

# CprE 488 – Embedded Systems Design

## Lecture 4 – Interfacing Technologies

Phillip Jones

Electrical and Computer Engineering

Iowa State University

[www.ece.iastate.edu/~phjones](http://www.ece.iastate.edu/~phjones)

[rcl.ece.iastate.edu](http://rcl.ece.iastate.edu)

*Never trust a computer you can't throw out a window. – Steve Wozniak*

# Announcements

- **Exam 1: Thursday 3/26**

- Can have 1-side of one-page of **handwritten** notes.
- There is a paper and a Canvas part of the Exam in class
  - Bring an electronic device for taking the Canvas part
- You are encouraged to have a calculator
- **See exam overview:**  
<https://class.ece.iastate.edu/cpre488/lectures/Exam1-review.pdf>

- **Class Project Topic:**

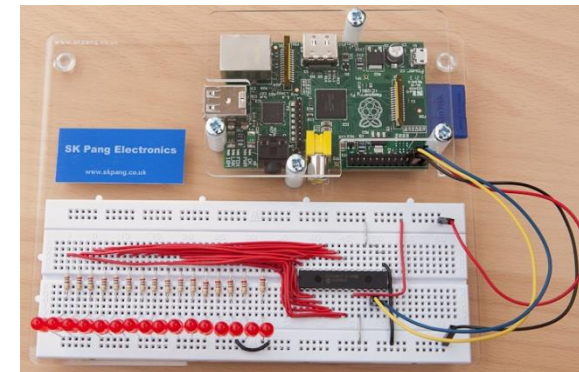
- Project Idea draft proposal, and draft Rubric (due **Wed 4/1**)
- Project Proposal Presentation (In class **Tue 4/7**)
- Project Demos in Lab at Final Exam time: **Thur 5/14, 9:45am**)
  - Demos similar to normal MP demo day, with the option to **additionally** give a live demo
  - **Make sure your grading Rubric is the last slide of your presentation.**

# System Connectivity

- The communication strategies we have explored so far (direct memory mapped I/O, shared buses and memory) are most commonly found in SoCs
- For multi-chip / multi-board systems, the communication needs can be different:
  - Performance is not (necessarily) a significant factor
  - Simplicity in terms of hardware and wiring (cost)
  - Configurability, expandability
- Many one-off solutions, but several formal and informal standards have emerged as well



Freescale Tower System  
Modular Development Platform



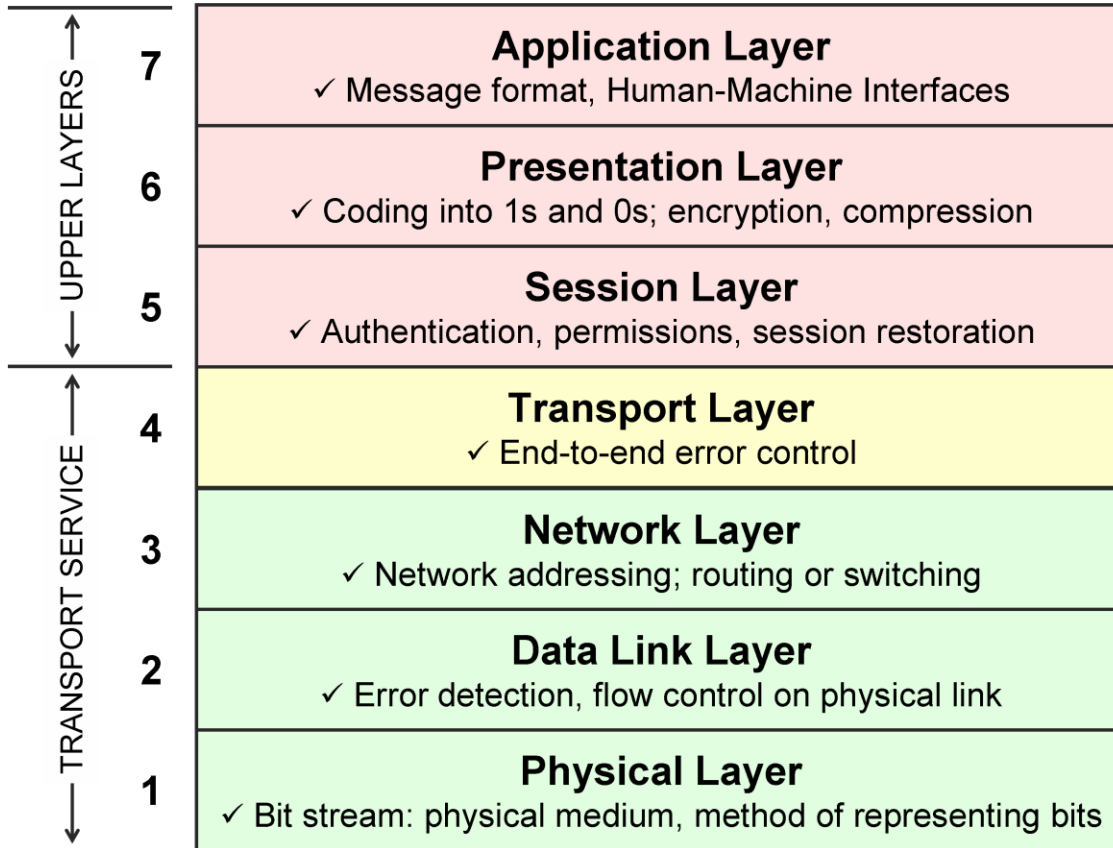
Raspberry Pi with i2c  
I/O Expander

# This Week's Topic

- Serial communication bus standards
  - Inter-Integrated Circuit (I<sup>2</sup>C)
  - Serial Peripheral Interface (SPI)
  - Controller Area Network (CAN)
  - Quick Path Interconnect (QPI)
- Reading: Wolf chapter 8.4.1, 8.4.2, 9 (CAN), 10.4.5 (I2C)

# A Note on Network Stacks

- Networks (including the serial buses we will discuss) can be classified using the well-known International Organization for Standardization Open Systems Interconnection (**ISO-OSI**) model:



- Our interest is mainly in layers 1 and 2, with software (potentially) providing higher level services

# UART

- 2-wire Bus (RX/TX), and a ground wire
- Covered in CPRE 288
- Summary of protocol properties

# UART

- 2-wire Bus (RX/TX), and a ground wire
- Covered in CPRE 288
- Summary of protocol properties

<b>Protocol</b>	<b>Expandable</b>	<b>Multi-Master</b>	<b>Duplex</b>	<b>Robustness</b>	<b>Speed</b>	<b>Flow Control</b>	<b>Protocol Overhead</b>
UART (2-wire)							

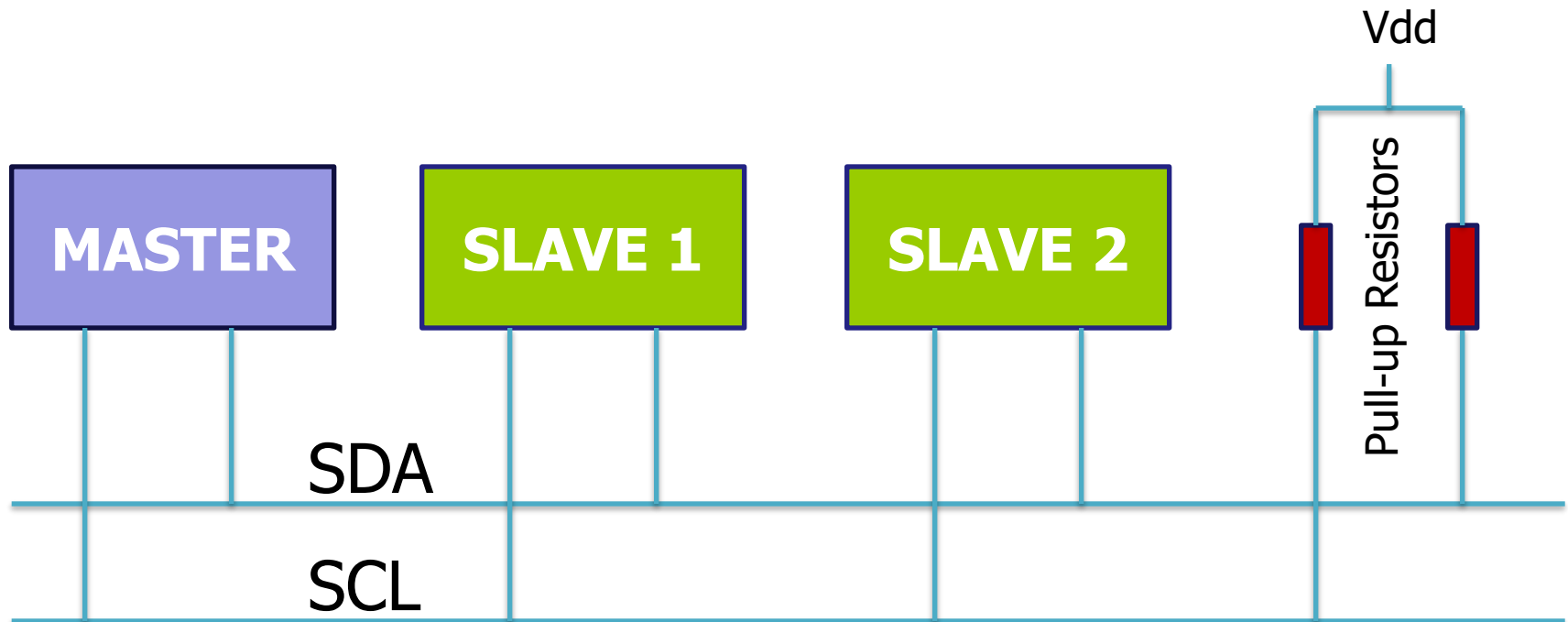
# UART

- 2-wire Bus (RX/TX), and a ground wire
- Covered in CPRE 288
- Summary of protocol properties

<b>Protocol</b>	<b>Expandable</b>	<b>Multi-Master</b>	<b>Duplex</b>	<b>Robustness</b>	<b>Speed</b>	<b>Flow Control</b>	<b>Protocol Overhead</b>
UART (2-wire)	No	No	Full	(Parity bit)	~100Kbps	No	Start, Stop, (parity)

# Inter-Integrated Circuit (I<sup>2</sup>C) Bus

- Low-bandwidth (100s of Kbps), short-distance, two-wire interface for communication amongst ICs and peripherals
- Originally developed by Philips Semiconductor for TV circuits, later became an industry standard

