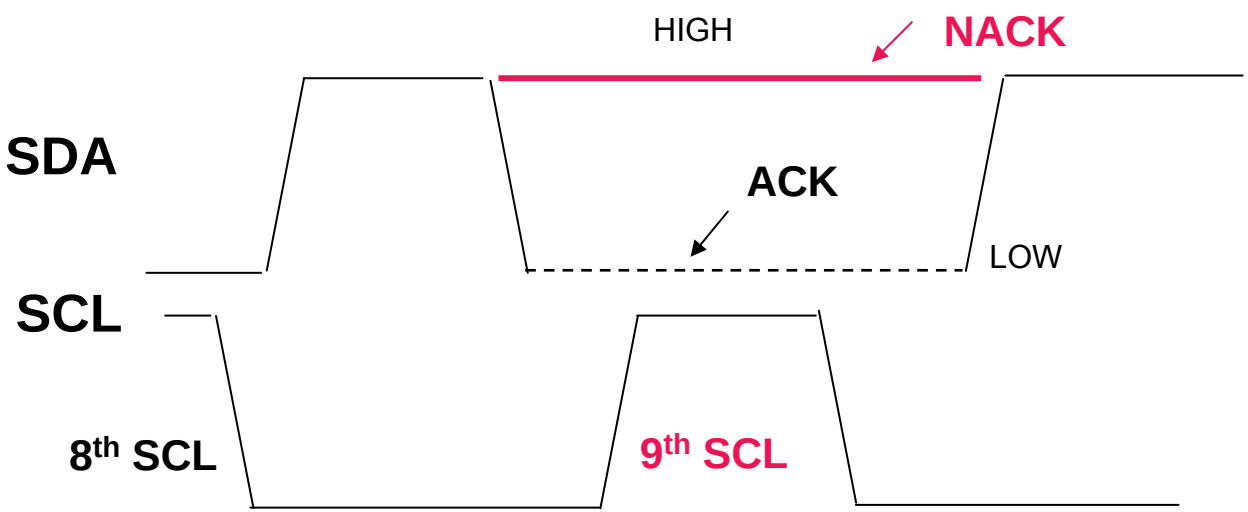


Acknowledge Example

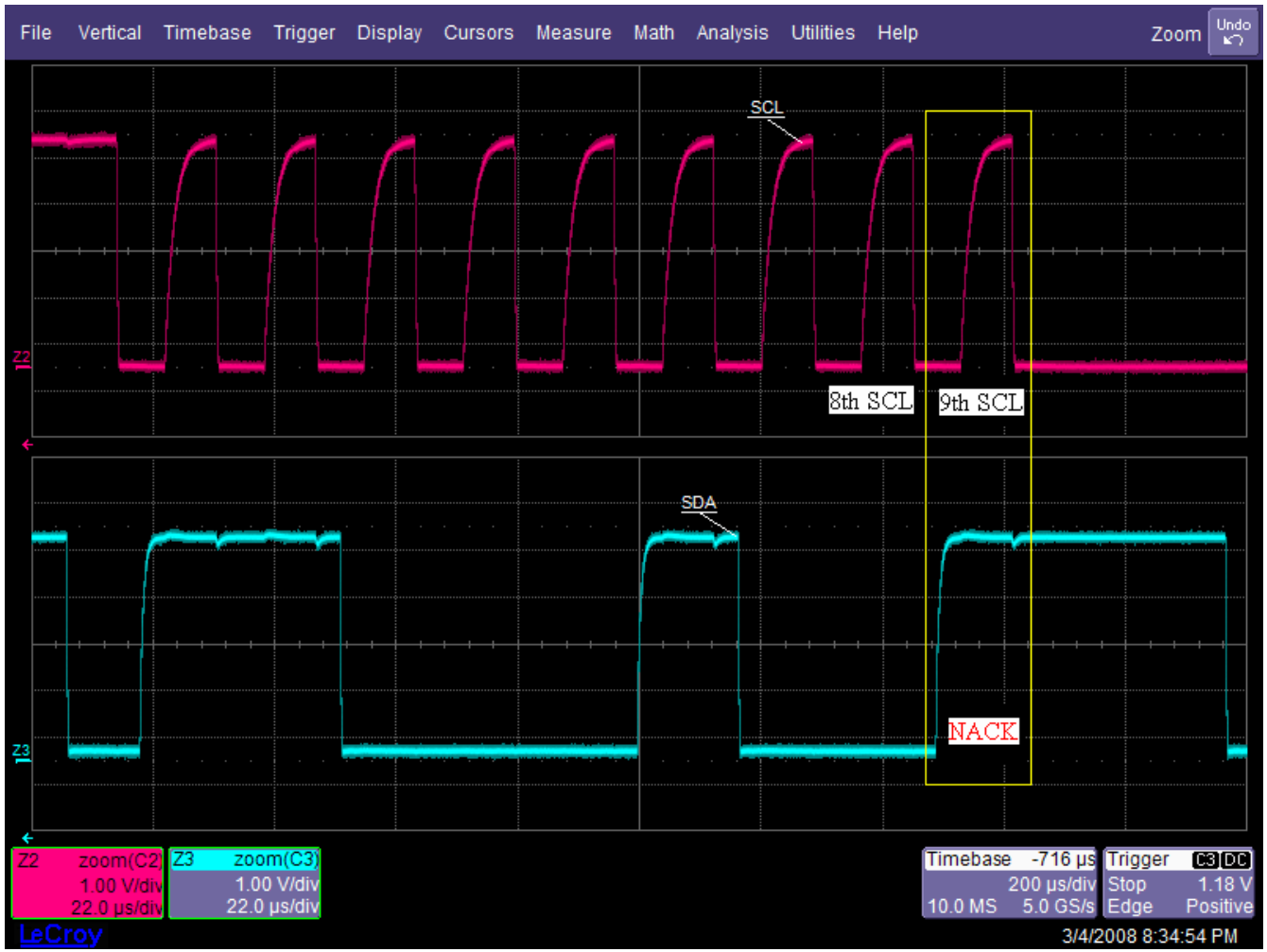
No Acknowledge (NACK)

Three Scenarios where there is no acknowledge taking place

1. The receiver being addressed is not present on the I^2C bus
2. The receiver is busy (cannot process the receiving information)
3. The master receiver wants to take control of SDA line

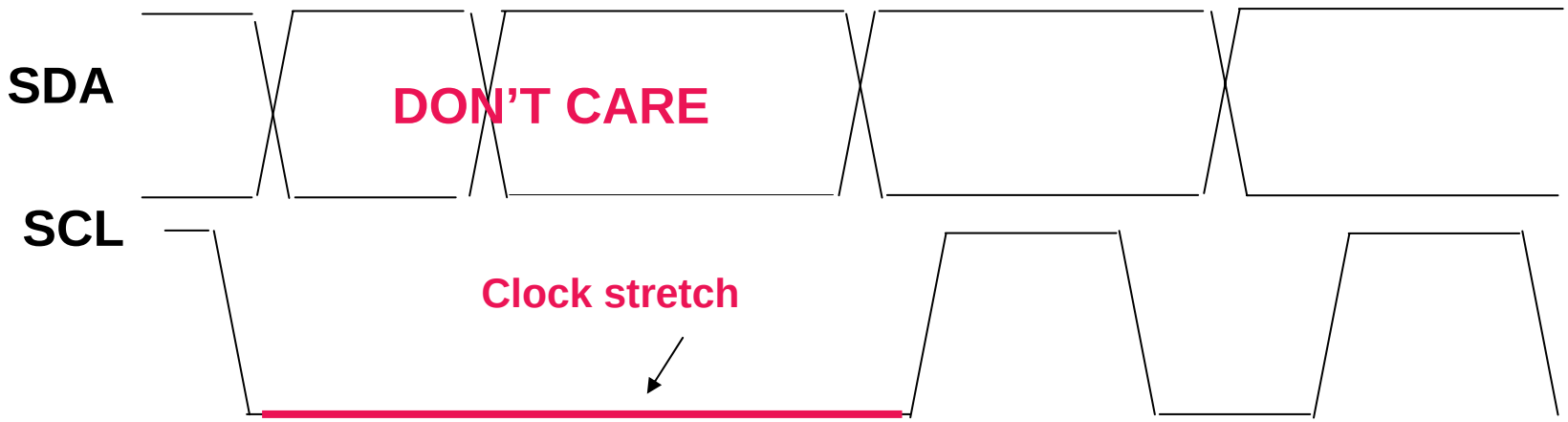


No Acknowledge Example



Clock Stretch

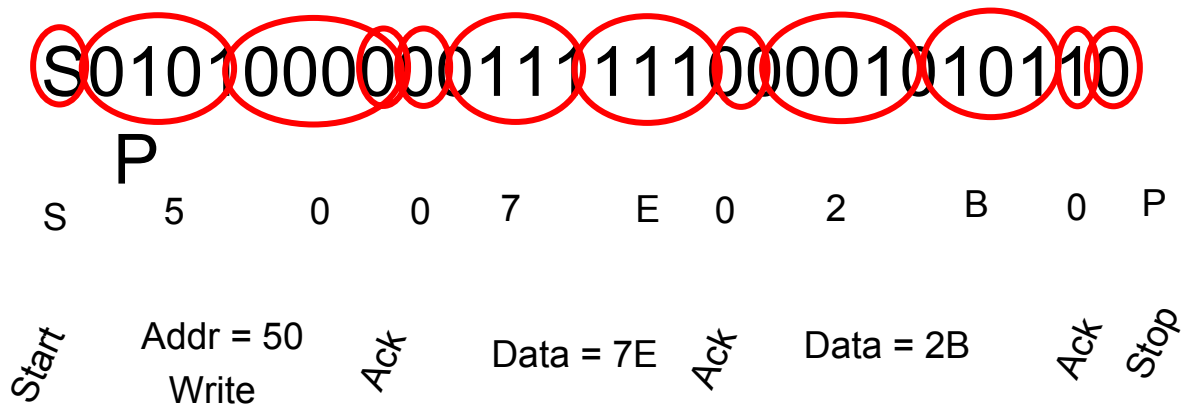
- A clock stretch is defined as the SCL line being pulled low – stretched – longer than its intended clock low period
- Why a device stretches the clock?
  - A master performs clock stretch when it needs to slow down the clock in order to accommodate a slower slave device
  - A slave performs clock stretch when it needs to perform other function
- Data is “DON’T CARE” when the clock is low



Debugging I²C signals on an Oscilloscope

I²C Messages

- Many engineers manually count bits on the screen and jot down 1's and 0's in log books then convert binary to hex on paper



I²C Messages Decoded

- Engineers can use decoding package in a digital oscilloscopes to decode long data packets

I²C Decoding

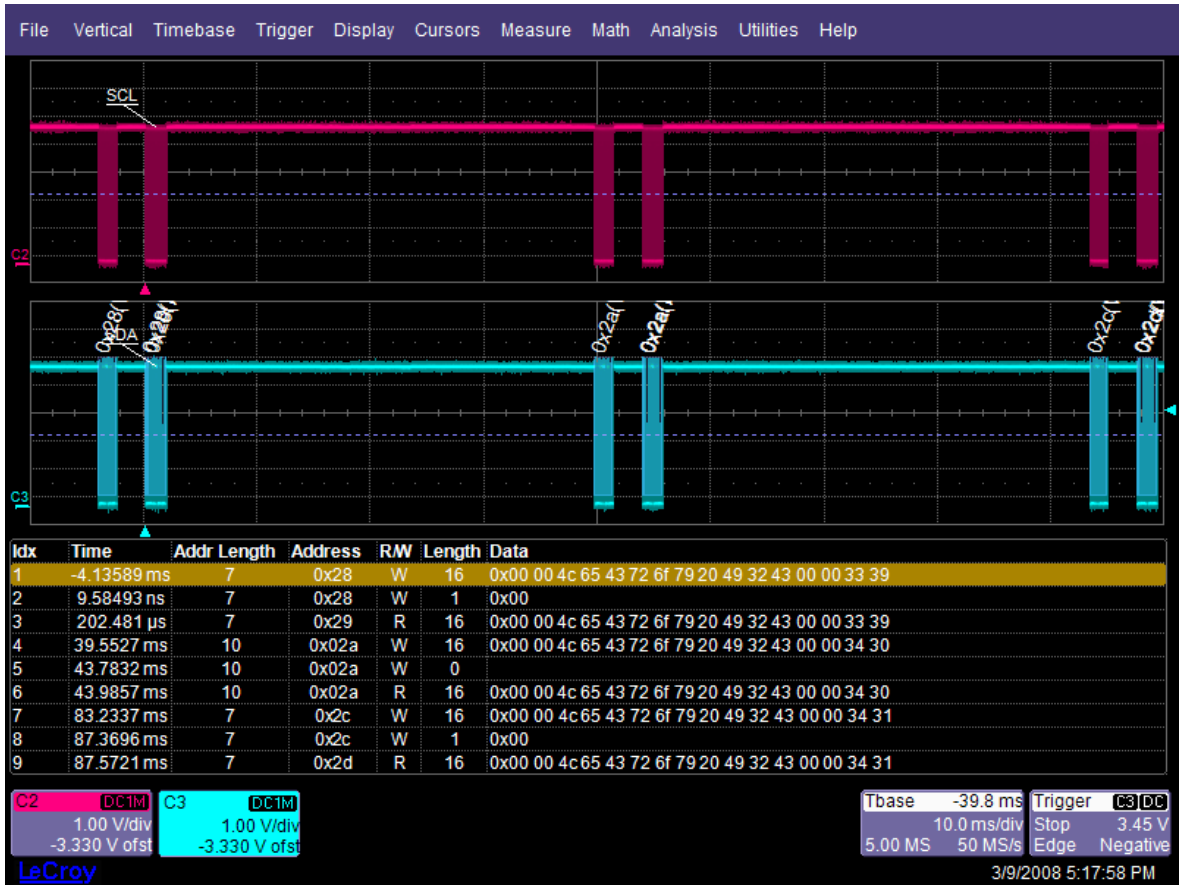
- Most commonly used decoding is hex decoding

