The I^{2}C Bus

The I²C Bus

- What is the I²C Bus and what is it used for?
- Bus characteristics
- I²C Bus Protocol
- Data Format
- Typical I²C devices
- Example device
- Sample pseudo code

What is I²C

- **The name stands for “Inter - Integrated Circuit Bus”**
- **A Small Area Network connecting ICs and other electronic systems**
- **Originally intended for operation on one single board / PCB**
  - Synchronous Serial Signal
  - Two wires carry information between a number of devices
  - One wire use for the data
  - One wire used for the clock
- **Today, a variety of devices are available with I²C Interfaces**
  - Microcontroller, EEPROM, Real-Timer, interface chips, LCD driver, A/D converter

What is I^{2}C used for?

- **Data transfer between ICs and systems at relatively low rates**
  - “Classic” I^{2}C is rated to 100K bits/second
  - “Fast Mode” devices support up to 400K bits/second
  - A “High Speed Mode” is defined for operation up to 3.4M bits/second
- **Reduces Board Space and Cost By:**
  - Allowing use of ICs with fewer pins and smaller packages
  - Greatly reducing interconnect complexity
  - Allowing digitally controlled components to be located close to their point of use

I²C Bus Characteristics

- **Includes electrical and timing specifications, and an associated bus protocol**
- **Two wire serial data & control bus implemented with the serial data (SDA) and clock (SCL) lines**
  - For reliable operation, a third line is required:
    Common ground
- **Unique start and stop condition**
- **Slave selection protocol uses a 7-Bit slave address**
  - The bus specification allows an extension to 10 bits
- **Bi-directional data transfer**
- **Acknowledgement after each transferred byte**
- **No fixed length of transfer**

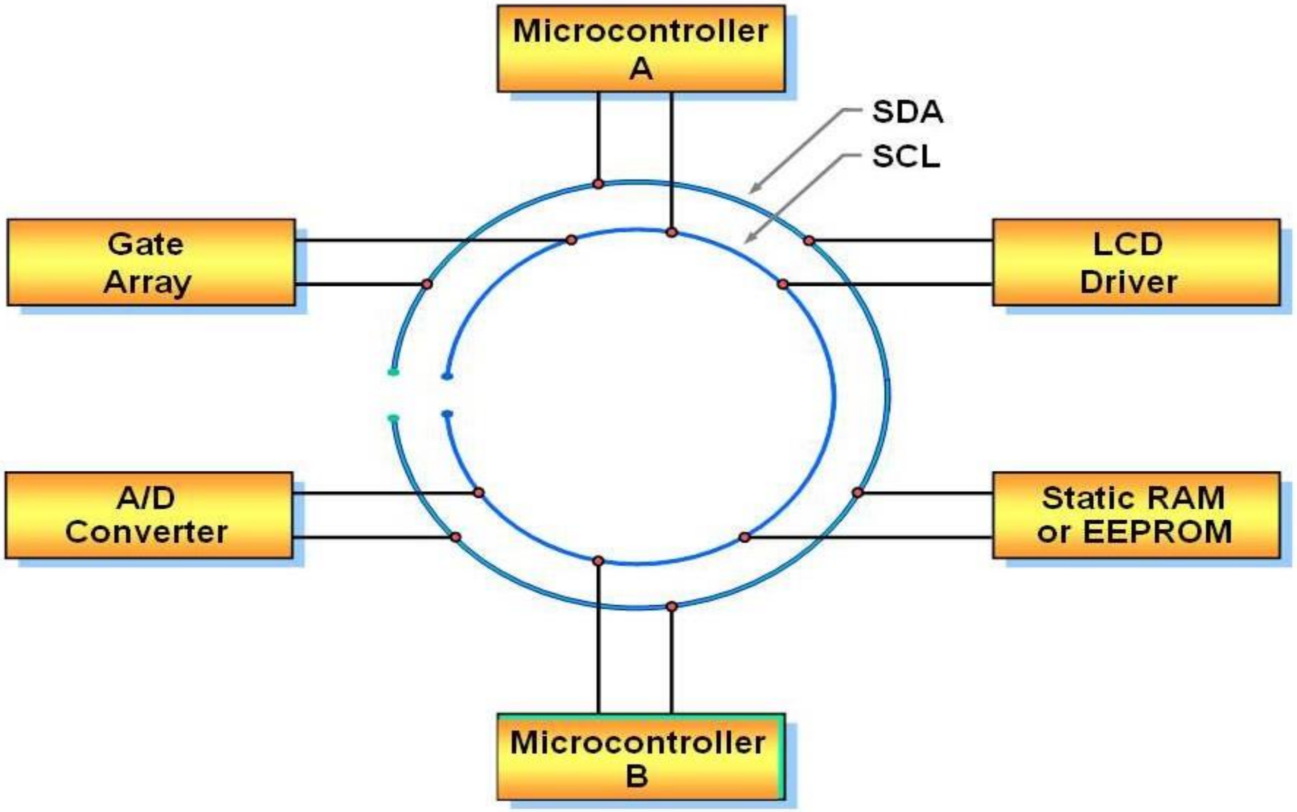
I^{2}C Bus Characteristics (cont'd)