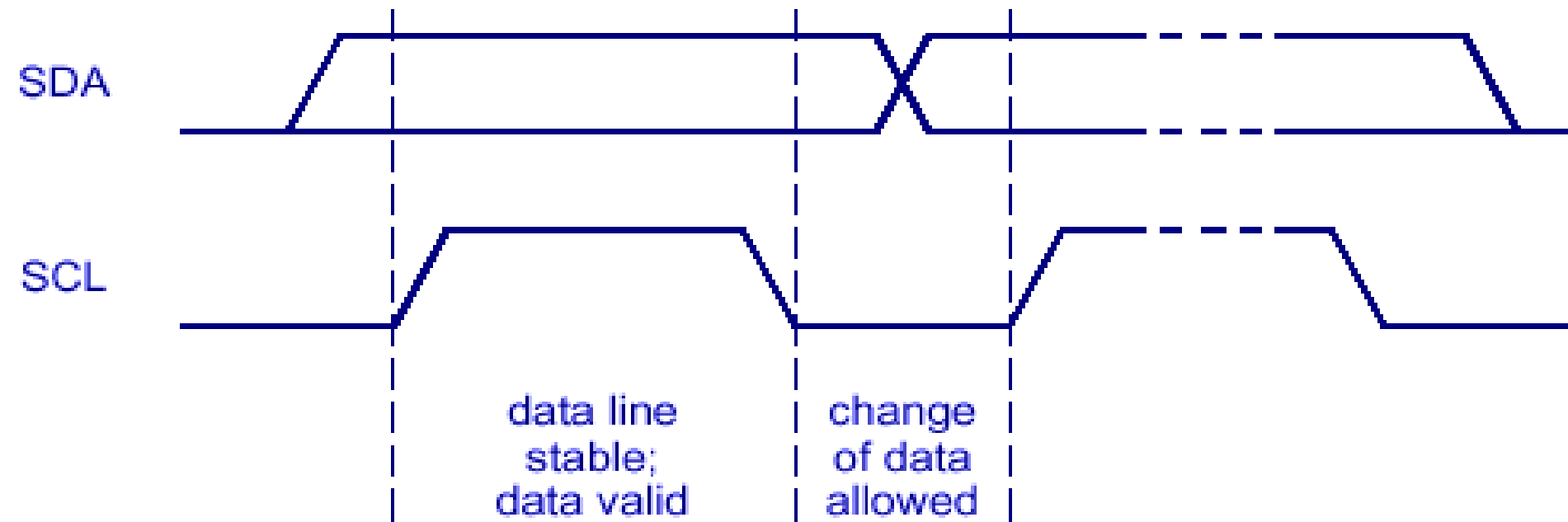


# I<sup>2</sup>C Features

- Multiple receivers do not require separate select lines as in other buses
  - At start of a I<sup>2</sup>C transaction a 7-bit (or 10-bit) device address is sent
  - Each device listens – if device address matches internal address, then device responds
- SDA (data line) is bidirectional, communication is half duplex
- SDA, SCL are open-drain:
  - require external pullup resistors
  - Allows multiple bus masters

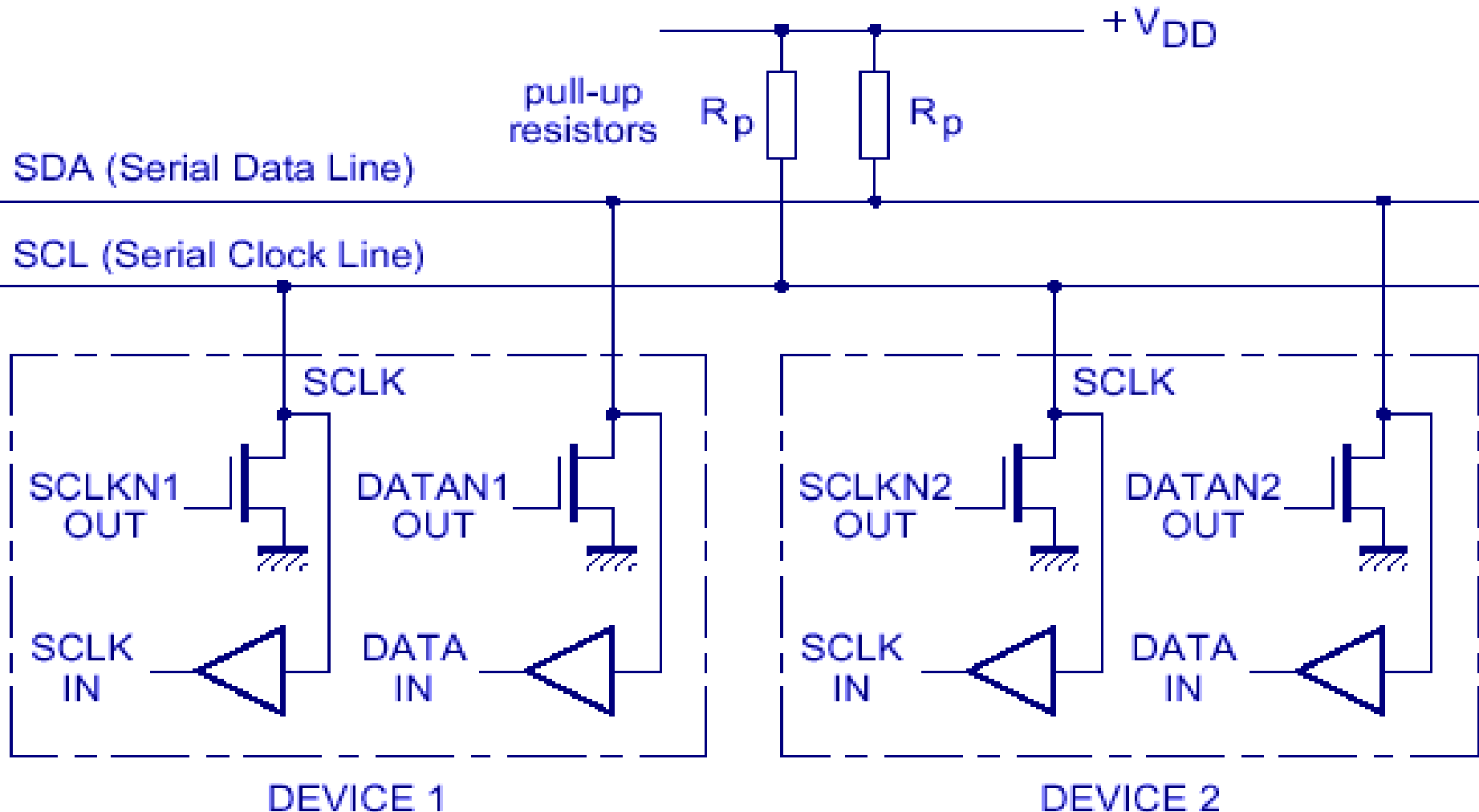
# I<sup>2</sup>C Bit-Transfer

- **Serial Clock (SCL):** A clock pulse generated for each bit transferred
- **Serial Data (SDA):**
  - Must be stable during the HIGH (i.e., 1) duration of SCL
  - SDA can change only when the SCL is LOW (i.e., 0)



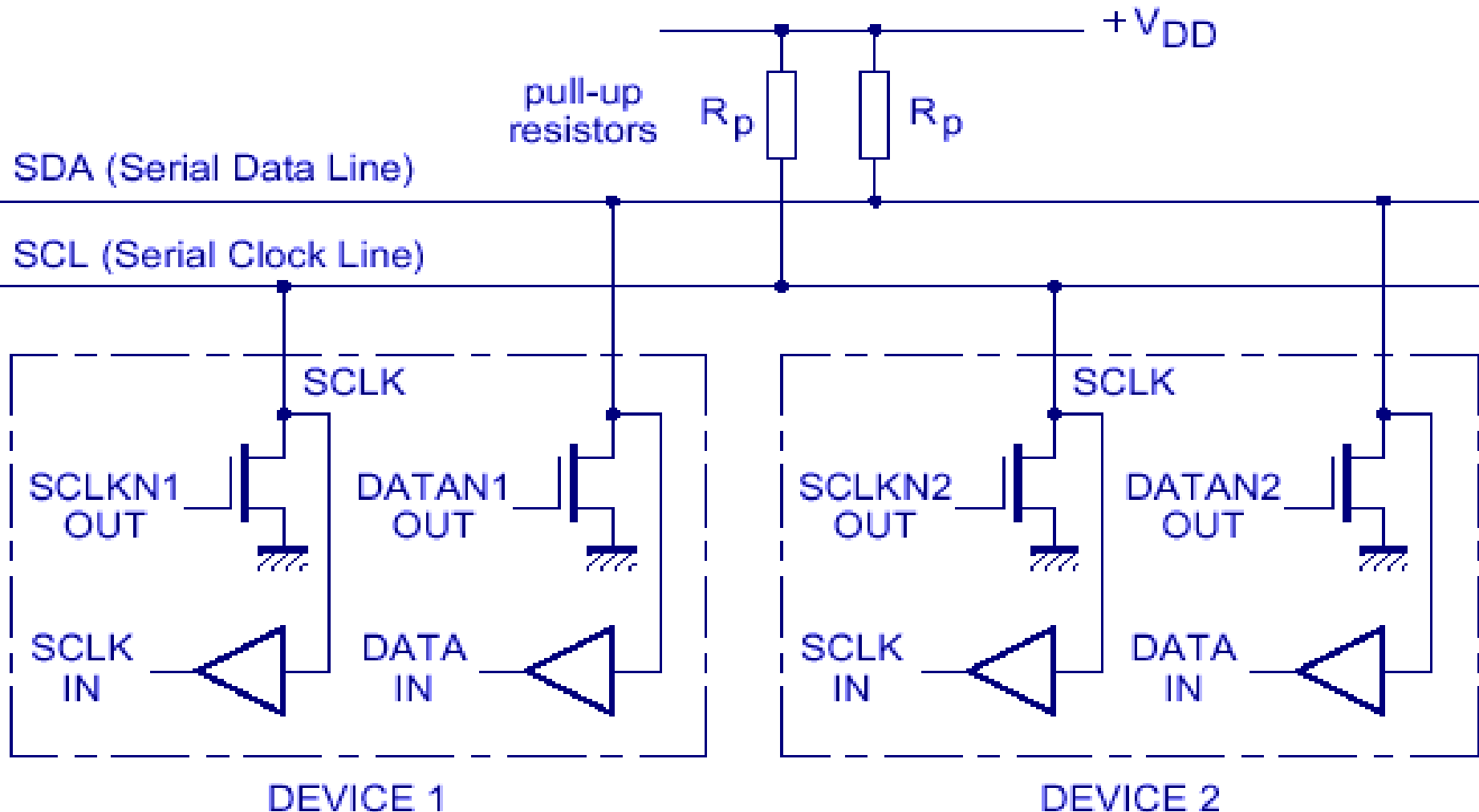
# I<sup>2</sup>C Bit-Transfer

- **Wired-and function:** open-drain or open-collector
  - SDA and SCL are pulled HIGH by a pull-up resistor
  - Devices can drive a LOW value, but not a HIGH value (important for Arbitration process)



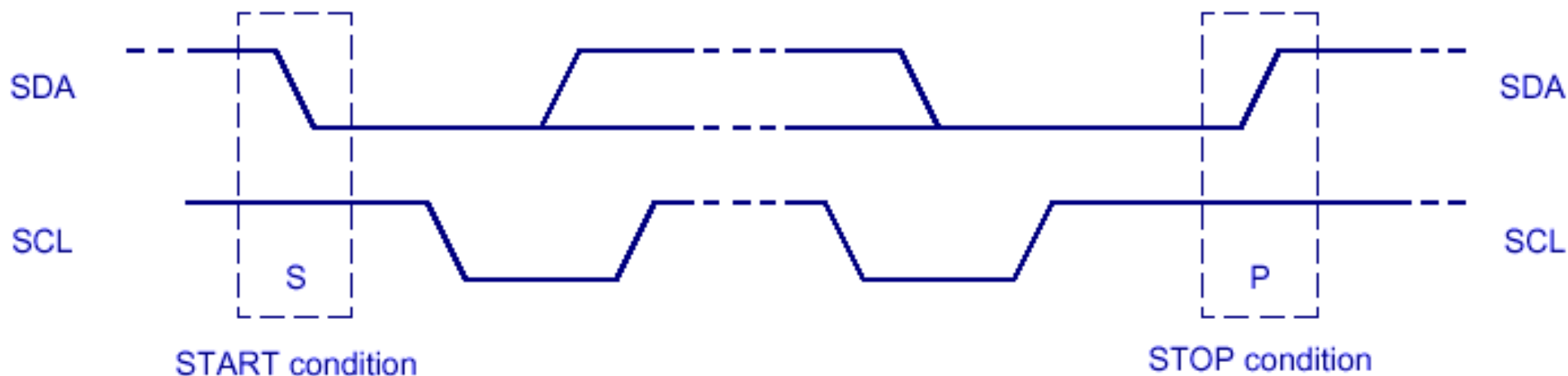
# I<sup>2</sup>C Bit-Transfer

- **Wired-and function:** open-drain or open-collector
  - SDA and SCL are pulled HIGH by a pull-up resistor
  - Devices can drive a LOW value, but not a HIGH value (important for Arbitration process)