How to decode multiple packets

- Use long memory of oscilloscopes for longer capture time

How to decode multiple packets

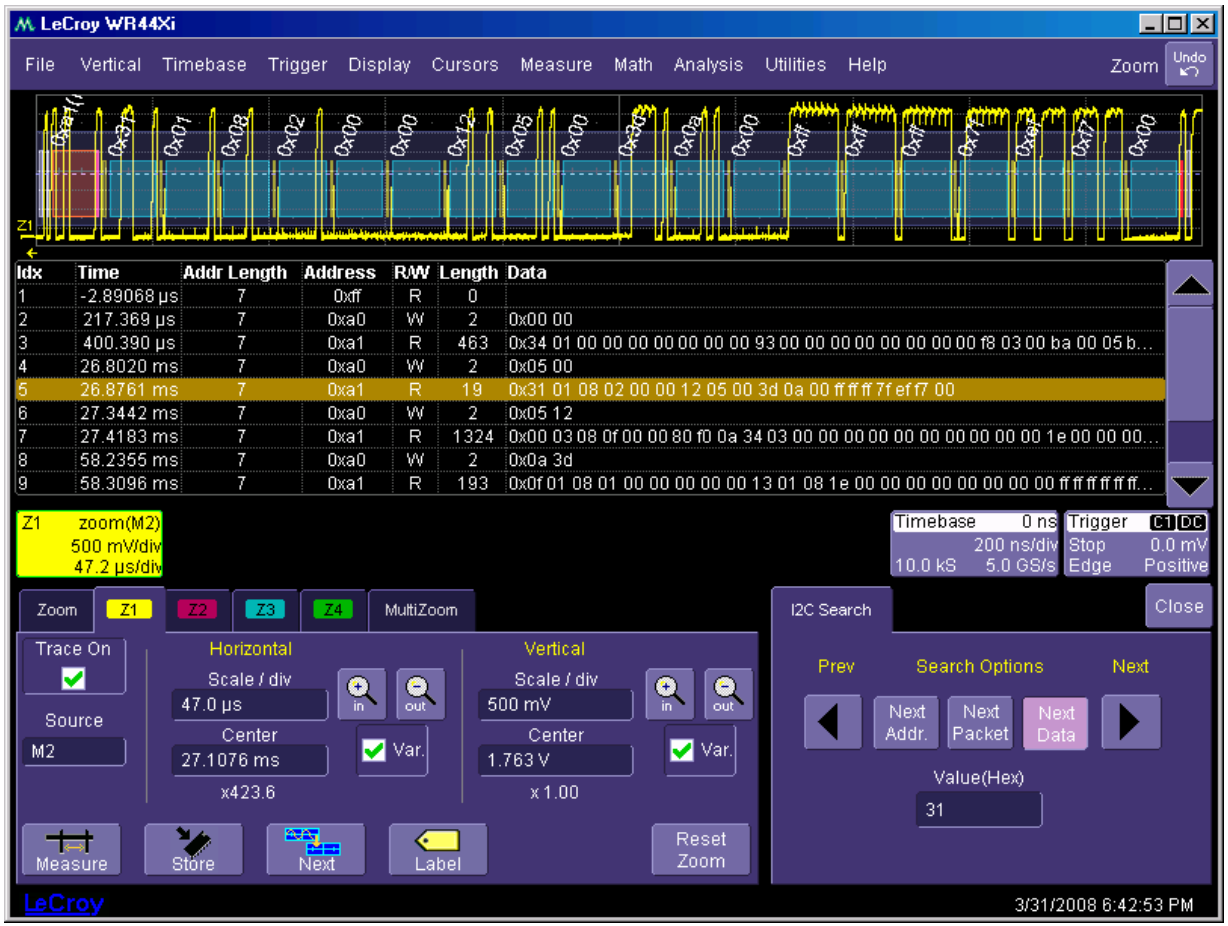
- Use decoding table to view contents of several data packets
- View acquisition time, address, R/W and data length



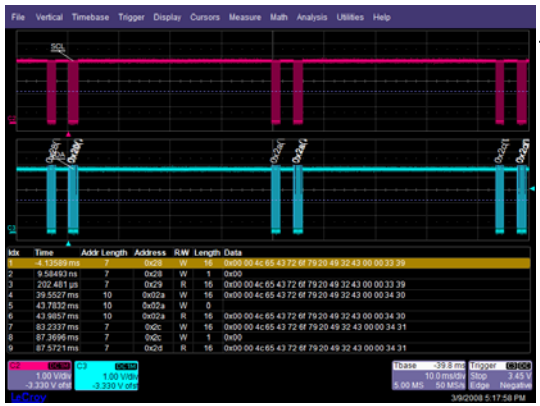
Idx	Time	Addr Length	Address	R/W	Length	Data
1	-4.13589 ms	7	0x28	W	16	0x00 00 4c 65 43 72 6f 79 20 49 32 43 00 00 33 39
2	9.58493 ns	7	0x28	W	1	0x00
3	202.481 µs	7	0x29	R	16	0x00 00 4c 65 43 72 6f 79 20 49 32 43 00 00 33 39
4	39.5527 ms	10	0x02a	W	16	0x00 00 4c 65 43 72 6f 79 20 49 32 43 00 00 34 30
5	43.7832 ms	10	0x02a	W	0	
6	43.9857 ms	10	0x02a	R	16	0x00 00 4c 65 43 72 6f 79 20 49 32 43 00 00 34 30
7	83.2337 ms	7	0x2c	W	16	0x00 00 4c 65 43 72 6f 79 20 49 32 43 00 00 34 31
8	87.3696 ms	7	0x2c	W	1	0x00
9	87.5721 ms	7	0x2d	R	16	0x00 00 4c 65 43 72 6f 79 20 49 32 43 00 00 34 31

Search feature…

- Use search feature to quickly find address or data packet of interest



Using export table



- Transfer captured data to Excel table for further analysis

Microsoft Excel - DecodeTable.csv

File Edit View Insert Format Tools Data Window Help

Type a question for help

Arial 10

F17

	A	B	C	D	E	F
1	Time	Addr Length	Address	R/W	Length	Data
2	-4.13589 ms	7	40	87	16	0 0 76 101 67 114 111 121 32 73 50 67 0 0 51 57
3	9.58493 ns	7	40	87	1	0
4	202.481 µs	7	41	82	16	0 0 76 101 67 114 111 121 32 73 50 67 0 0 51 57
5	39.5527 ms	10	42	87	16	0 0 76 101 67 114 111 121 32 73 50 67 0 0 52 48
6	43.7832 ms	10	42	87	0	
7	43.9857 ms	10	42	82	16	0 0 76 101 67 114 111 121 32 73 50 67 0 0 52 48
8	83.2337 ms	7	44	87	16	0 0 76 101 67 114 111 121 32 73 50 67 0 0 52 49
9	87.3696 ms	7	44	87	1	0
10	87.5721 ms	7	45	82	16	0 0 76 101 67 114 111 121 32 73 50 67 0 0 52 49
11	Bitrate	100578				
12	DecodeMode	Hex				
13	Level	2.553835				
14	Protocol	I2C				
15						

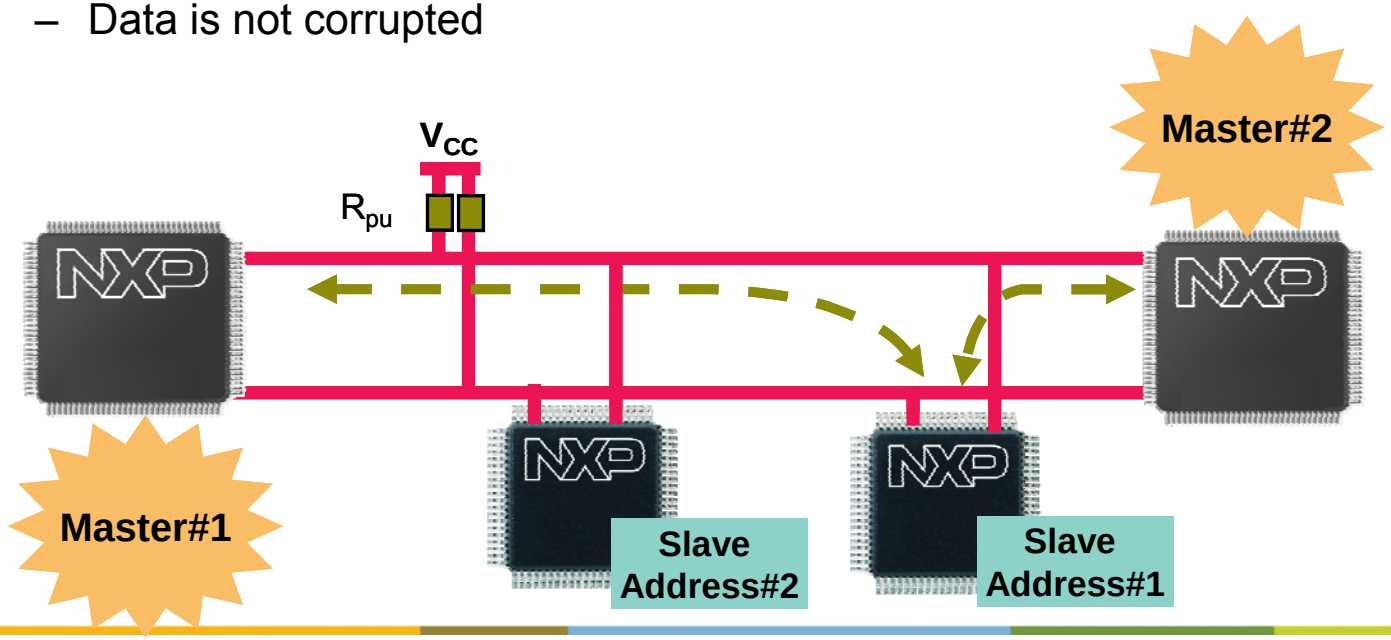
DecodeTable

Ready

Multi-master Synchronization/Arbitration

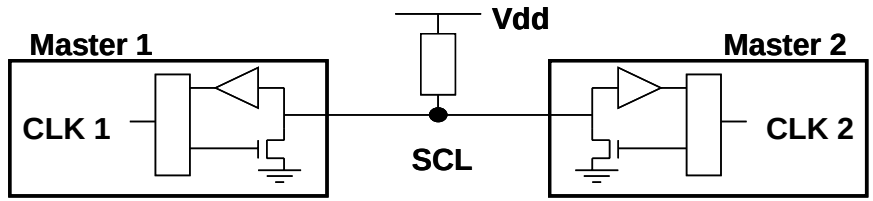
Clock Synchronization & Bus Arbitration

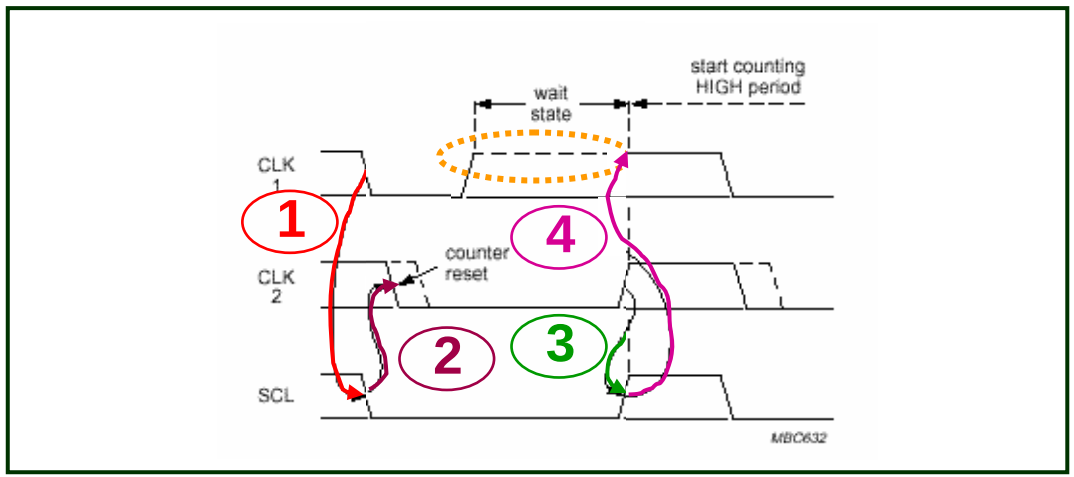
- Multi-master environment requires clock synchronization and data arbitration
- Masters synchronize on clock line
  - Result in new clock frequency = longest low + shortest high
- Masters arbitrate on data line
  - Result in the losing master stop sending data
  - Losing master can continue to send SCL until the end the byte
  - Data is not corrupted