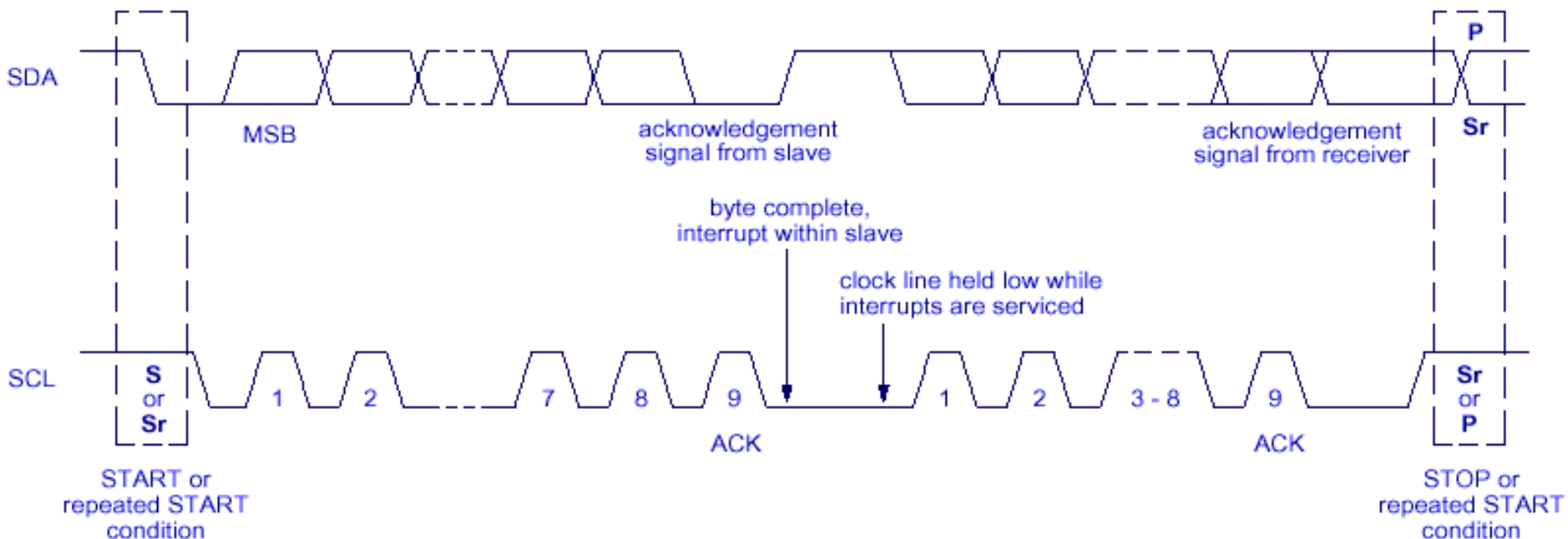
# I<sup>2</sup>C START/STOP Conditions

- **START condition:** Signals beginning of I2C transfer (obtain bus)
  - A HIGH to LOW transition on the SDA line while SCL is HIGH
- **STOP condition:** Signals end of transfer I2C (releases the bus)
  - A LOW to HIGH transition on the SDA line while the SCL is HIGH
- Both the START and STOP are always generated by the Master
- Repeated START condition is allowed by a Master
  - Repeated start is used for changing the slave, or changing the direction of data transfer (Send/Receive) for the same slave



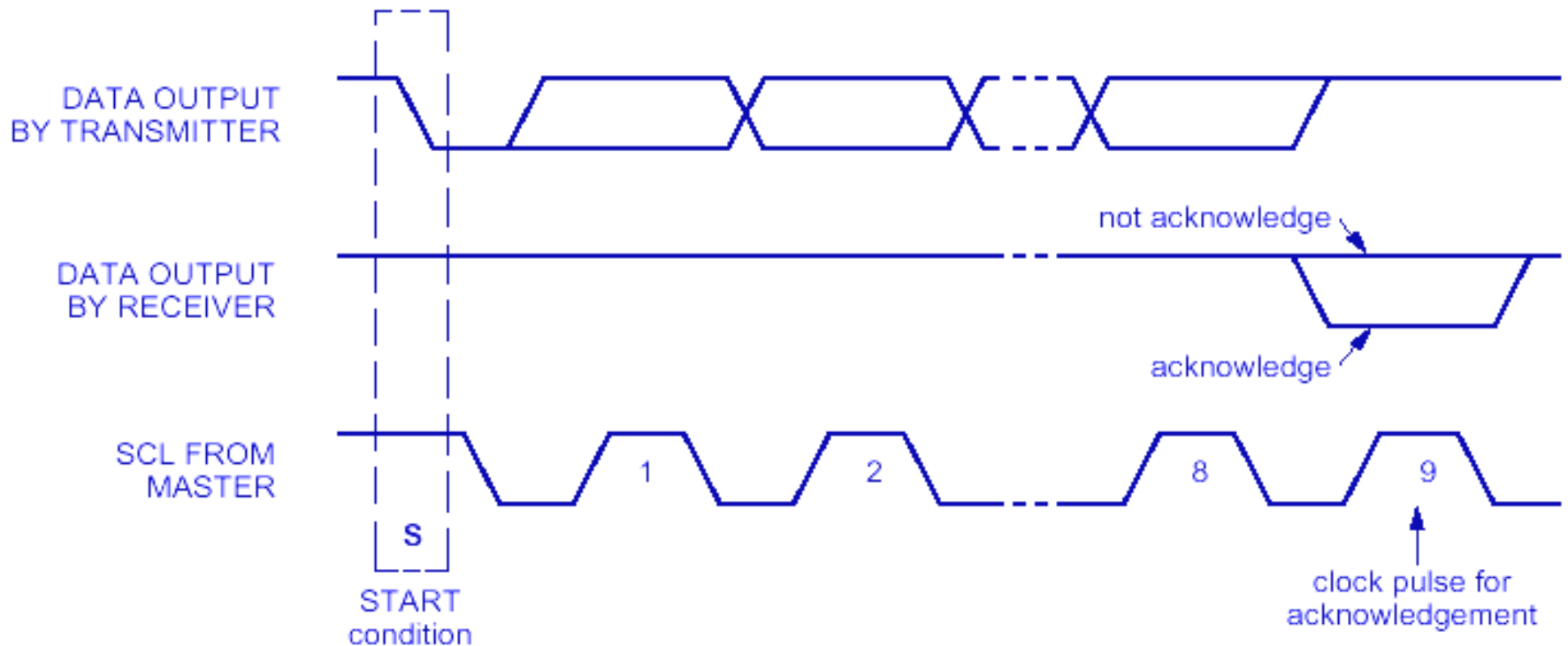
# I<sup>2</sup>C Data Transfer

- Every data transfer on the SDA line must be 8-bits long
- Each byte must be followed by an acknowledgement from receiver
- Data byte is transferred bit-wise with the MSB as the first bit sent
- Slave can force the master to wait by holding the clock line SCL LOW



# Acknowledgement Scheme

- The acknowledge-related clock-pulse is generated by the master
- The transmitter (master or slave) releases the SDA line i.e. SDA is HIGH for the ACK clock pulse
- Receiver must pull-down SDA line during the acknowledge clock pulse (stable LOW) during the HIGH period of the clock pulse

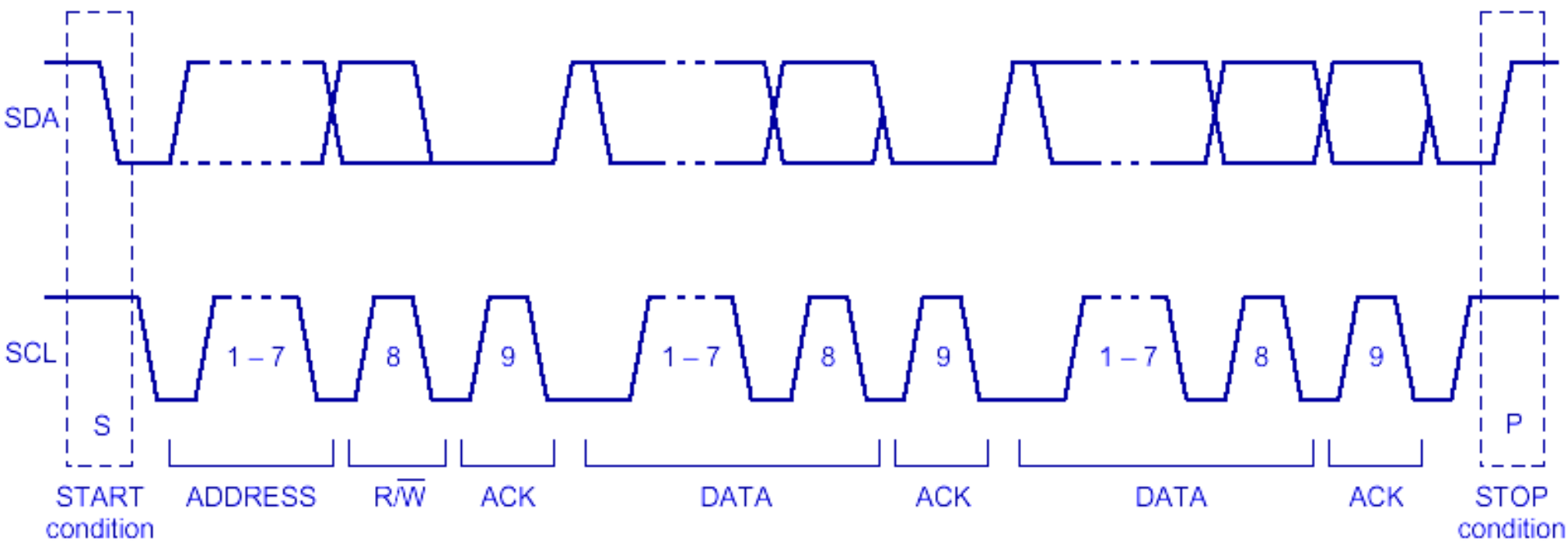


# Acknowledgement Scheme

- Receiver obliged to generate an acknowledge after each byte received
- When a slave does not acknowledge slave address (when busy), it leaves the data line HIGH. The master then generates either STOP or attempts repeated START
- If a slave-receiver does ack the slave address, but some time later during the transfer cannot receive more data (this is done by leaving SDA HIGH during the ack pulse), then the master either generates STOP or attempts repeated START
- If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an ack on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition

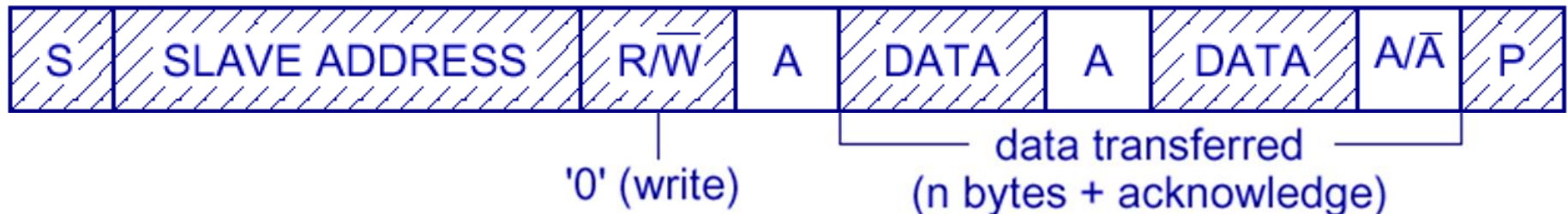
# Data Transfer With 7-Bit Device Address

- After START condition (S), a slave address (7-bit) is sent.
- A read/write (R/W') direction is then sent (8th bit)
- Data transfer occurs, and then always terminated by STOP condition. However, repeated START conditions can occur.