Clock Synchronization Timing Diagram

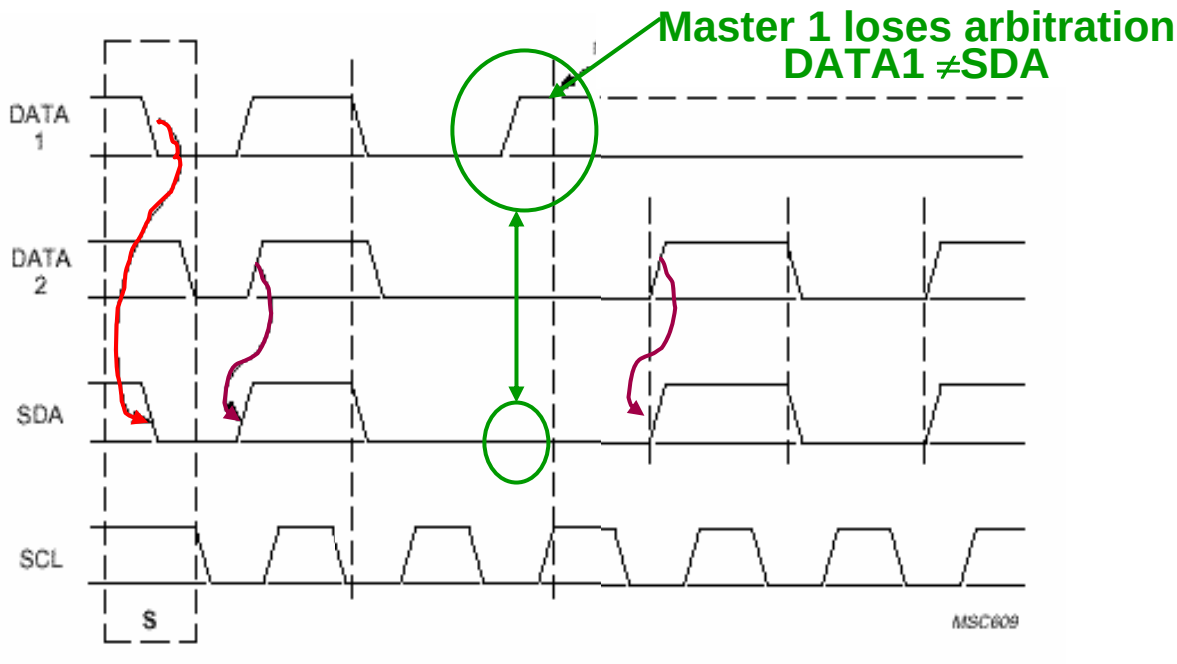
- **LOW** period determined by the **longest clock low** period
- **HIGH** period determined by **shortest clock high** period





Arbitration Timing Diagram

- Two or more masters may generate a START condition at the same time
- Arbitration is done on SDA while SCL is high - Slaves are not involved



Bus Recovery Method

Problem:

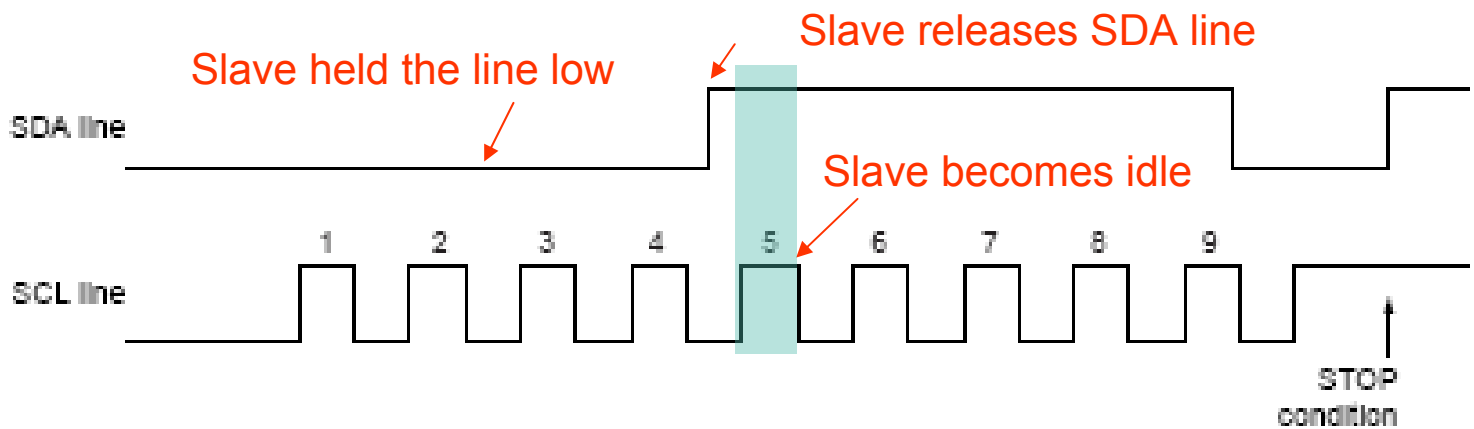
- Typical case is when a master fails during reading the slave device
- The SDA line is stuck low because the slave is stuck in the transmitter mode

Solution

- Three methods to recover the bus:
  1) Use a hardware reset pin to reset the slave device (assuming the device has a reset pin)
  2) Use a hardware self-timeout function such as SMBus timeout
  3) Use an I^2C-bus recovery sequence to leave the “Slave-Transmitter” mode

I²C-bus Recovery Sequence

- An I²C-bus recovery sequence is done as follows:
  1) Send 9 clock pulses on SCL line while the master “keeps” the SDA line high until the “Slave-Transmitter” releases it including the time for the ACK operation. “Keeping” SDA high during the ACK means that the “Master-Receiver” does not acknowledge the previous received byte
  2) The “Slave-Transmitter” then goes in an idle state (mandatory when no acknowledge is received)
  3) The master then sends a STOP command to initialize the bus



Fast-mode Plus – released June 2007

- Develop to target high bandwidth or long cable system management for
  - Future proof faster I²C-bus
  - Gaming application – long cable or heavy loading
  - Fun lighting application – long cable or heavy loading
- Support bus frequency up to 1 MHz, fully compatible with existing I²C and SMBus
- Support 10 times stronger drive than normal I²C drivers

Specification – UM10204, downloadable from www.nxp.com/i2c

Fast-mode Plus – Additional Features

Additional features

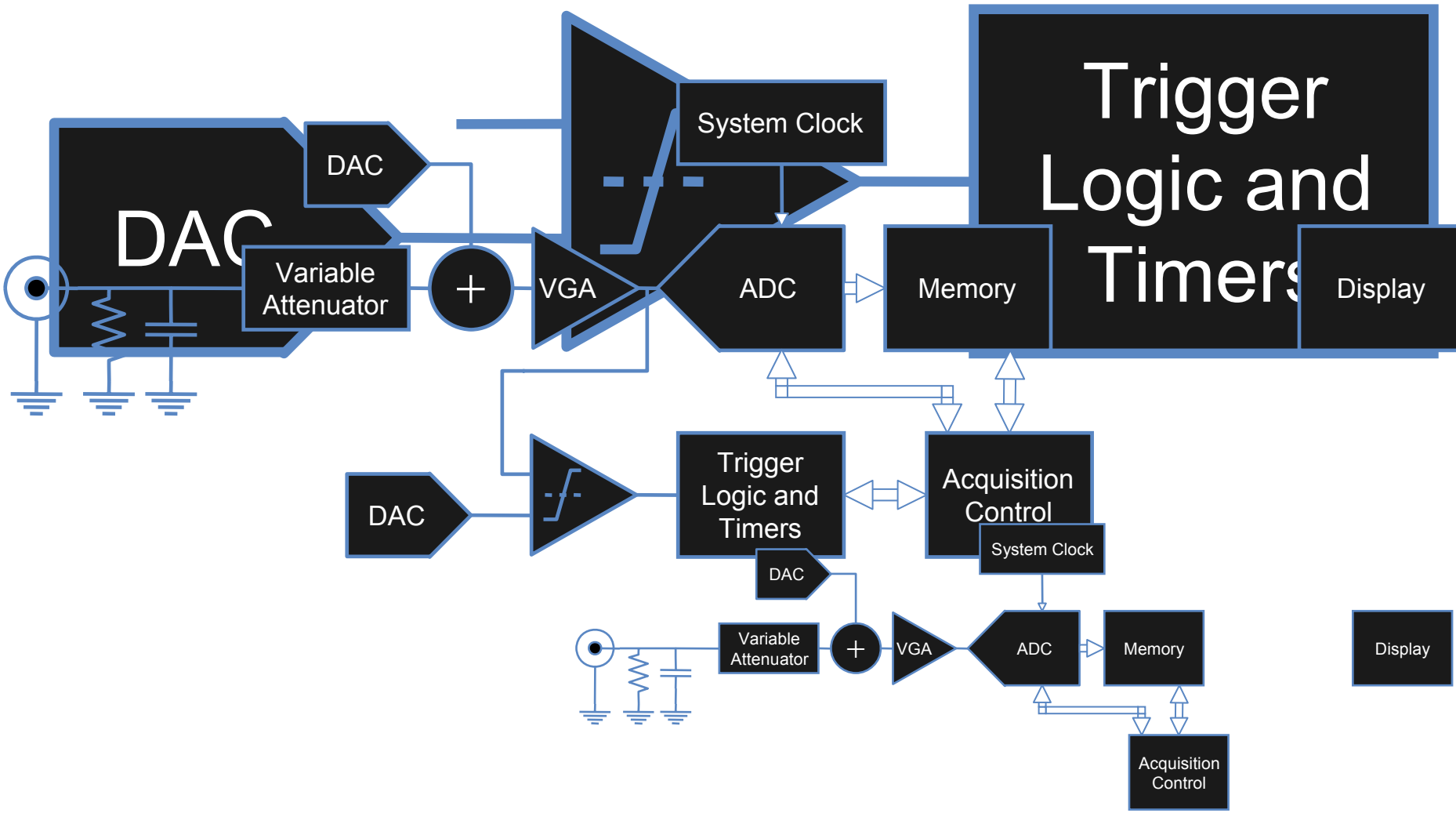
- Integrate a software reset command (POR) by I^2C sequence
- Support manufacturers I.D.
- Support hardware address pins connecting to V_{CC}, GND SCL, SDA making it possible to have up to 64 addresses with only three pins versus normally 8

Agenda Cont'd

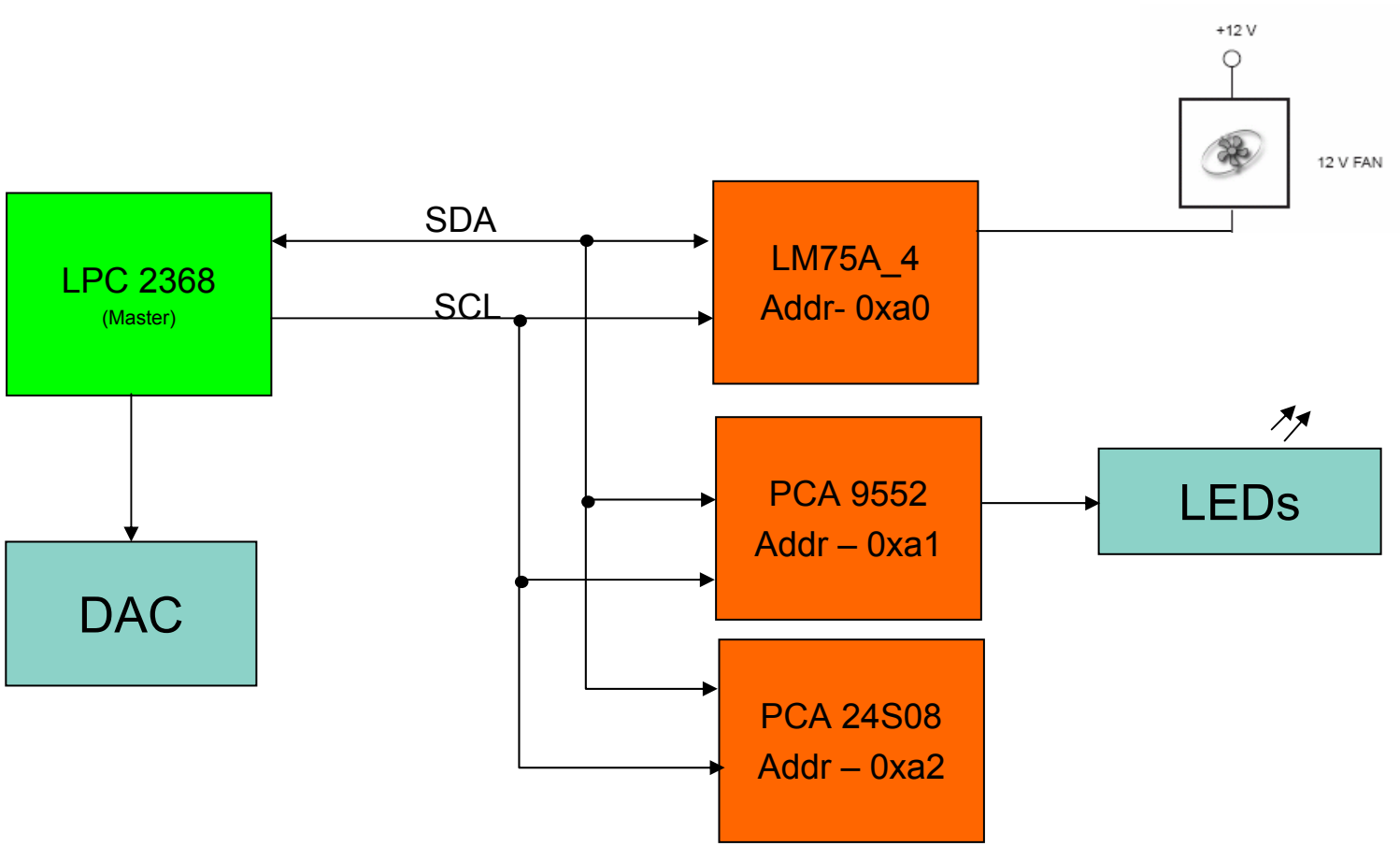
- I²C-bus Introduction
- Bus Protocol
- I²C Protocol Decoding Technique – using an Oscilloscope
- Multi-master Synchronization & Arbitration
- Bus recovery
- Fast-mode Plus
- Oscilloscope Scope triggering to debug embedded circuits
- Electrical Characteristics
- Using Oscilloscope to make meaningful measurement
- Application
- Useful links for I²C products and support
- I²C-bus Summary
- Q&A

Using I²C Trigger on Oscilloscope

How does the scope find a trigger event?



Debugging embedded circuit …



I²C Start Trigger

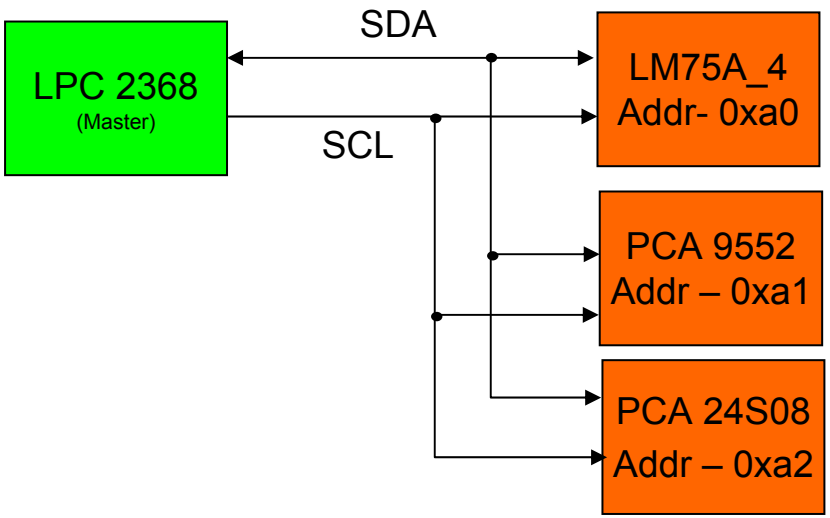
- Allows user to debug messages based on the protocol

I²C Stop Trigger

- Allows user to debug messages based on the protocol

I²C Address Trigger

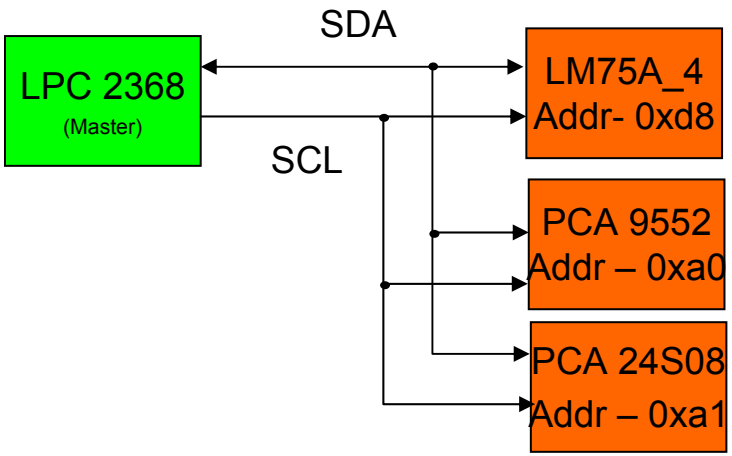
- Trigger to monitor data exchange on system with multiple Master/Slave configuration
- Enables user to probe the common I²C bus and monitor traffic for a specific device on the bus

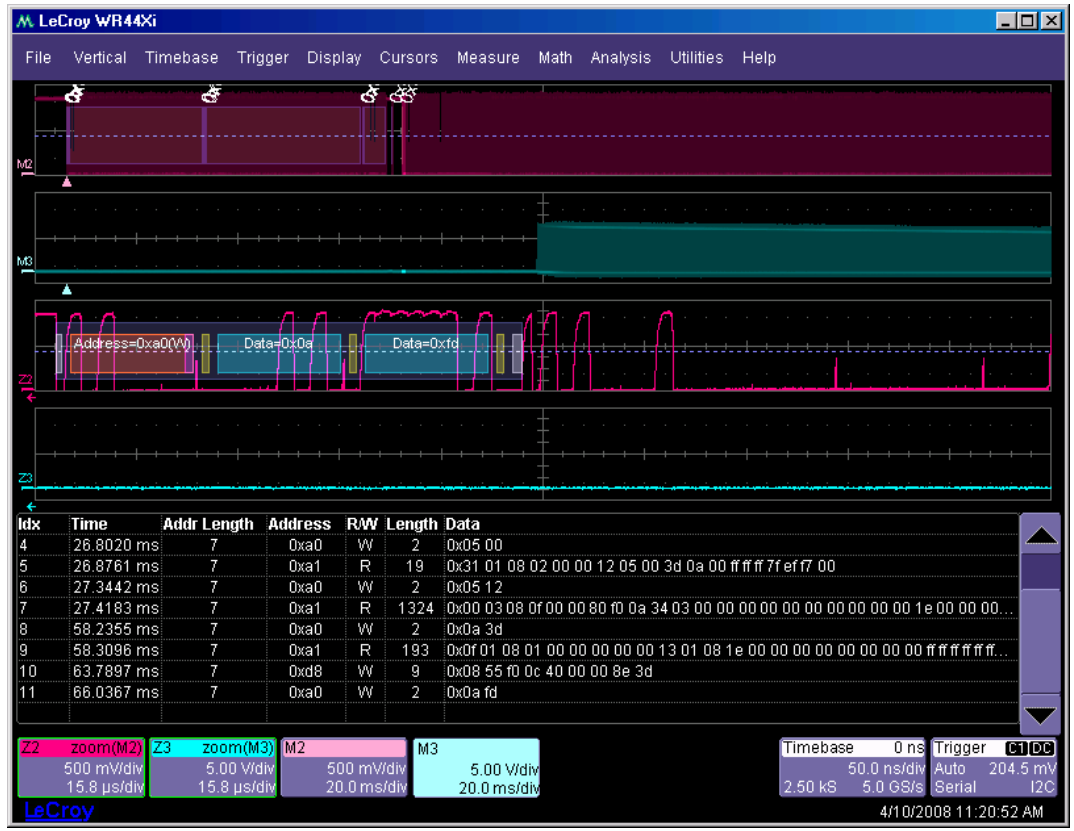


I²C Nack Trigger

- Useful for looking at non-functional devices
- Generally helpful, when combined with Address Trigger to check for intermittent failures

Frame Length





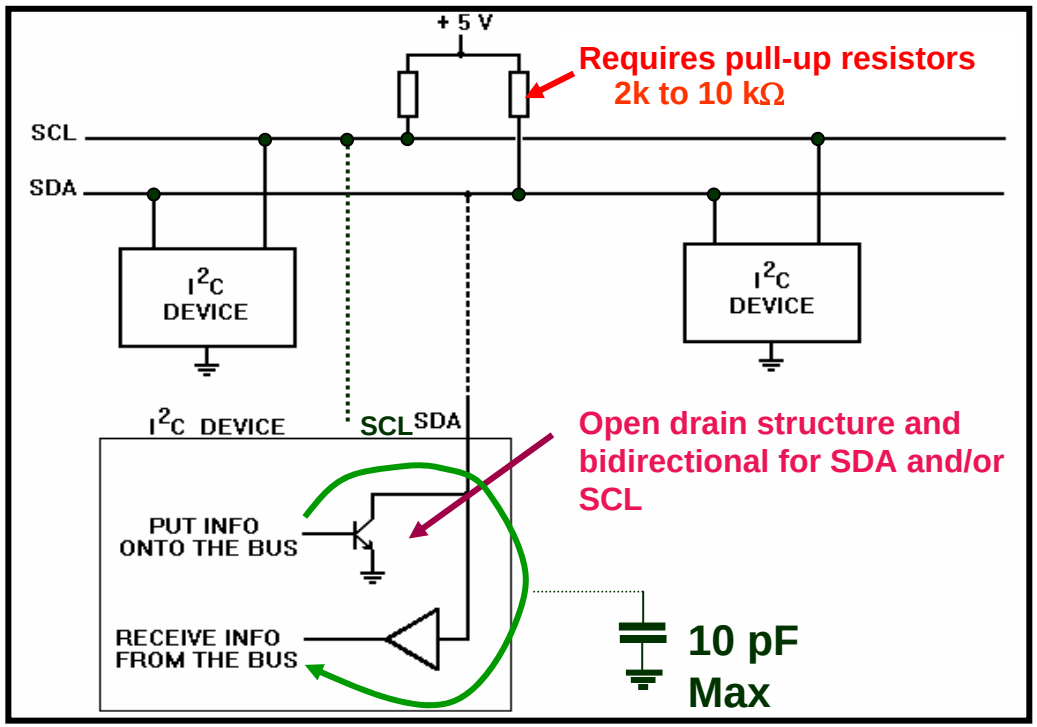
I²C Protocol Summary

START	High to low transition on SDA while SCL is high
STOP	Low to high transition on SDA while SCL is high
DATA	8-bit word, MSB first (Address, Control, Data) • Must be stable when SCL is high • Change only when SCL is low • Number of bytes transmitted is unrestricted
ACKNOWLEDGE	• Performed on each 9th clock pulse during SCL high • Transmitter releases the bus – SDA high • Receiver pulls down the bus – SDA low
CLOCK	• Generated by the master • Maximum speed specified – no minimum speed • A receiver may hold SCL low when it is busy (transmitter wait) • A master can stretch the clock for slow devices
CLOCK SYNCHRONIZATION	• Automatically performed by masters in a multi-masters environment • Resulting clock is longest low and the shortest high
ARBITRATION	• Automatically performed by masters in a multi-masters environment • Master can start a transfer only when the bus is free • Several masters can start transfer at the same time • Arbitration is performed on SDA line • Master loses arbitration must stop transmitting

I²C-bus Electrical Characteristics

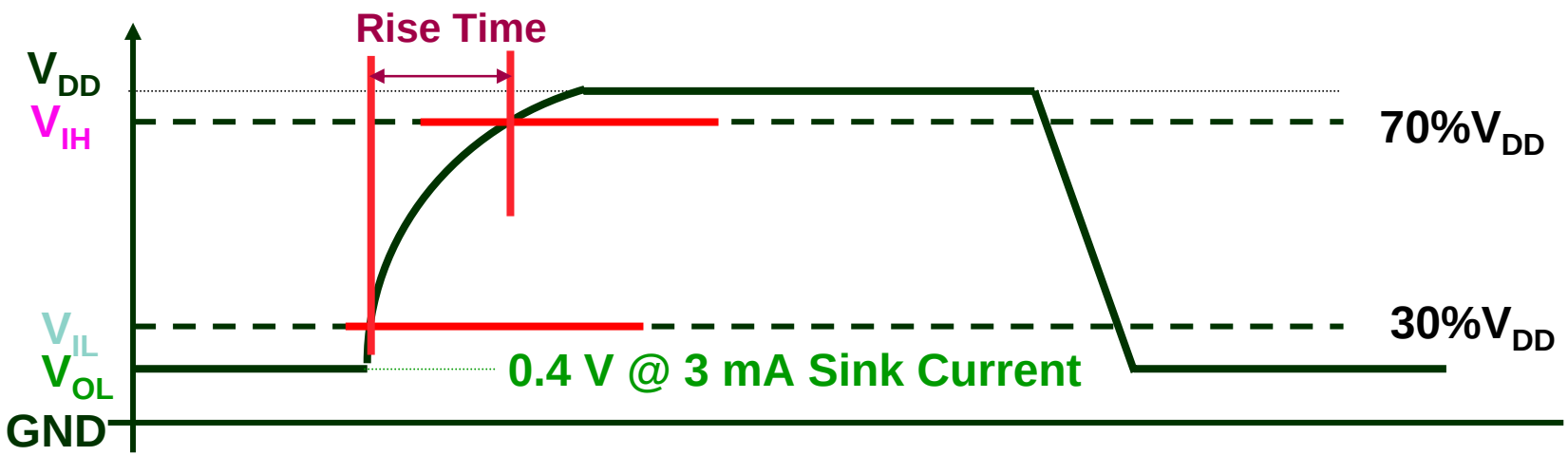
SDA/SCL Driver Architecture

- SDA and SCL are open drain/collector
  - Required pull-up resistors to pull the line to logic “1”
- Clock stretch is possible when a device is busy



Key Electrical Parameters

	Standard Mode	Fast Mode	Fast Mode Plus	High Speed Mode	
Bit Rate (kb/s)	0 to 100	0 to 400	0 to 1000	0 to 1700	0 to 3400
Max Load (pF)	400	400	560	400	100
Rise time (ns)	1000	300	120	160	80
Noise filter (ns)	-	50	10	10	10



Calculating Pull-up Resistors

1) $R_{MIN} < R_{PU} < R_{MAX}$

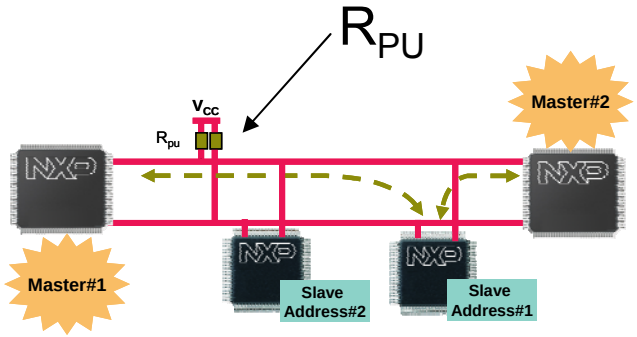
2) $R_{MIN} = (V_{DDMAX} - V_{OLMAX}) / I_{OLMAX}$

V_{DDMAX}	V_{OLMAX}	R_{MIN}		
		I_{OLMAX} = 3mA	I_{OLMAX} = 6mA*	I_{OLMAX} = 30mA**
3.6 V	0.4 V	**1.1 kΩ**	**533 Ω**	**106 Ω**
5.5 V	0.4 V	**1.7 kΩ**	**850 Ω**	**170 Ω**

*With a buffer; **Fast-mode Plus

3) $R_{MAX} * C_{MAX} = 1.18 * t_r$

MODE	Frequency	t_r	C_{MAX}	R_{MAX}
Standard	100 kHz	1000 ns	400 pF	**2.96 kΩ**
Fast Mode	400 kHz	300 ns	400 pF	**885 Ω**
Fast Mode Plus	1000 kHz	120 ns	560 pF	**252 Ω**