# Master to Slave Data Transfer (Write)

- **Master:**
  - Initiates the data transfer by generating the START condition
  - Then sends slave address "byte" (read/write bit set to 0, i.e. write mode)
- **Slave:**
  - ACK if address provided is for it
  - Then ACKs every 8-bits sent by the Master
- **Master:**
  - Sends DATA in 8-bit busts.
  - Generates STOP condition after Slave ACKs final byte sent by the Master



 from master to slave

 from slave to master

A = acknowledge (SDA LOW)

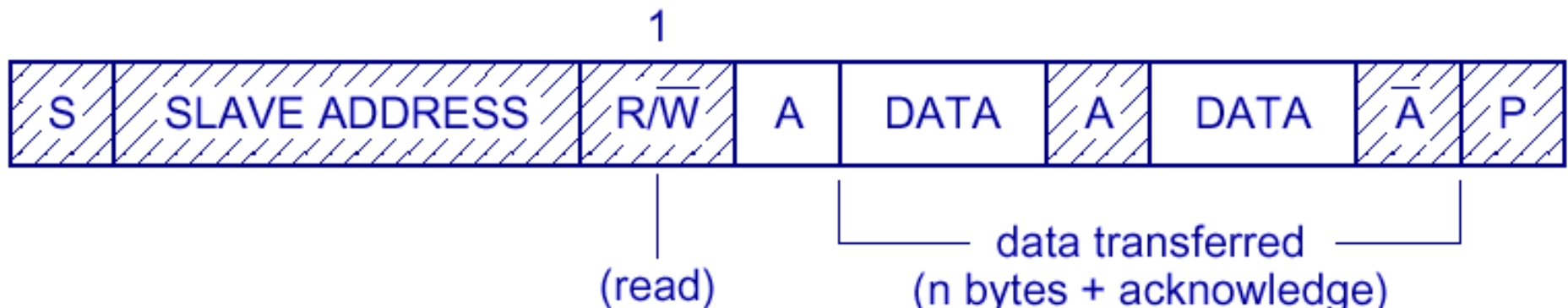
$\bar{A}$  = not acknowledge (SDA HIGH)

S = START condition

P = STOP condition

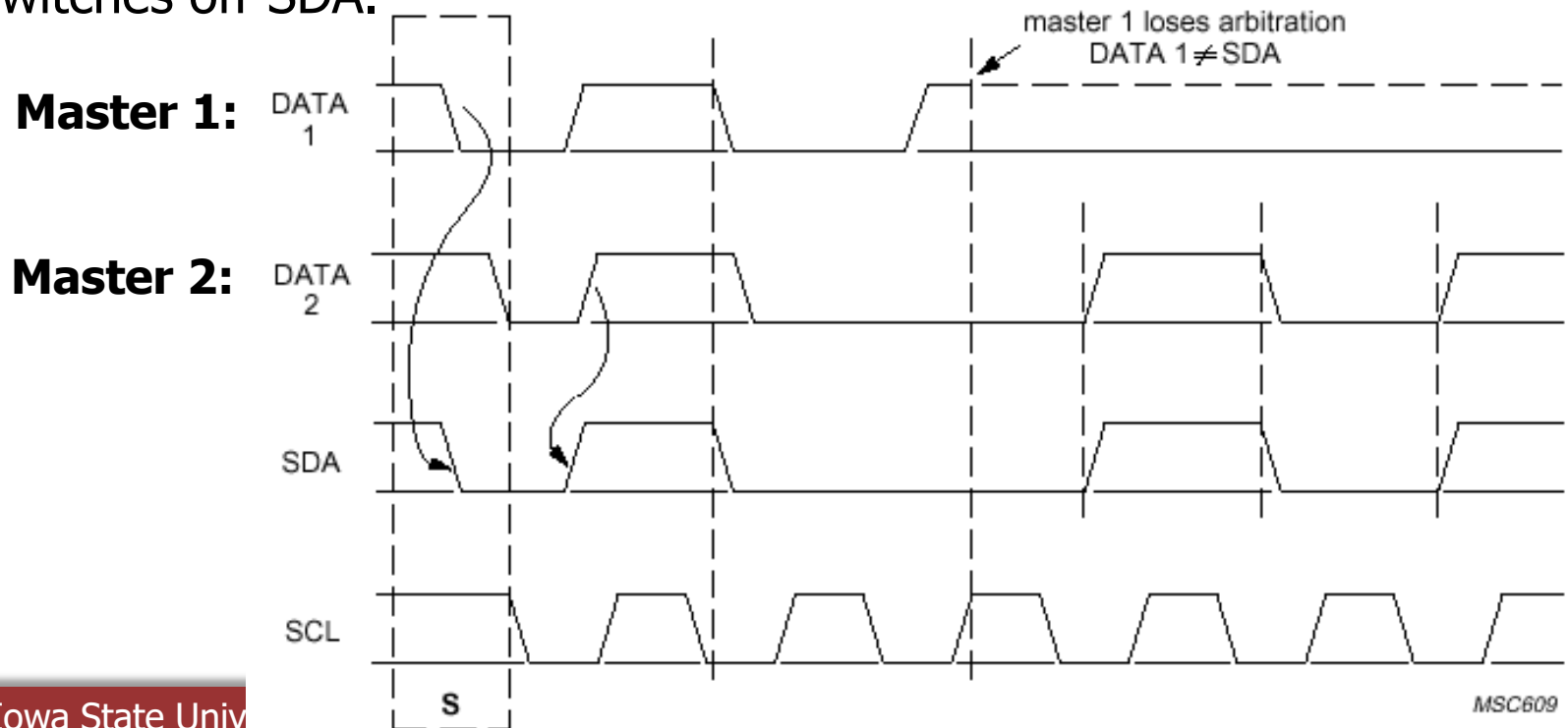
# Slave to Master Data Transfer (Read)

- **Master:**
  - Initiates the data transfer by generating the START condition
  - Then sends slave address “byte” (read/write bit set to 1, i.e., read mode)
- **Slave:**
  - ACK if address provided is for it
  - Sends DATA in 8-bit busts. (note change in Data bus direction: S -> M)
- **Master:**
  - ACKs for each 8-bits slave sends
  - STOP condition is generated by the master (sends not-ACK before generating the STOP)



# Multi-Master Arbitration

- If more than one device can be a master, then an arbitration mechanism is needed to choose the master that takes control of the bus
- Arbitration takes place on SDA, while SCL is HIGH
  - First Master to transmit a HIGH level, while another is driving LOW loses.
  - **Example:** Master 1 (M1), lose arbitration during bit-3 of Slave address.
    - During bit-3 of Slave address, the SDA value observed by M1 does not match what it expects, since M2 is transmitting a LOW level that switches off SDA.



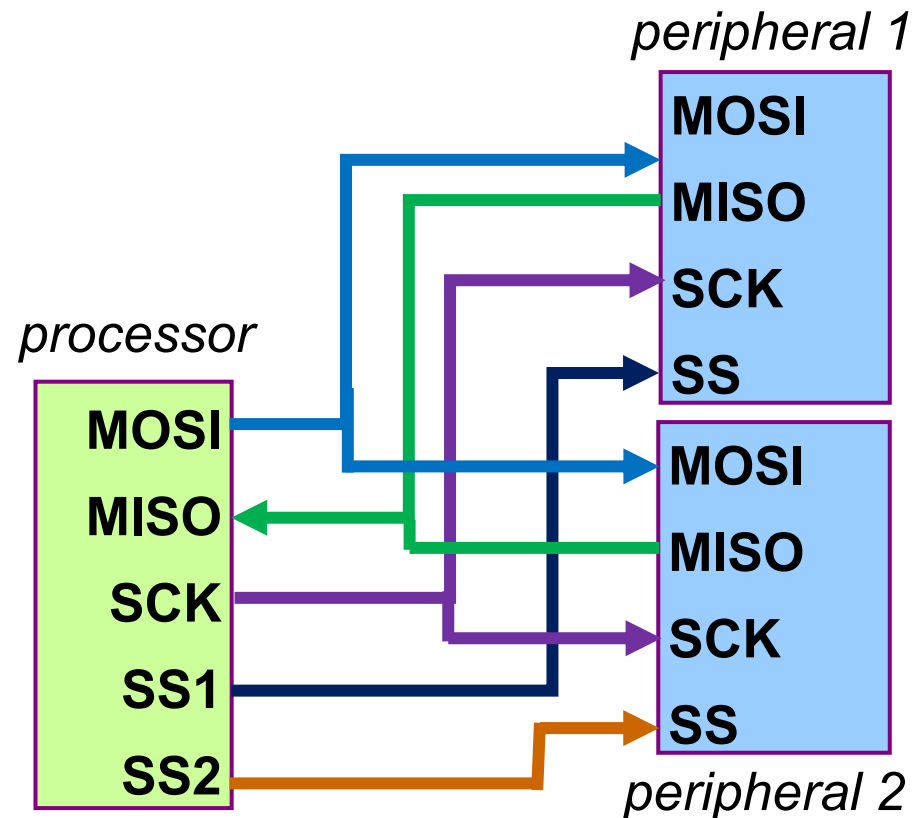
# I<sup>2</sup>C Summary

Protocol	Expandable	Multi-Master	Duplex	Robustness	Speed	Flow Control	Protocol Overhead
UART (2-wire)	No	No	Full	(Parity bit)	~100Kbps	No	Start, Stop, (parity)
I2C	Large	Yes	Half	ACK/NACK	S:100Kbps F:400Kbps Hs:3.4Mbps	Yes	Start, 7-bit addr, R/W, ACK/NACK

- What aspect of the I2C protocol limits how many slaves the bus can support?
- What is the maximum data bits per second, if sending 1 byte to a different slave on each transaction.

# Serial Peripheral Interface (SPI)

- Defined by Motorola on the MC68HCxx line of microcontrollers
  - Generally faster than I<sup>2</sup>C, capable of several Mbps
  - Better suited for “data streams”, i.e. ADC converters
- Full duplex, synchronous, serial communication between CPU (Master) and peripheral devices (Slave)
  - Bi-directional
  - Synchronous serial clock
- Signals:
  - SCK: serial clock
  - SS: select from master to slave
  - MOSI: master out slave in
  - MISO: master in slave out