Glossary

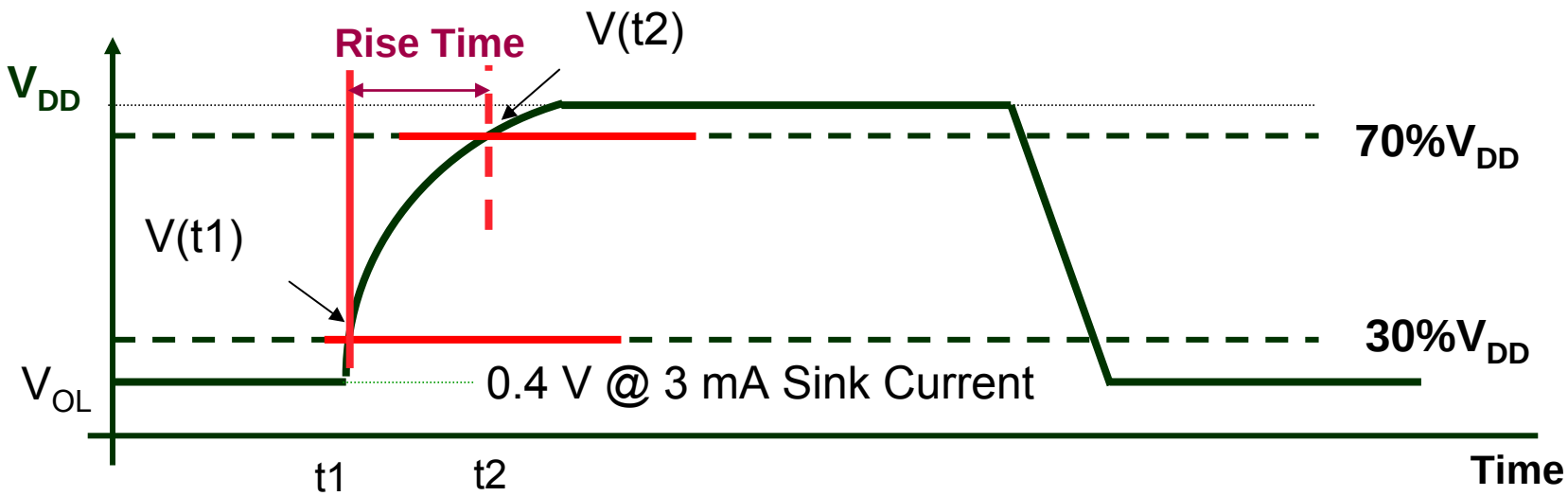
- R_{PU}: Pull-up resistor
- R_{MIN}: Minimum pull-up resistor
- R_{MAX}: Maximum pull-up resistor
- V_{DDMAX}: Maximum supply rail
- V_{OLMAX}: Maximum output voltage low
- I_{OLMAX}: Maximum sink current
- C_{MAX}: Maximum load capacitance
- t_r: Rise time

How Do You Derive Rise Time for I²C-bus

I²C-bus rise time is determined as in the following :

1) $V(t1) = 0.3*V_{DD} = V_{DD}\ (1–1/et1/RC) \rightarrow t1 = 0.3566749*RC$ (EQ1)
2) $V(t2) = 0.7*V_{DD} = V_{DD}\ (1–1/et2/RC) \rightarrow t2 = 1.2039729*RC$ (EQ2)
3) Subtract EQ1 from EQ2 → $t_{\text{rise time}} = t2–t1 = 0.8472979*RC$ or

   $R*C = 1.18*t_{\text{rise time}}$



Effects of Pull-up Resistors

- Minimum pull-up resistor limits the maximum current sink that affects the voltage output low (V_{OL}).
  - Increasing pull-up resistor above R_{MIN} leads to decreasing V_{OL} and higher noise margin
  - Decreasing pull-up resistor below R_{MIN} leads to increasing V_{OL} and lower noise margin
- Maximum pull-up resistor affects the rise time and speed
  - Increasing pull-up resistor above R_{MAX} leads to slower/possible rise time violation or lower speed
  - Decreasing pull-up resistor below R_{MAX} leads to faster rise time and speed

Measurements using an Oscilloscope

Using Cursors to make measurements

- Allows one to make quick measurements for ΔV Δt and
- Handy in making rough rise time measurements for I²C-bus

Effects of Pull-up Resistors

- Rise time for SDA and SCL lines

using 9K pull up resistors

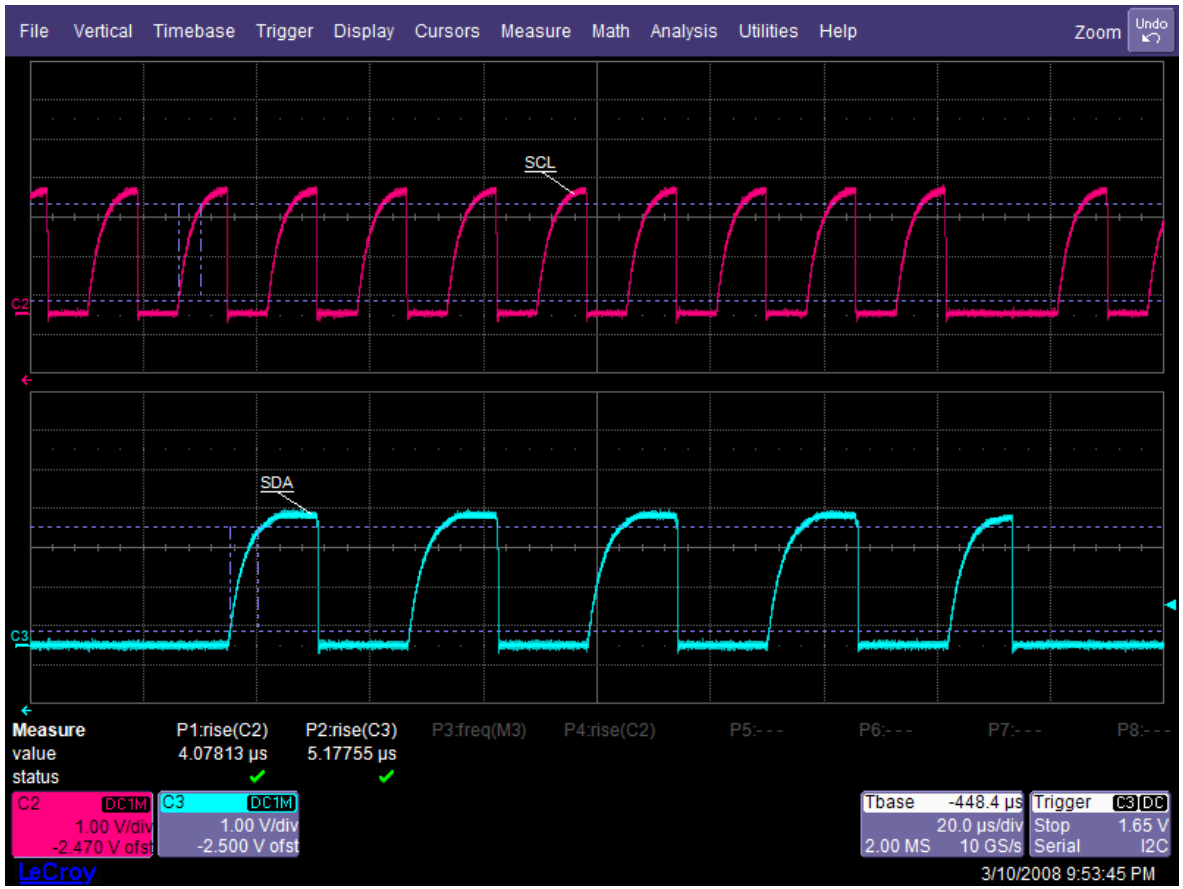
using 2.2K pull up resistors

Effects of Pull-up Resistors

- Side by side comparison of the difference in rise time using two different pull-up resistor values

Effects of Pull-up Resistors

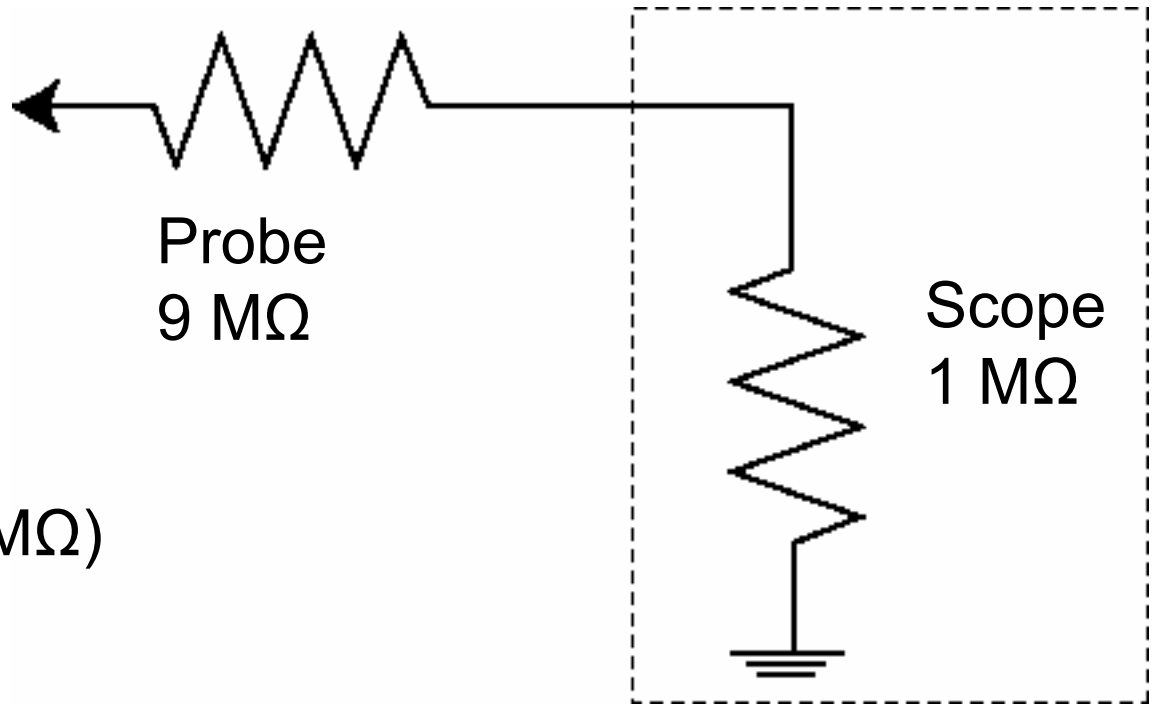
- Rise time for SDA and SCL lines using 100K pull up resistors
- Slow rise time, no stable logic high state.



Probing your I²C-bus

Passive Probe Basics

- Passive probes make an attenuator circuit with the probe impedance and scope impedance



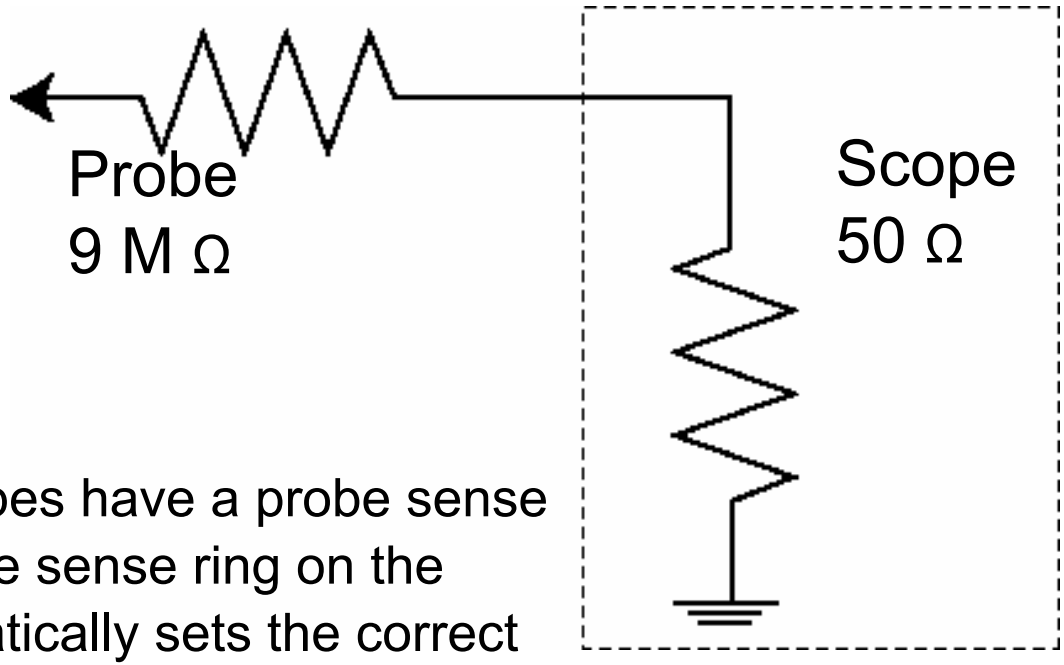
Attenuation ratio

1 M Ω /(9 MΩ +1MΩ)
= 1/10

Importance of Oscilloscope Coupling

- Signal is attenuated too much if the coupling is set incorrectly

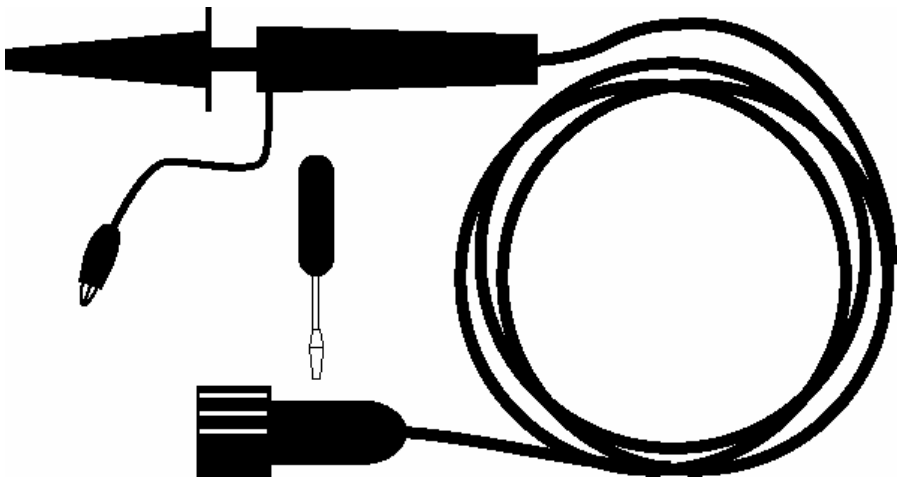
Attenuation ratio
50 Ω /(9M Ω +50 Ω)
= 1/180,001

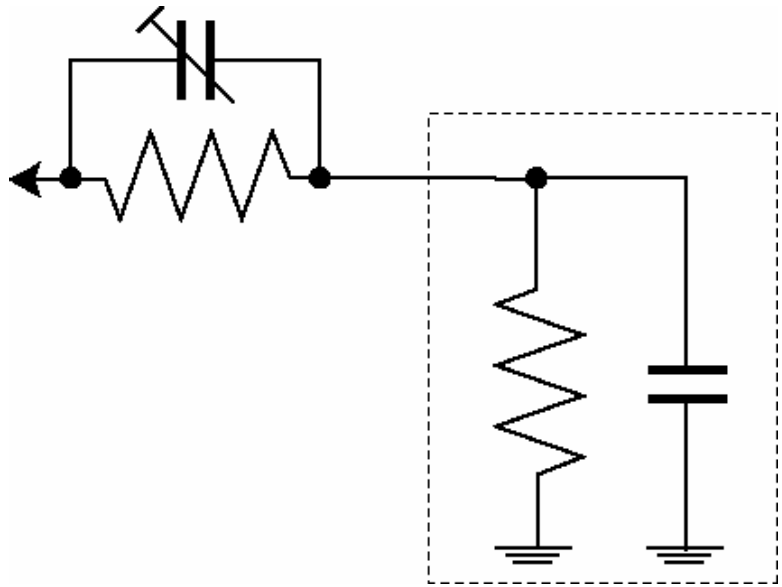


- Most modern passive probes have a probe sense pin that mates with a probe sense ring on the oscilloscope – this automatically sets the correct coupling and attenuation factor

Scope/Probe Impedance Matching

- Passive probes have an adjustment that allows the user to tweak the impedance of the passive probe to match the impedance of the scope it is connected to, this low frequency adjustment results in improved pulse shapes on the oscilloscope display



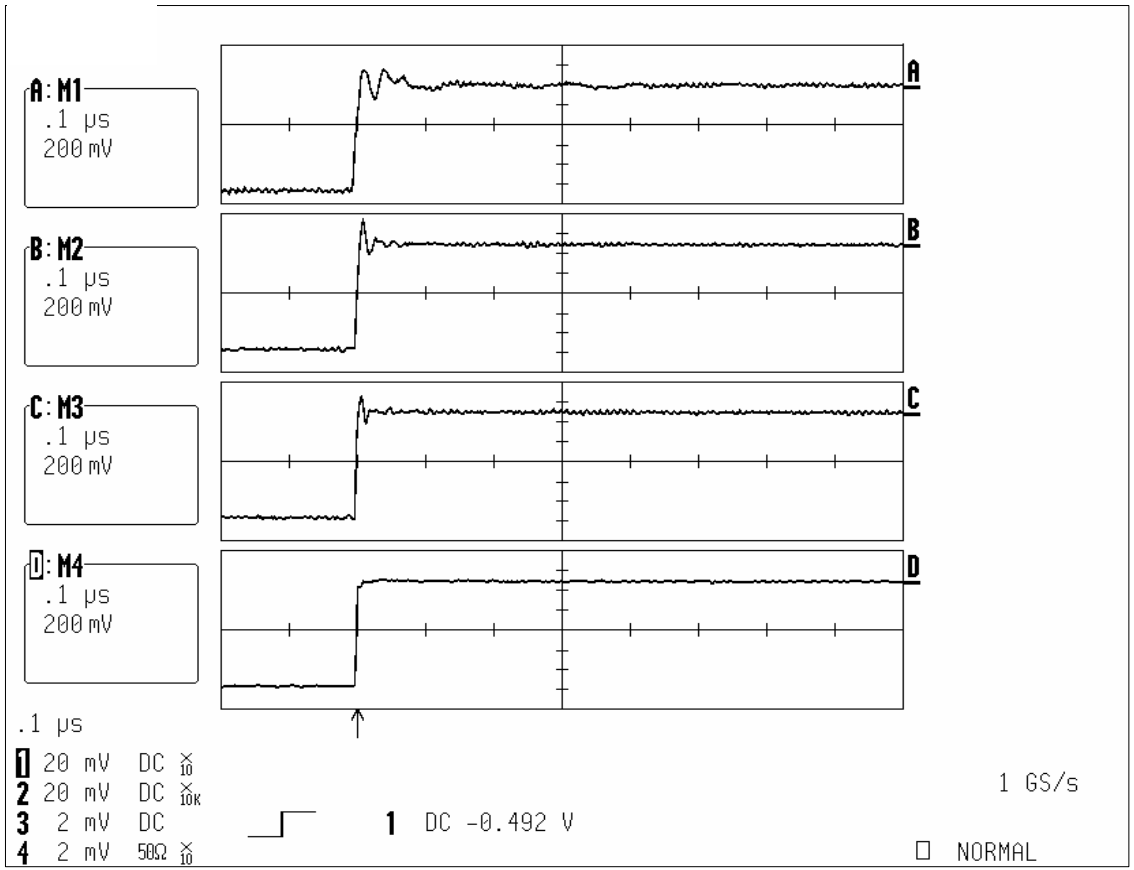


Passive Probe Adjustment

- All oscilloscopes have a “Cal Out” which provides a clean square wave for passive probe adjustment and compensation
- Adjusting the capacitor in the probe allows the probe to be tuned for that scope and resulting in the good pulse shape shown below

Importance of ground lead length while probing

- Not having proper ground connection, shows distorted signal



Pulse measured with ÷10 passive Probe

A: Without Ground Lead

B: 50 cm Ground Lead

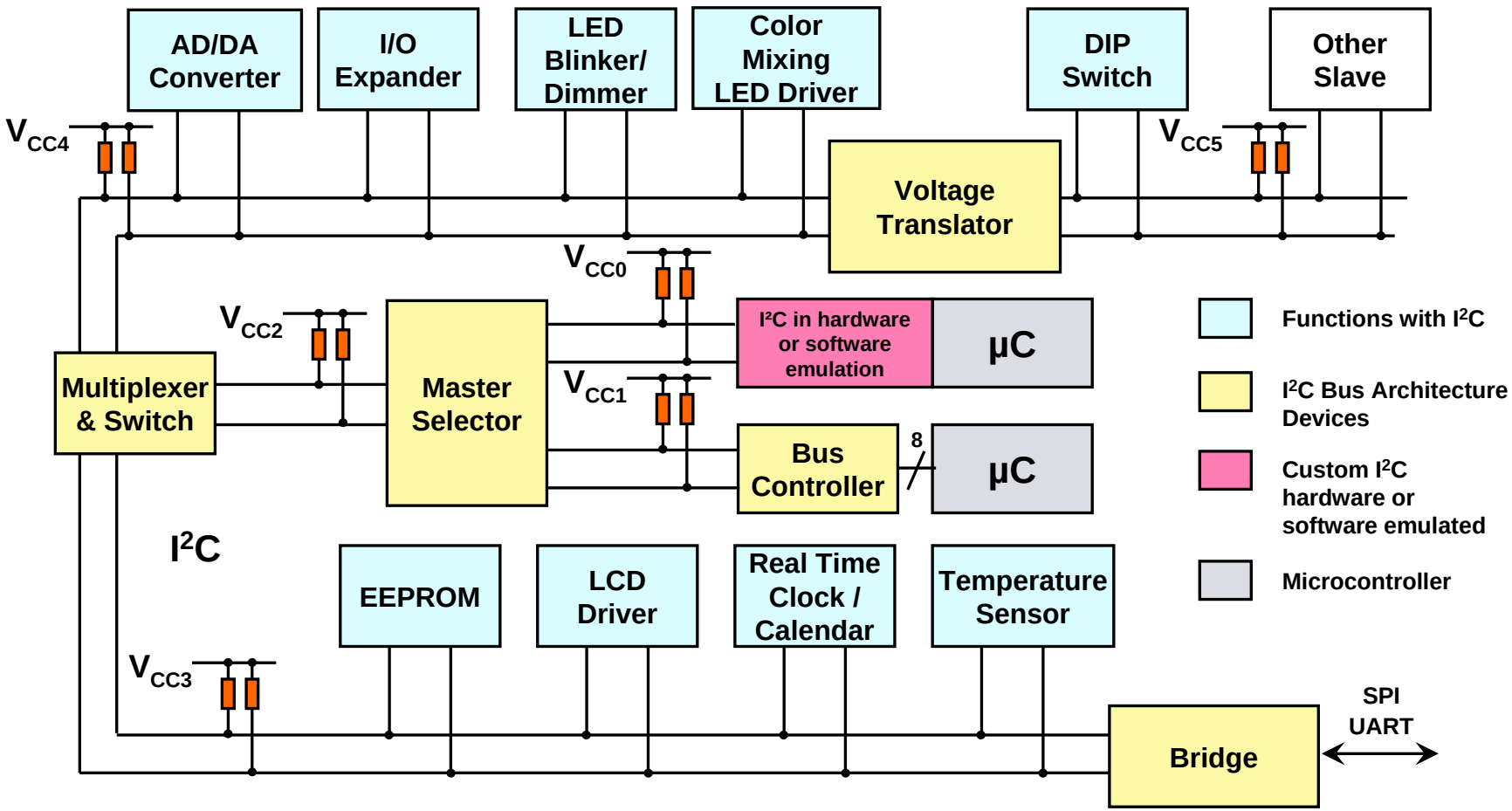
C: 10 cm Ground Lead

D: BNC Direct Cable Connection

(true signal shape)

Applications

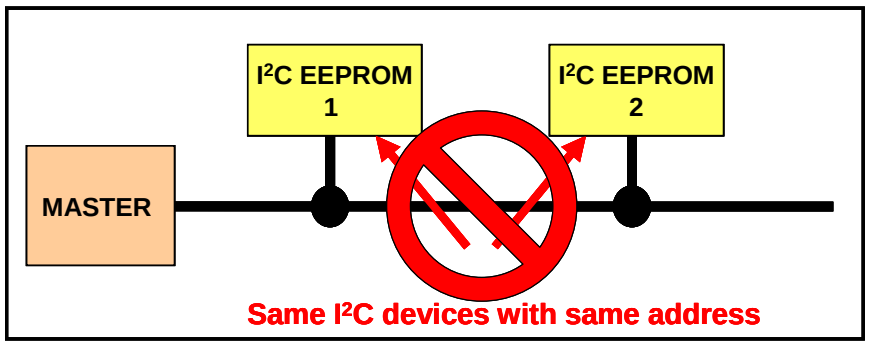
I²C Device Category



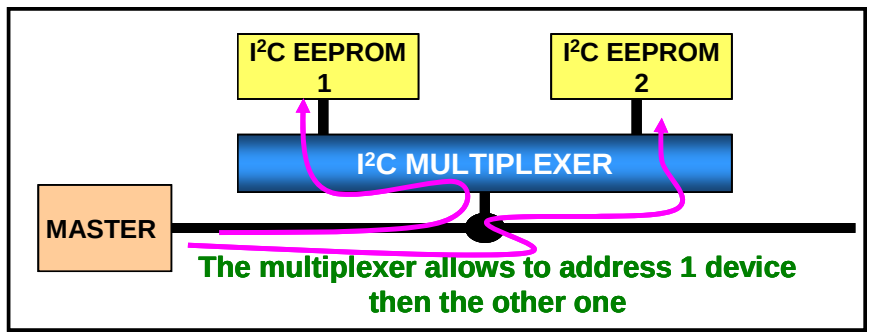
I²C Multiplexers to De-conflict Address

- Problem: More than one device with the same I²C address is not allowed on the bus
- Solution: Use an I²C multiplexer to allow more than one same device in the bus
- Multiplexer allows to split dynamically the main I²C bus into several isolatable sub branches
  - Programmable through I²C so no additional pins are required for control
  - More than one multiplexer can be plugged in the same I2C bus

EEPROM 1 and 2 have same address



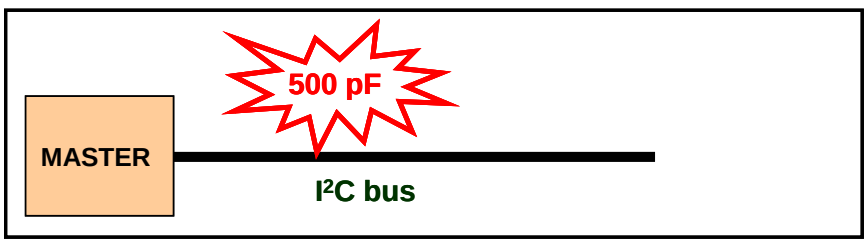
I²C multiplexer de-conflicts the address



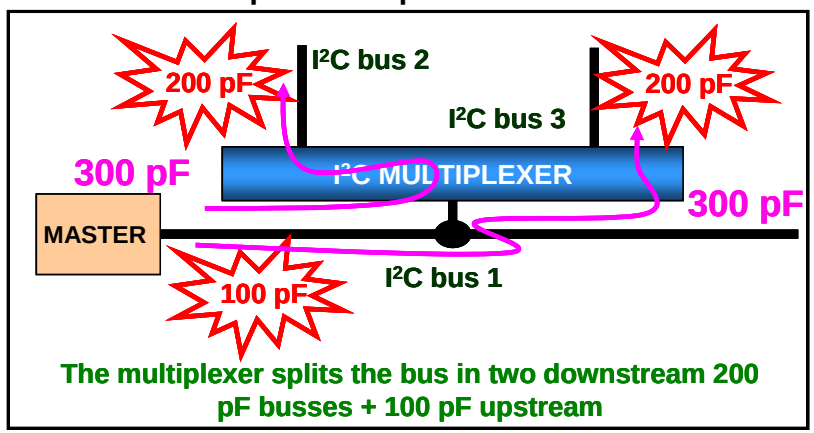
I²C Multiplexers Split Capacitive load

- Problem: Bus capacitance exceeds maximum load limit of 400 pF. If the load is higher, AC parameters will be violated
- Solution: Use an I²C-bus multiplexer to split dynamically the main I²C-bus to several sub-branches in order to divide the bus capacitive load
  - Programmable through I²C-bus so no additional pins are required for control
  - More than one multiplexer can be plugged in the same I²C-bus
  - LIMITATION: All the sub-branches cannot be addressed at the same time