# SPI Signaling

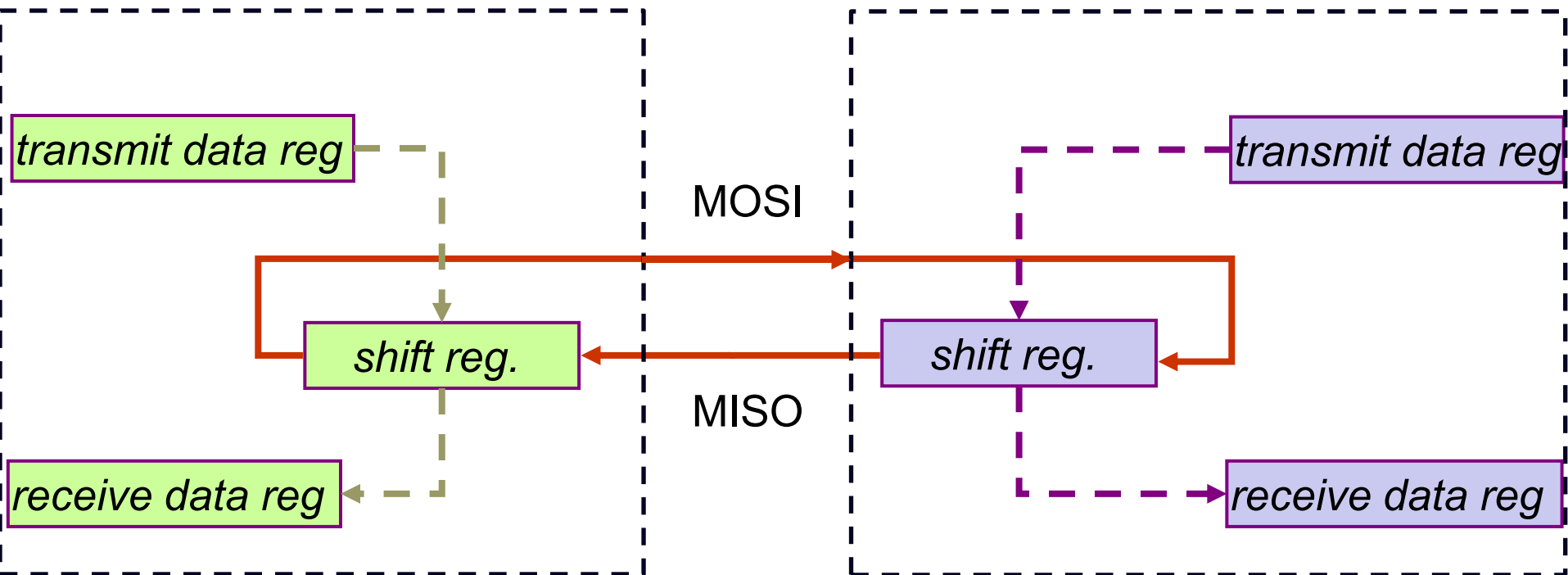
- **Topology:** Single master device communicates with one or more slave devices
- **Slave Select:**
  - Master “activates” slave select input of slave it want to communicate
  - If only one slave device on bus, then SS pin may be fixed to “active”
- **Slave Device Detail:** Most slave devices have tri-state outputs so their MISO signal becomes high impedance (logically disconnected) when the device is not selected
  - Devices without tri-state outputs can not share a SPI bus segment with other slave devices (due to potential for short-circuits, i.e., VCC -> GND)

# SPI Operation

- **Distributed Register:** Data registers in the master and the slave form a distributed register
  - When a data transfer operation is performed, this distributed register is serially shifted by the SCK clock from the master
  - Burst Transfer: shift a burst of bits

## Master

## Slave

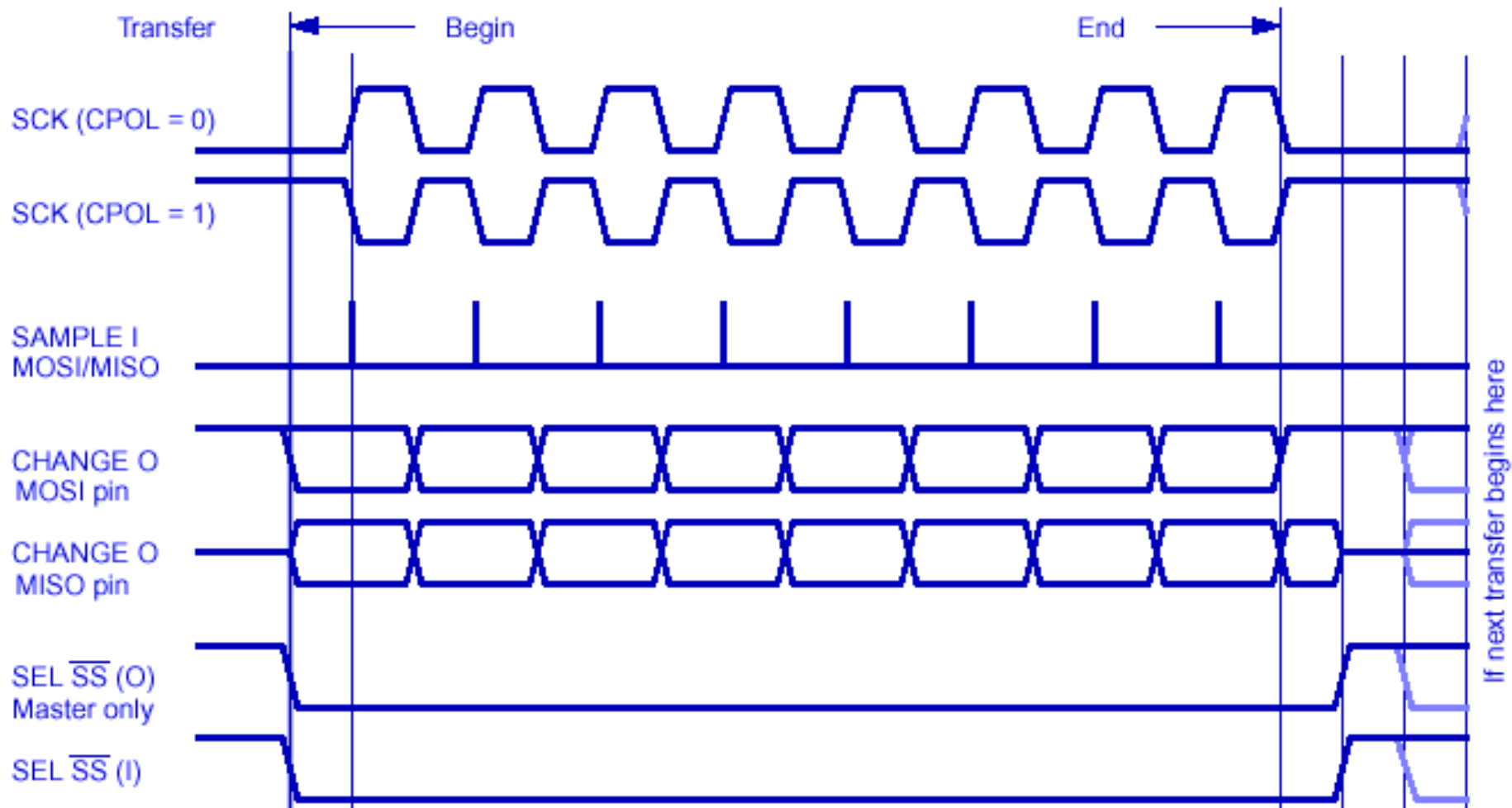


# Data Transmission

- **Configure Clock Speed:** To begin a communication, the bus master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are commonly in the range of 10 kHz–100 MHz
- **Select Slave Device:** The master then transmits the logic 0 for the desired chip over the chip select line. A logic 0 is transmitted because the chip select line is **active low**, meaning its off state is a logic 1; on is asserted with a logic 0
- **Clock Data:** If a waiting period is required (such as for analog-to-digital conversion), then the master must wait for at least that period of time before starting to issue clock cycles

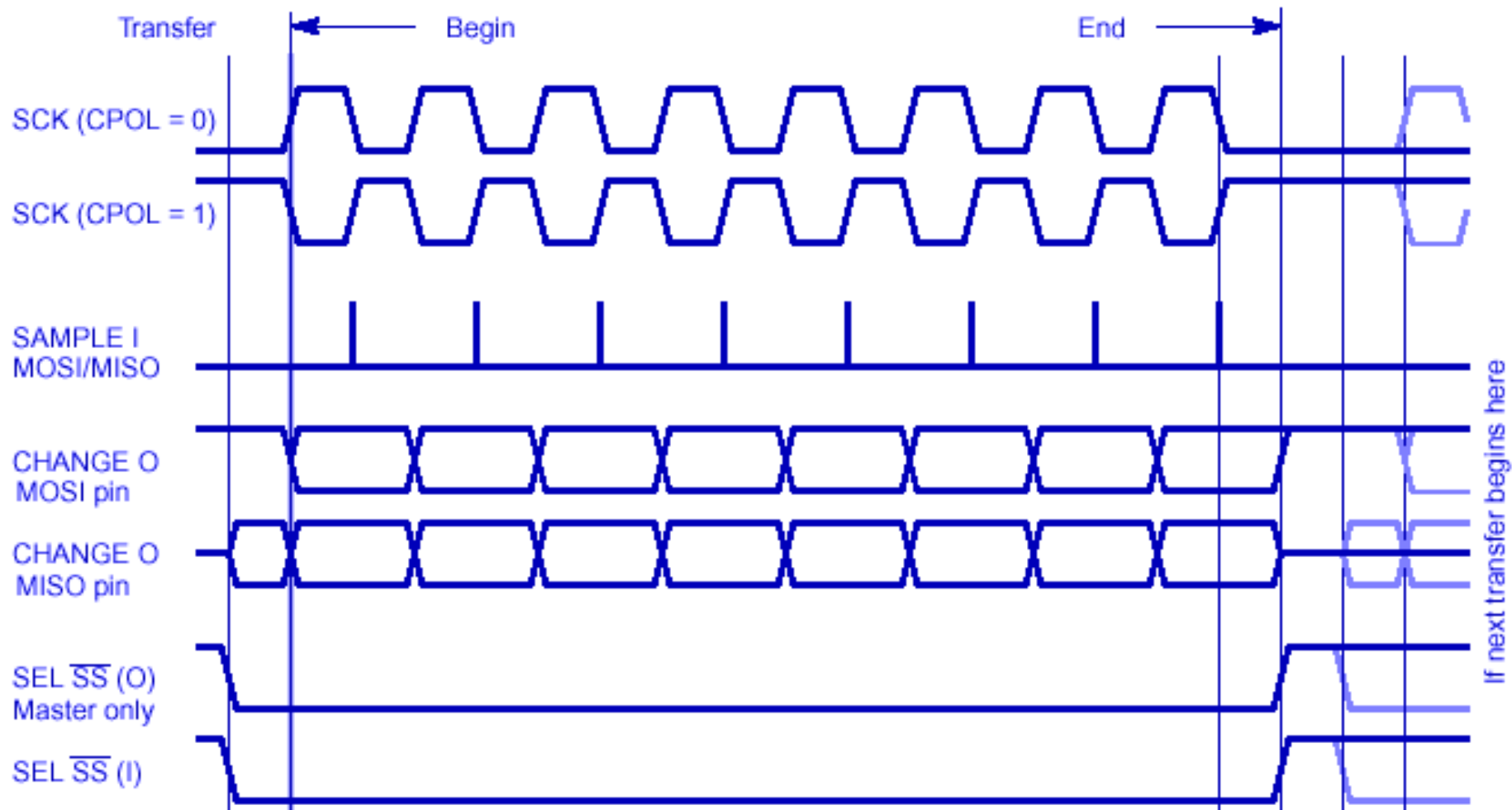
# CPOL and CPHA (Polarity and Phase)

- Two phases and two polarities (four clocking modes)
  - **CPHA=0** – The first edge on the SCK line is used to clock the first data bit (i.e., the first bit of the data must be ready when selected)
  - **CPHA=1** – The second SCK edge is used to clock the first data bit



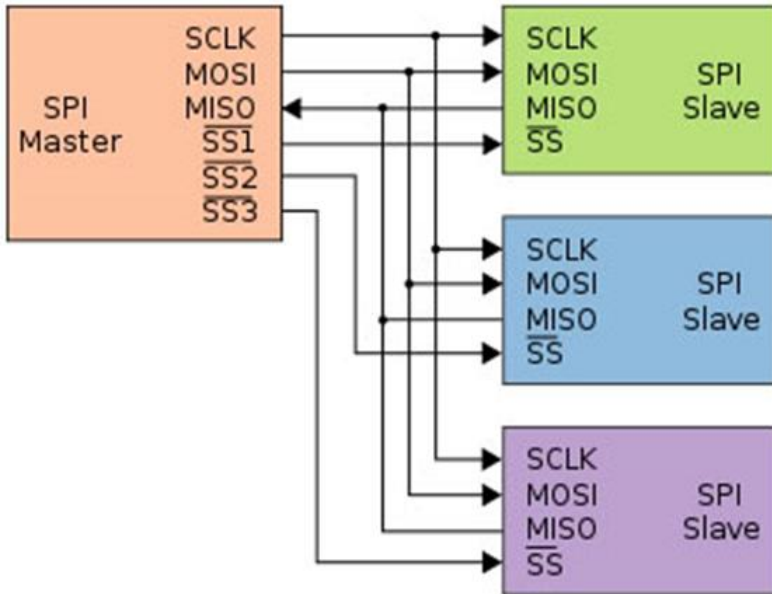
# CPOL and CPHA (Polarity and Phase)