Capacitive load exceeds 400pF

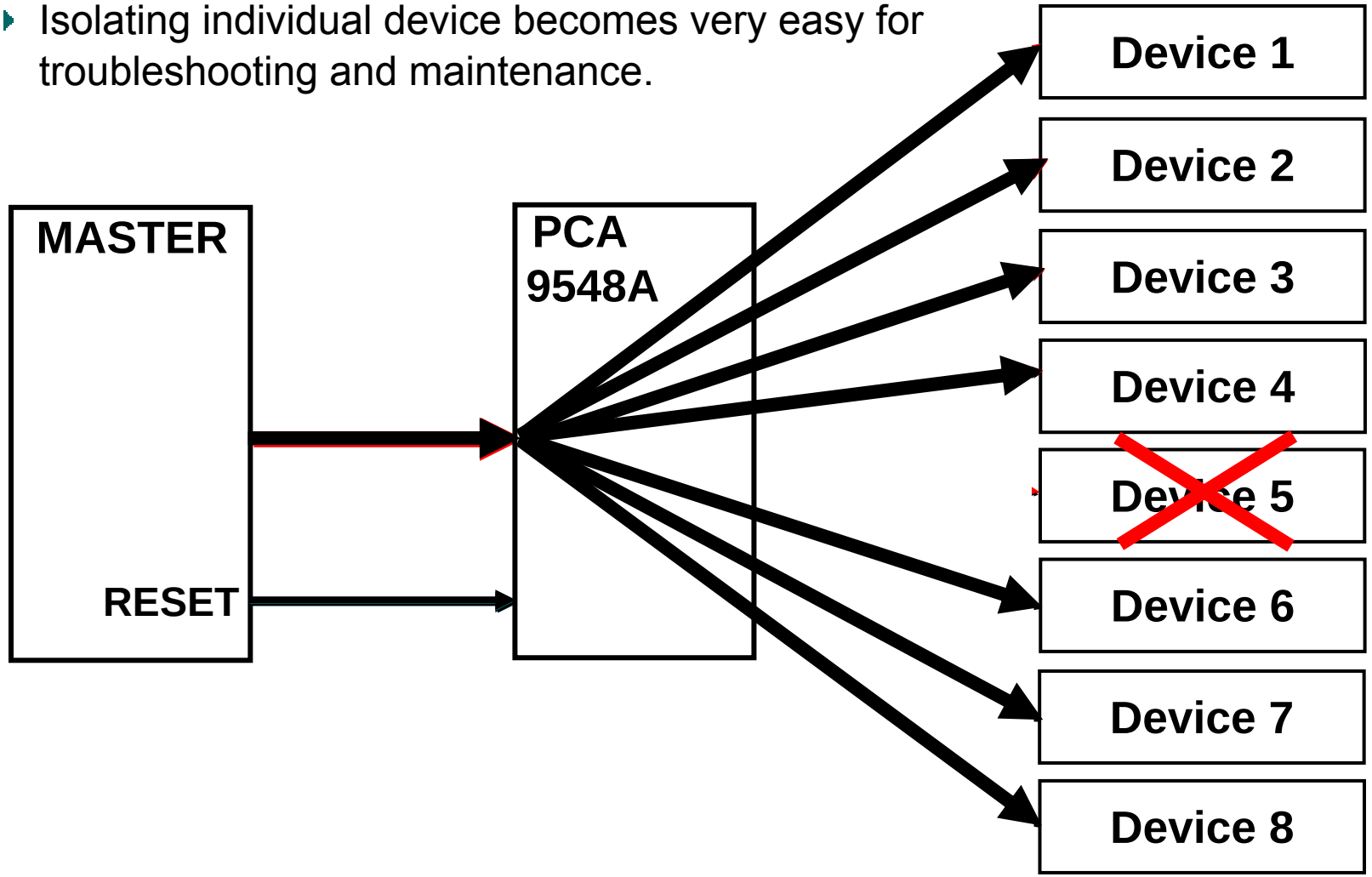


I²C MUX splits capacitive load



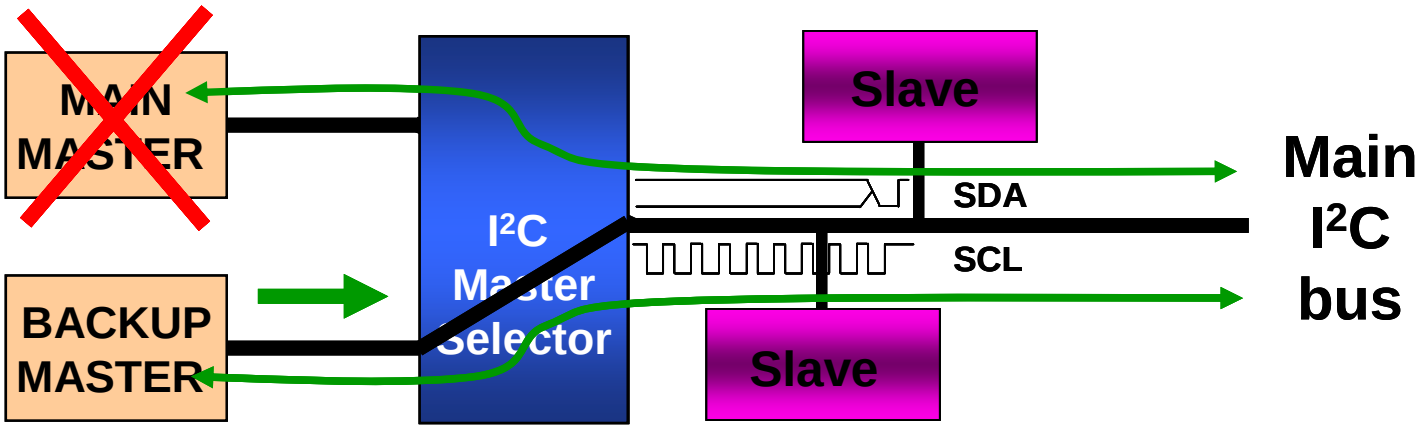
I²C Switch Isolates An I²C Hanging Segment

- Isolating individual device becomes very easy for troubleshooting and maintenance.



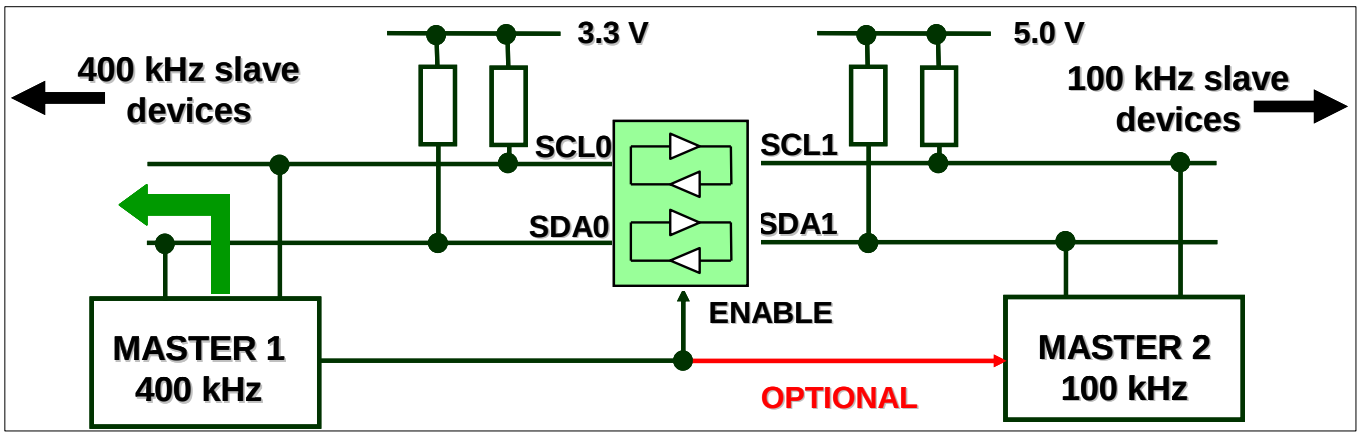
Master selector Isolates Failing Master

- Main Master control the I²C-bus
- When it fails, backup master asks to take control of the bus
- Previous master is then isolated by the selector
- Downstream bus is initialized (all devices waiting for START condition)
- Switch to the new master is done



Buffer Accommodates Different Bus Speeds

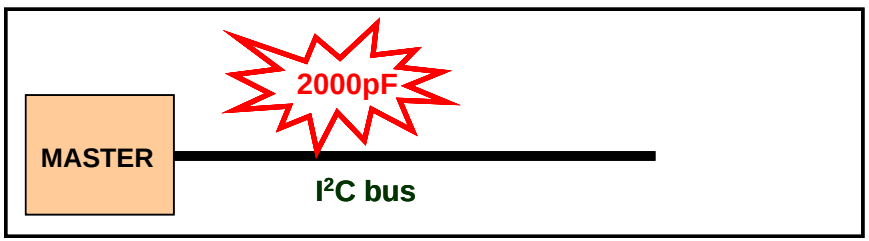
- Master 1 works at 400 kHz and can access 100 & 400 kHz slaves at their maximum speed (100 kHz only for 100 kHz devices)
- Master 2 works at only 100 kHz
- PCA9515A is disabled (ENABLE = 0) when Master 1 sends commands at 400 kHz



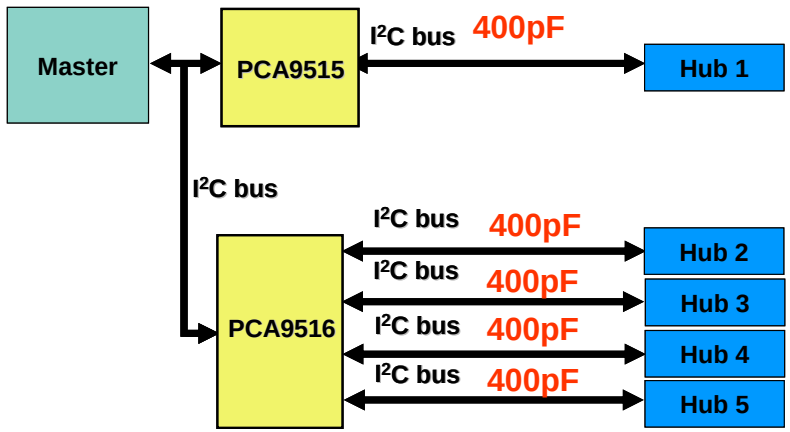
I²C Bus Repeater/Hub Split Capacitance

- Problem: Line capacitance is 2000pF.
- Solution: Isolate it with buffer and repeater
- PCA9515 and PCA9516 were designed to isolate up to 400 pF on each segment and uses an offset VOL to allow bi-directional signaling without use of a direction pin.