- Two phases and two polarities (four clocking modes)
  - CPHA=0 – The first edge on the SCK line is used to clock the first data bit (i.e., the first bit of the data must be ready when selected)
  - CPHA=1** – The second SCK edge is used to clock the first data bit

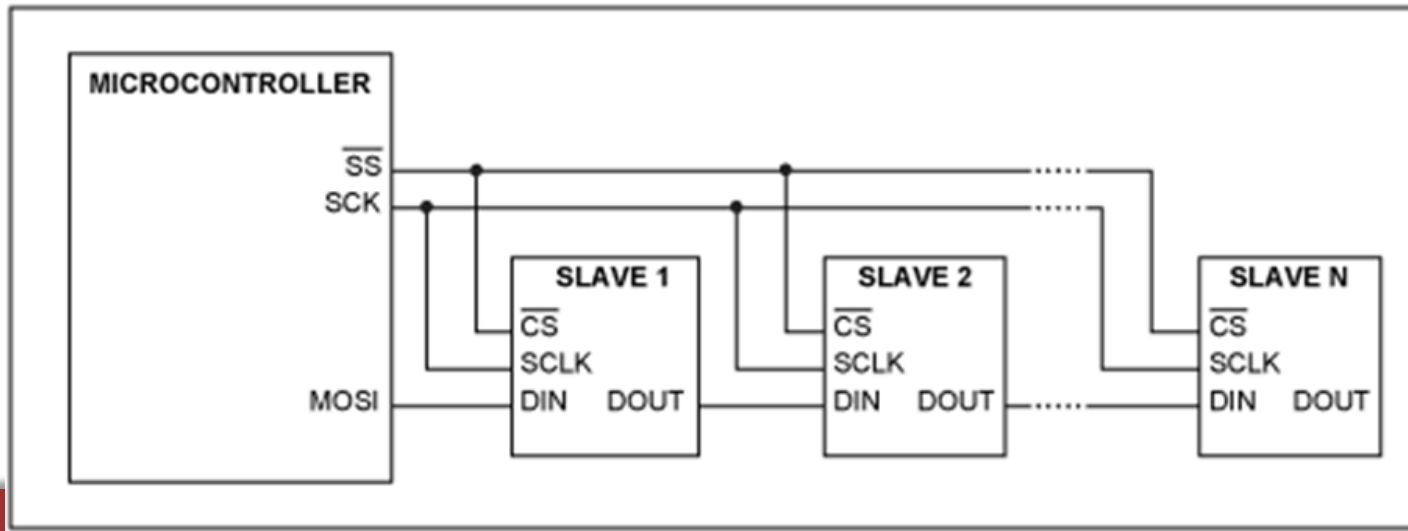


# Bus Configuration Modes

- Master with multiple independent slaves:



- Master with daisy-chained slaves:



# SPI Summary

Protocol	Expandable	Multi-Master	Duplex	Robustness	Speed	Flow Control	Protocol Overhead
UART (2-wire)	No	No	Full	(Parity bit)	~100Kbps	No	Start, Stop, (parity)
I2C	Large	Yes	Half	ACK/NACK	S:100Kbps F:400Kbps Hs:3.4Mbps	Yes	Start, 7-bit addr, R/W, ACK/NACK
SPI	At cost of extra wire per slave	No	Full	None	8Mbps+	No	None

- What aspect of SPI limits how many slaves the bus can support?
- What is the maximum data bits per second, if sending 1 byte to a different slave on each transaction?

# SPI Summary (cont)

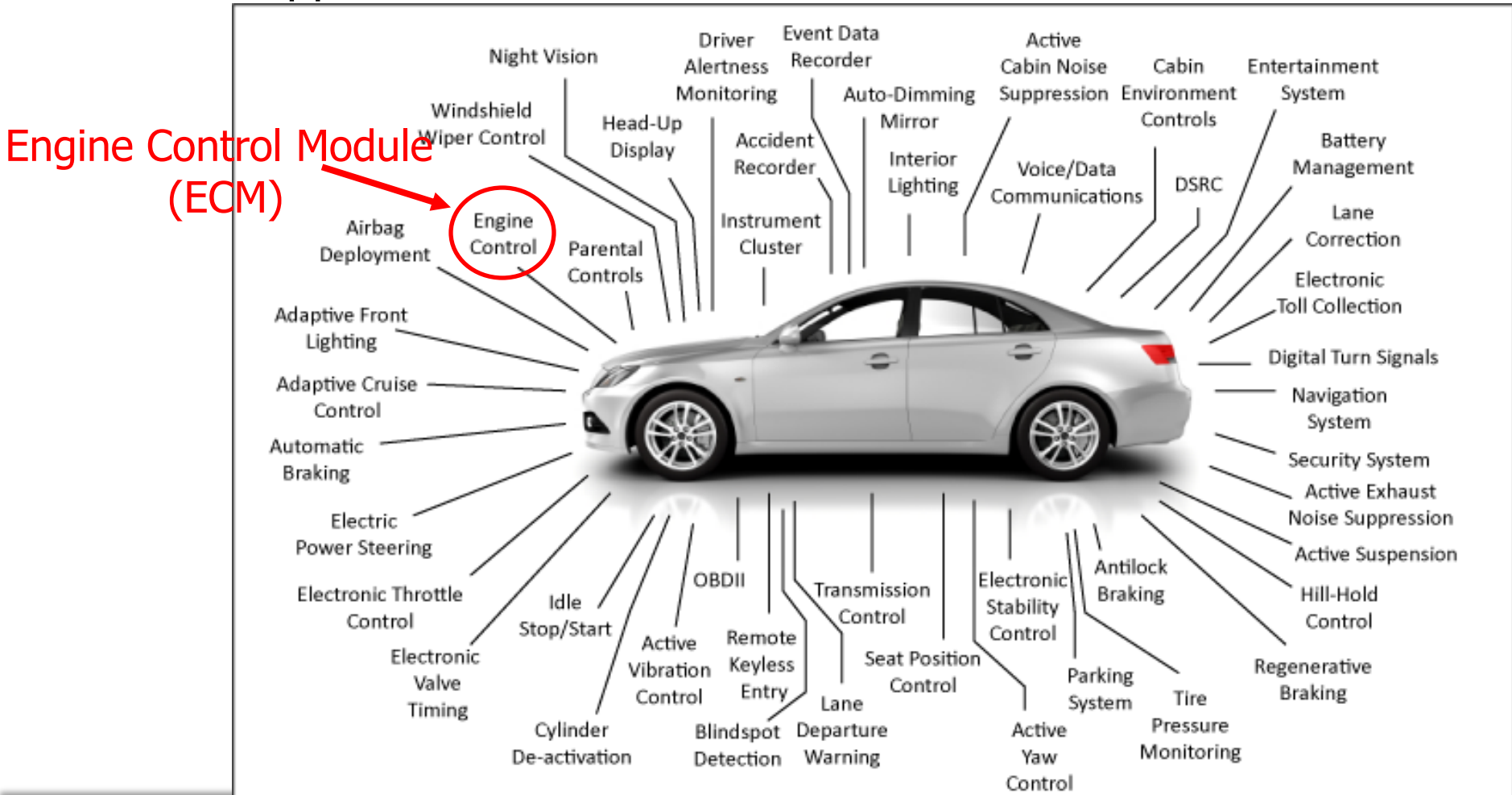
- Pros
  - Fast for point-to-point connections
  - Easily allows streaming/constant data inflow
  - No addressing in protocol, so it's simple to implement
  - Broadly supported
- Cons
  - Slave select/chip select makes multiple slaves more complex
  - No acknowledgement (can't tell if clocking in garbage)
  - No inherent arbitration
  - No flow control (must know slave speed)

# Vehicles as Networks

- 1/3 of cost of car/airplane is electronics/avionics
- Dozens of microprocessors are used throughout the vehicle
- Network applications:
  - Vehicle control
  - Instrumentation
  - Communication
  - Passenger entertainment systems

# Vehicles as Networks

- 1/3 of cost of car/airplane is electronics/avionics
- Dozens of microprocessors are used throughout the vehicle
- Network applications:



# Controller Area Network (CAN)

- The Controller Area Network is a fast serial bus designed to provide a link between sensors and actuators this is:
  - Efficient
  - Reliable
  - Economical
- Introduced by Bosch in 1980s
- Uses a twisted pair cable to communicate at speeds up to 1Mbit/s (max) with up to 131 feet (40m) of total cabling
  - Differential signal used to improve noise immunity
- Originally developed to simplify the wiring of automobile components
  - Each component attaches to the bus using a **Electronic Control Unit (ECU)**, with the Engine ECU (or called Engine Control Module) playing the most central role.
- CAN (and other fieldbus standards) is now used in machine and factory automation products as well
- OSI - Physical and Data link layers

# CAN Bus Physical Layer

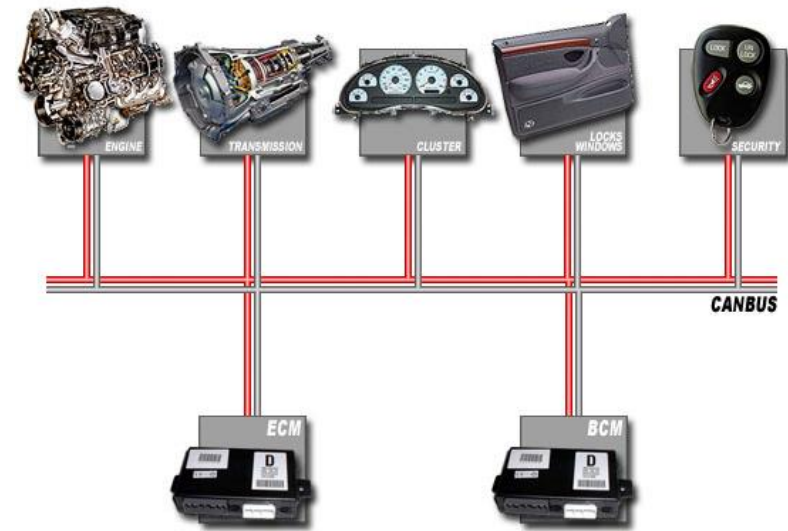
- Physical medium – two wires terminated at both ends by resistors.
  - Differential signal: improves noise immunity
  - Benefits:
    - Reduced weight, Reduced cost
    - Fewer wires = Increased reliability