Bus has 2000pF

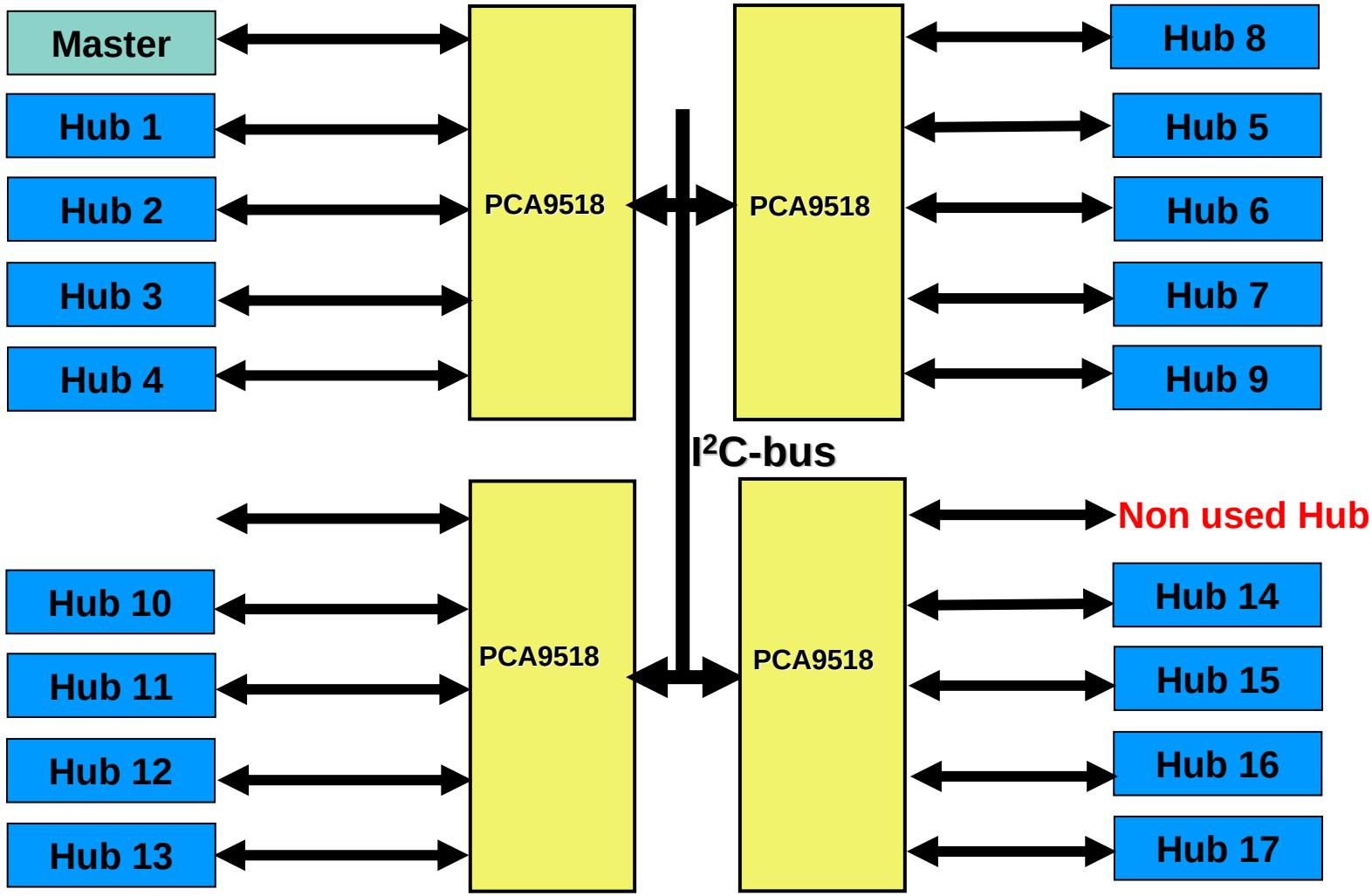


Repeater/Hub split capacitance



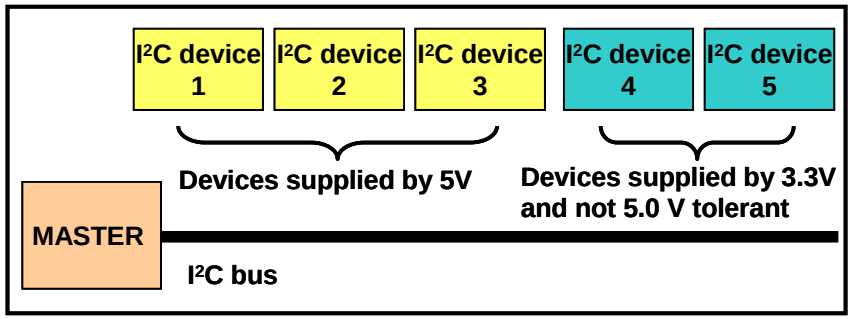
PCA9518 Hub Expander Applications

- **PCA9518 was design to allow expansion to an unlimited number of segments of 400 pF each.**

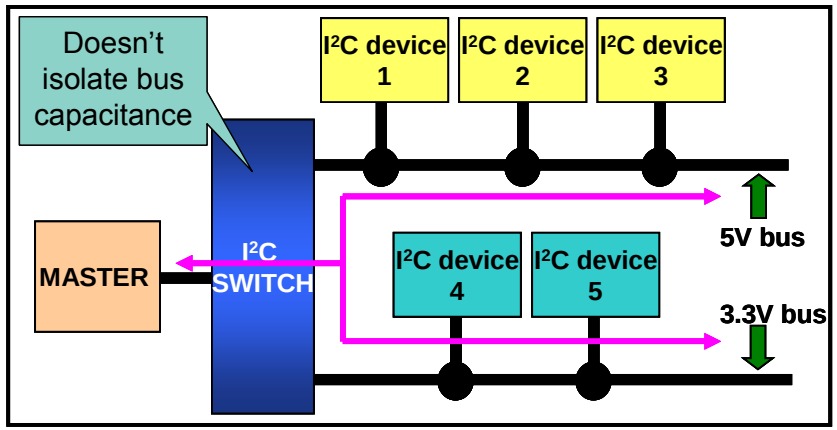


Voltage Level Shift

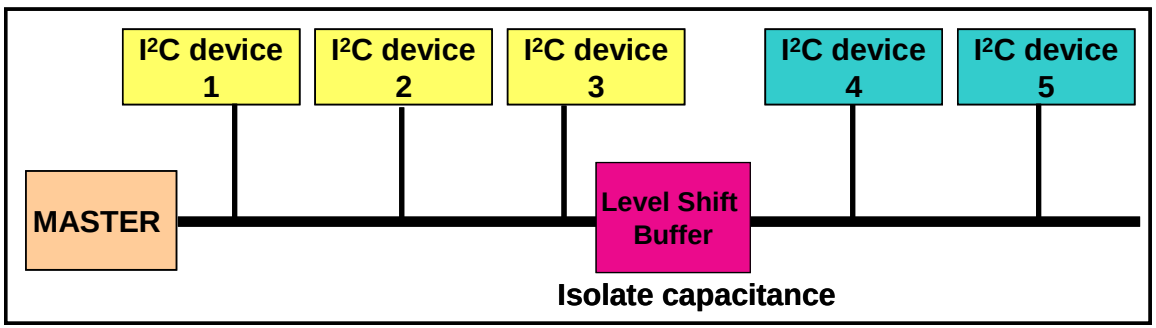
Bus has different voltage levels



Level shift with I2C switch



Level switch with level shift buffer

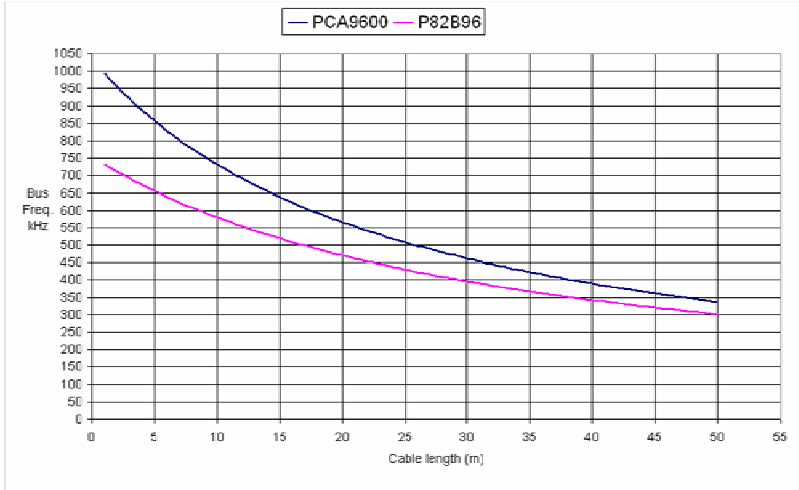


Two types of Level Shifters:

1. No capacitance isolation
2. With capacitance isolation

I²C-bus Application Over Long Cable

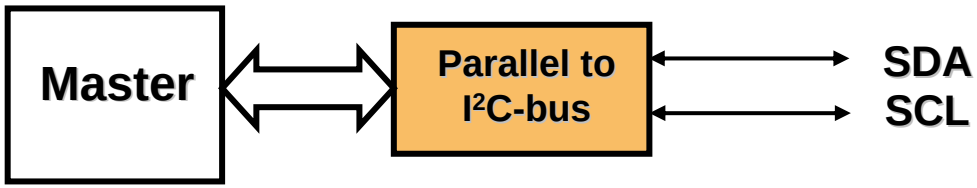
- Problem: Due to the bus 400 pF maximum capacitive load limit, sending commands over wire (80 pF/m) long distances is hard to achieve
- Solution: Use an I²C-bus extender
- Bus Extenders' characteristics
  - High drive outputs, up to 30mA
  - Possible distances range from 50 meters at 85 kHz to 1km at 31 kHz over twisted-pair phone cables. Up to 400 kHz over short distances (~ 10 m)



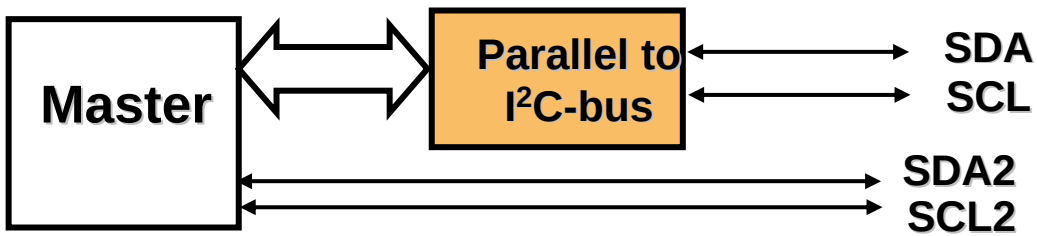
Device	Distance	I²C-bus Speed
PCA9517A	**10m**	0 – 400kHz
PCA9507	**20m**	0 – 100kHz
P82B96	**50m**	0 – 400kHz
PCA9600	**50m**	0 – 1MHz

Parallel Interface to I²C-bus Master

- Allow a master without I²C-bus interface to send I²C command



- Allow master add/isolated I²C-bus with the same device



Summary

I²C-Bus Features Summary

- **Only 2 bus lines required: data (SDA) and clock (SCL)**
- **Each device is software addressable by a unique address**
- **2 communication modes:**
  - Master-Transmitter/slave receiver
  - Master-Receiver/slave transmitter
- **Multi-master capable protocol:**
  - Clock synchronization
  - Data arbitration
- **Serial bi-directional data transfers:**
  - 100 kbit/s — Standard-mode
  - 400 kbit/s — Fast-mode (Fm)
  - 1 Mbit/s — Fast-mode Plus (Fm+)
  - 1.7 – 3.4 Mbit/s — High-speed mode (HS)
- **Maximum bus capacitance**
  - 400 pF (without repeaters) which is about 20 – 30 devices or 10 ft of wire for Standard/Fast-mode
  - 560 pF to 4000pF for Fast-mode Plus

I²C-bus Benefits

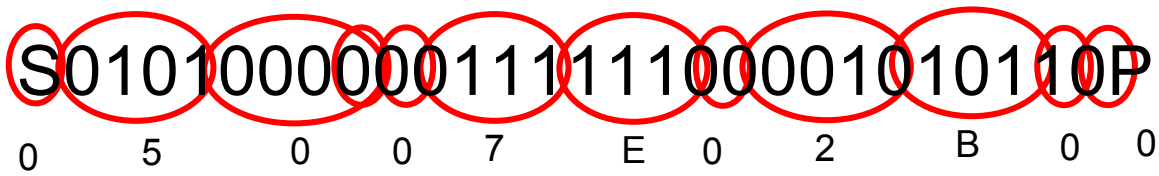
- Well known bus
- Standard adopted by all the industry:
  - Computing - Networking - Automotive
  - Industrial - Telecom - Consumer
- Used in many types of applications:
  - PC - DVD/STB - Mobile Devices
  - Printers - Routers - LCDTV
- Adopted by leading High-Tech companies
  - Intel - IBM - Compaq
  - Nokia - Cisco - HP
- Life of products: designed to stay in the market for several years

I²C Designer Benefits

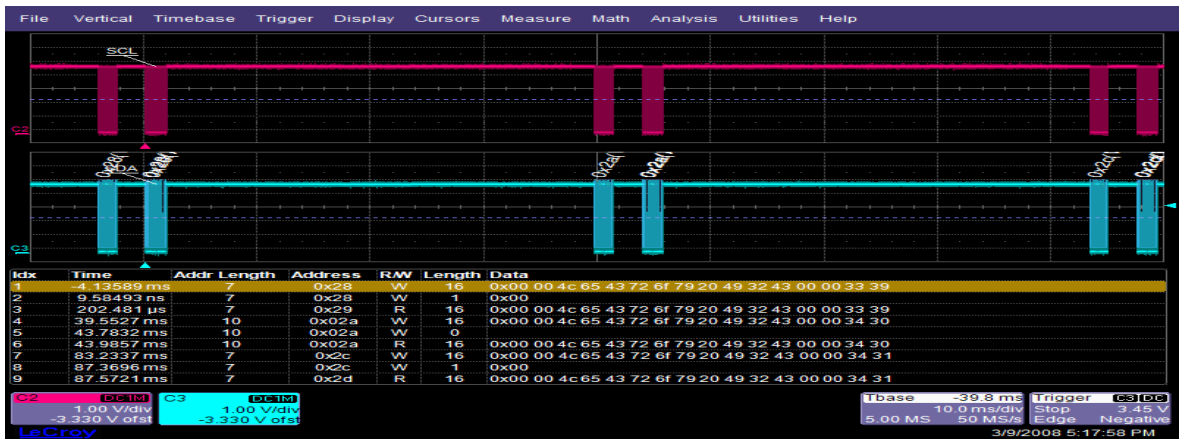
- No need to re-invent new bus interfaces
- Allow systems to be completely software-defined
- The same IC types used across in many different applications
- Master and slave can be added to or removed from a system without affecting any other devices on the bus.
- Fault diagnosis and debugging are simple
- Simplify software development time

I²C Debugging

- Measurements are made from message to message or from a message to some event that occurs before or after that message
- This is a tedious, time consuming process that is prone to a lot of mistakes



- Difficult for the user to find bit widths in long runs of 1s or 0s



I²C Debugging

- Need to locate and isolate specific address, or data message on a bus or always trigger on the start or end of a packet
- Traditional oscilloscope triggers provides no way to guarantee the user captures a specific data packet
  - Edge and width triggers might trigger on any bit
  - I²C trigger enables one to trigger on data of interest
  - Start, Stop, Frame Length, Address, etc…
- Perform various measurements such as
  - Rise time
  - Frequency
  - Amplitude

Useful Links for I²C Products and Support

I²C Discussion Forum

http://forums.nxp.com/forums/viewforum.php?f=6 (Forum)

- I²C Products

  http://www.standardics.nxp.com/interface/

Products used …

- **LeCroy Corporation:**

  WaveRunner 64Xi
  - 600 MHz Digital Storage Oscilloscope
  - 4 channel 10 GS/s
  - Serial Trigger & Decode options - I^2C, SPI, UART

  http://www.lecroy.com

  technical.support@lecroy.com (E-mail Support)

- **NXP Semiconductors**

  -LPC2368 – ARM7 Microcontroller

  - PCA9552 - 16 bit LED driver

  - PCA24S08 - I^2C Security EEPROM

  http://www.nxp.com/i2c

  I2C.Support @ nxp.com (E-mail Support)