Pre-CAN Conventional multi-wire looms

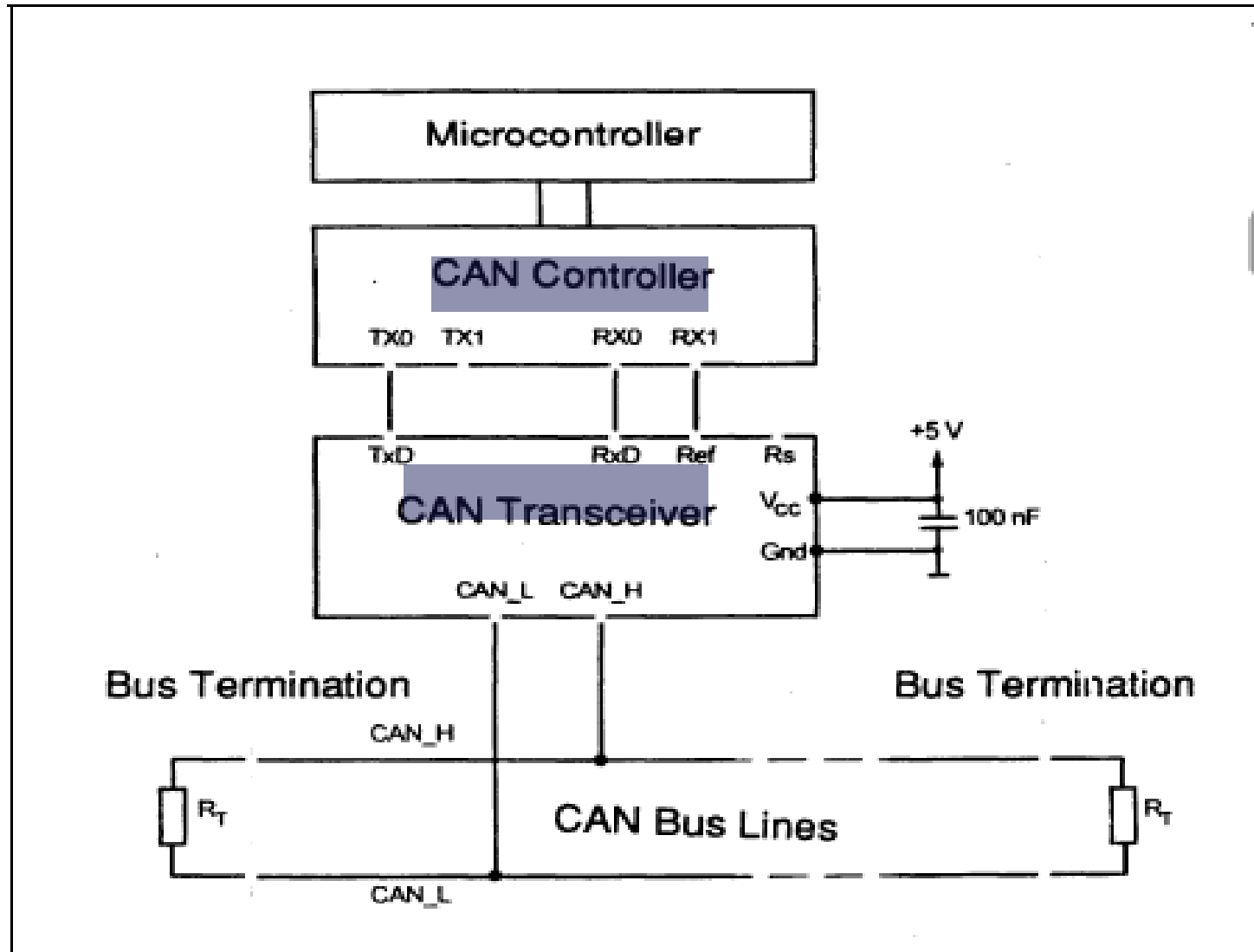


CAN bus network

vs.



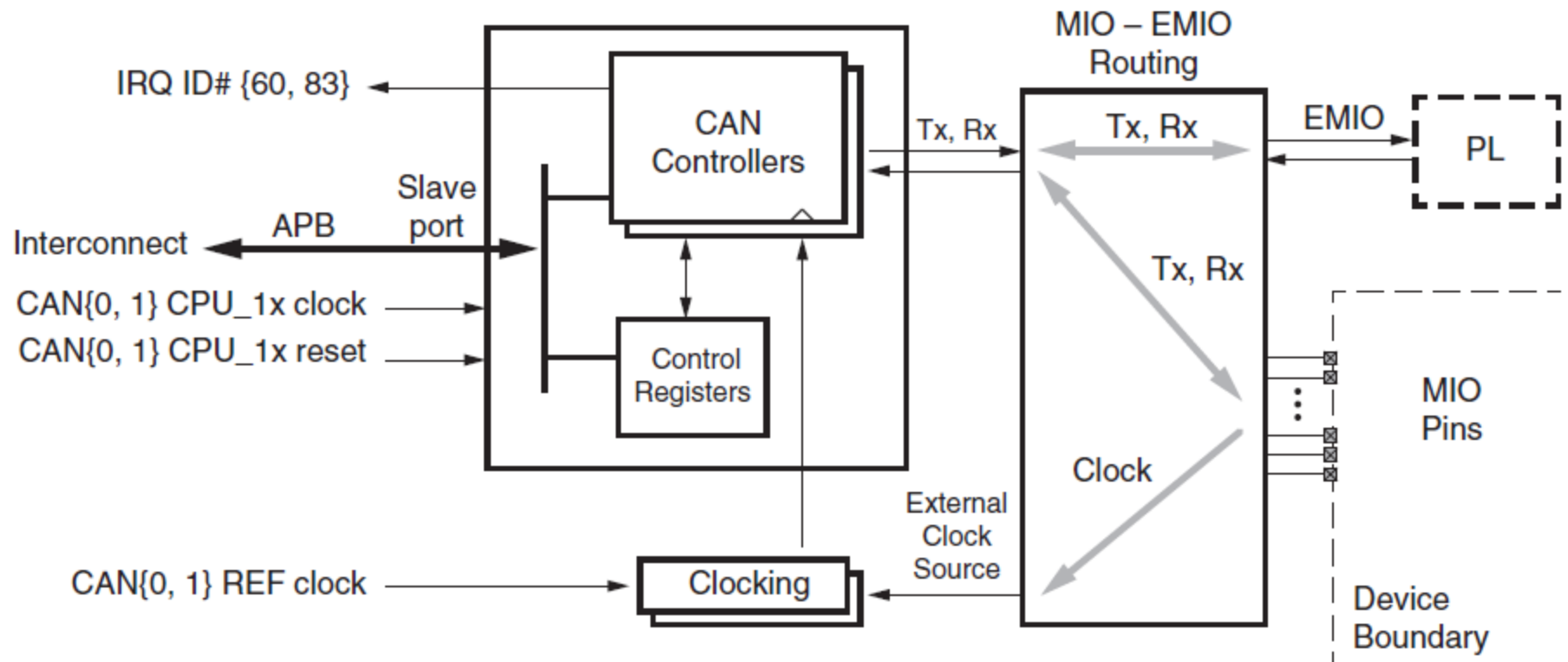
# CAN Physical Layer (cont.)



*Physical CAN Connection according to ISO 11898*



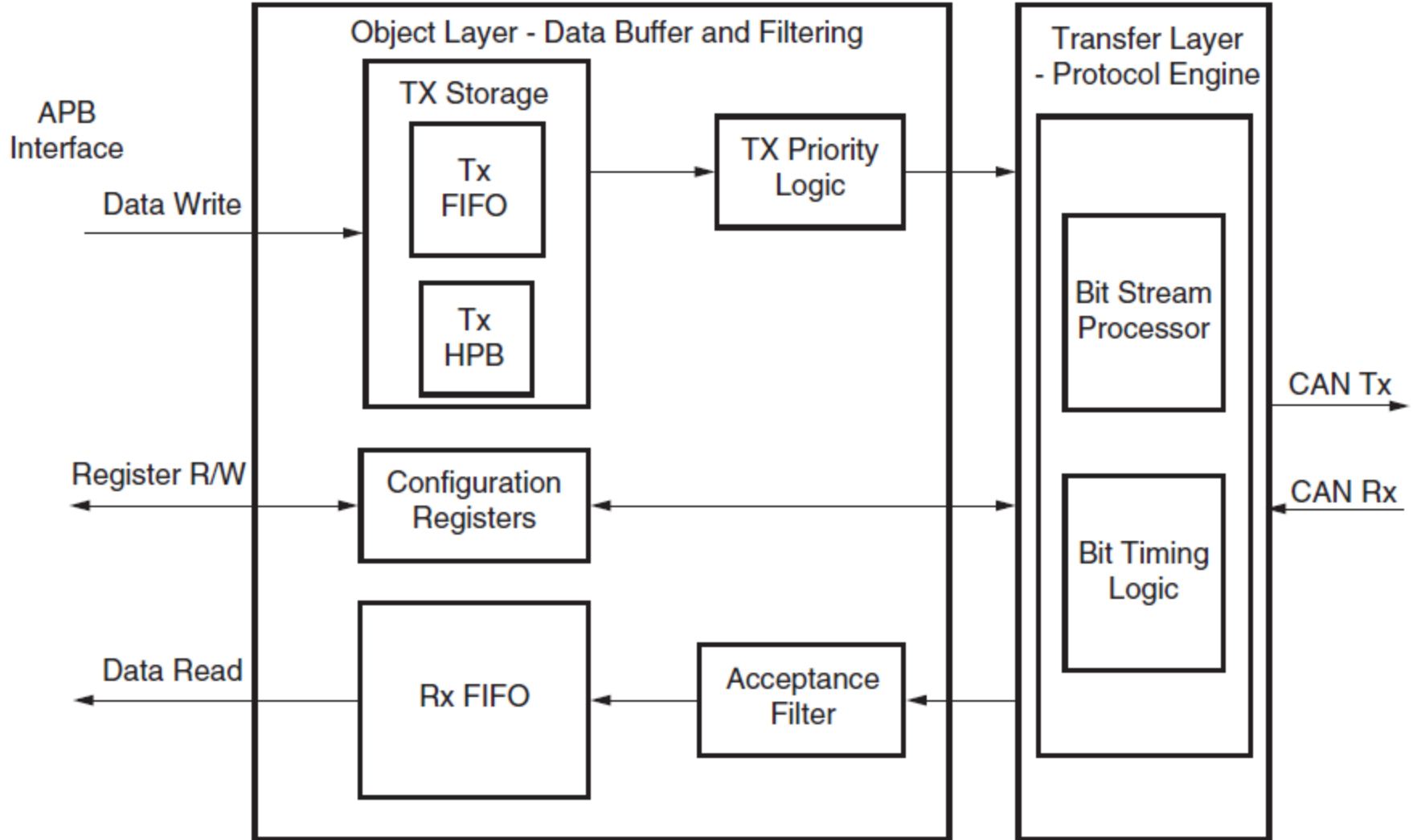
# CAN: Zynq FPGA (from Data Sheet)



UG585\_c18\_01\_071612

Figure 18-1: CAN Controller System Viewpoint

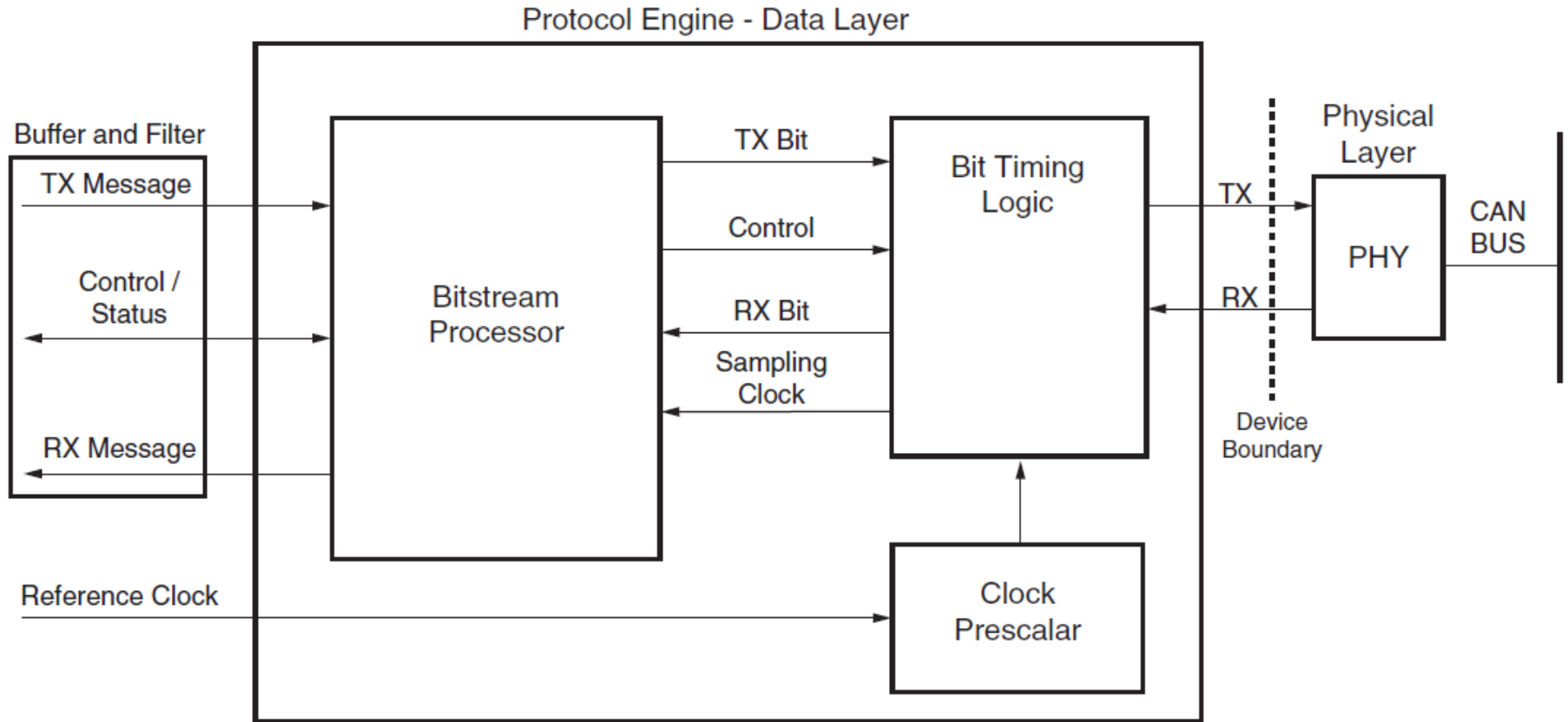
# CAN: Zynq FPGA (from Data Sheet)



UG585\_c18\_02\_021213

Figure 18-2: CAN Controller Block Diagram

# CAN: Zynq FPGA (from Data Sheet)

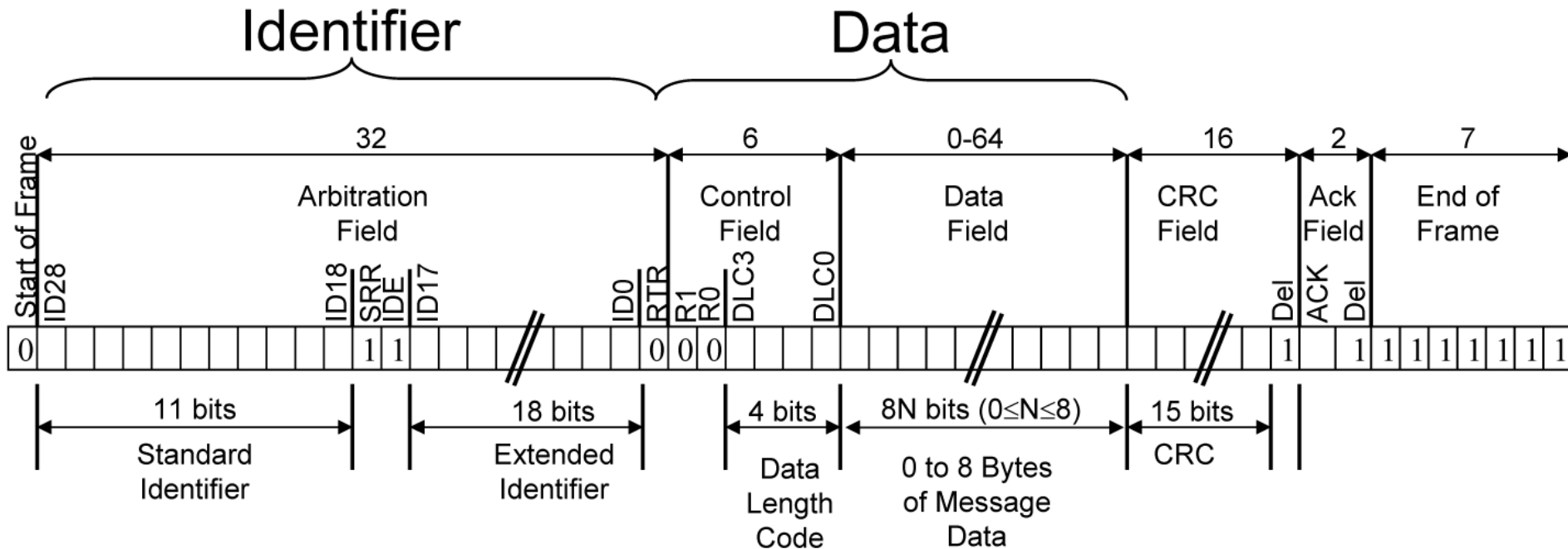


UG585\_c18\_03\_101012

Figure 18-5: CAN Protocol Engine

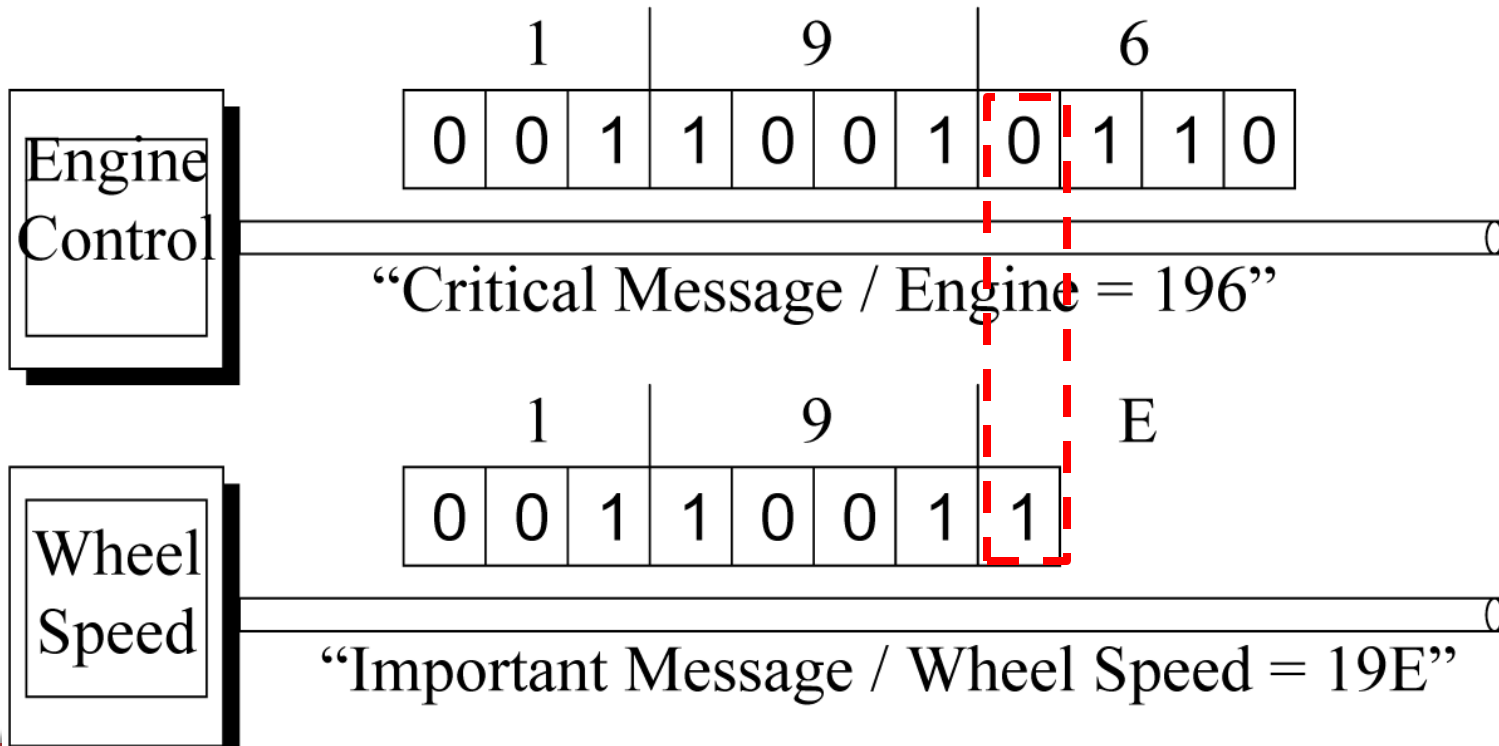
# CAN Message Format

- Each message has an ID, Data and other overhead.
- Data – 8 bytes max per CAN packet
- Overhead – start, end, ID, CRC, ACK



# Bus Arbitration

- Message importance is encoded in message ID
  - Lower value = More important
- As a node transmits each bit, it verifies that it sees the same bit value on the bus that it transmitted
- A "0" on the bus wins over a "1" on the bus
- Losing node stops transmitting, winner continues



# CAN: Zynq FPGA (from Data Sheet)

Table 18-2: CAN Message Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Message Identifier [IDR]	ID[28:18]									STR/RTR	IDE	ID[17:0]															RTR					
Data Length Code [DLCR]	DLC[3:0]			Reserved											Time Stamp (Rx only, Reserved for Tx)																	
Data Word 1 [DW1R]	DB0[7:0]						DB1[7:0]						DB2[7:0]						DB3[7:0]													
Data Word 2 [DW2R]	DB4[7:0]						DB5[7:0]						DB6[7:0]						DB7[7:0]													

# CAN Summary

Protocol	Expandable	Multi-Master	Duplex	Robustness	Speed	Flow Control	Protocol Overhead
UART (2-wire)	No	No	Full	(Parity bit)	~100Kbps	No	Start, Stop, (parity)
I2C	Large	Yes	Half	ACK/NACK	S:100Kbps F:400Kbps Hs:3.4Mbps	Yes	Start, 7-bit addr, R/W, ACK/NACK
SPI	Extra wire per slave	No	Full	None	8Mbps+	No	None
CAN	Large	Yes	Half	<ul style="list-style-type: none"> <li>• 16-bit CRC</li> <li>• ACK/NACK</li> <li>• Differential signaling</li> </ul>	1Mbps	No (for single frame)	Start, 11-bit ID, 6-bit control, 16-bit CRC, 2-bit ACK, 7-bit EOF

- What aspect of the CAN protocol CAN limits how many slaves the bus can support?
- What is the maximum data bits per second, if sending 1 byte to a different slave on each transaction?

# Interconnect Summary

Protocol	Expandable	Multi-Master	Duplex	Robustness	Speed	Flow Control	Protocol Overhead
UART (2-wire)	No	No	Full	(Parity bit)	~100Kbps	No	Start, Stop, (parity)
I2C	Large	Yes	Half	ACK/NACK	S:100Kbps F:400Kbps Hs:3.4Mbps	Yes	Start, 7-bit addr, R/W, ACK/NACK
SPI	At cost of extra wire per slave	No	Full	None	8Mbps+	No	None
CAN	Large	Yes	Half	16-bit CRC ACK/NACK Differential signaling	1Mbps	No (for single frame)	Start, 11-bit ID, 6-bit control, 16-bit CRC, 2-bit ACK, 7-bit EOF

# Acknowledgments

- These slides are inspired in part by material developed and copyright by:
  - Marilyn Wolf (Georgia Tech)
  - Yann-Hang Lee (Arizona State)
  - Prabal Dutta (Michigan)
  - Manimaran Govindarasu (Iowa State)
  - William Stallings