- **True multi-master capability**
  - Clock synchronization
  - Arbitration procedure
- **Transmission speeds up to 100Khz (classic I2C)**
- **Max. line capacitance of 400pF, approximately 4 meters (12 feet)**
- **Allows series resistor for IC protection**
- **Compatible with different IC technologies**

I²C Bus Definitions

- **Master:**
  - Initiates a transfer by generating start and stop conditions
  - Generates the clock
  - Transmits the slave address
  - Determines data transfer direction
- **Slave:**
  - Responds only when addressed
  - Timing is controlled by the clock line

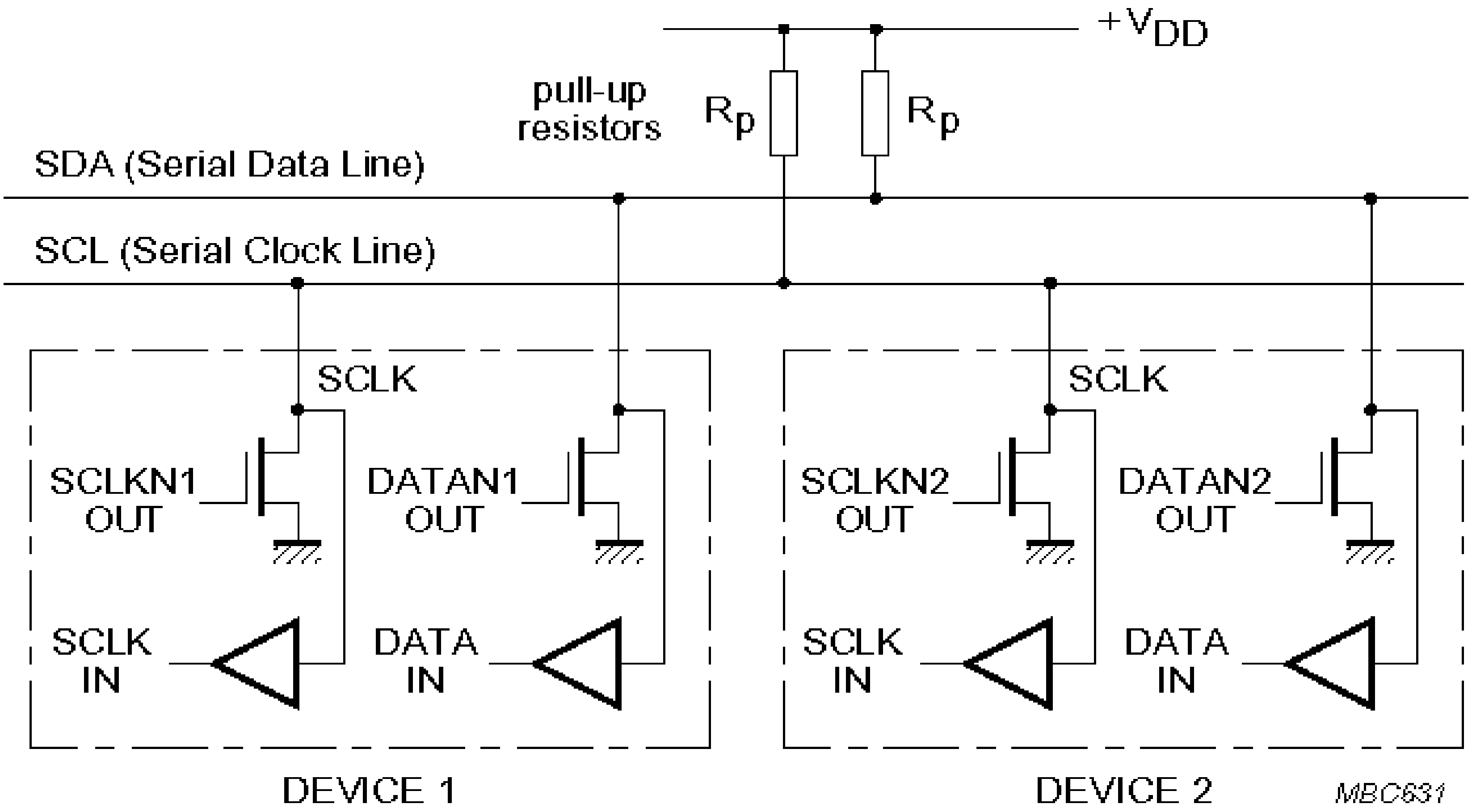
I^{2}C Bus Configuration Example



I^{2}C Hardware Details

- Devices connected to the bus must have an open drain or open collector output for serial clock and data signal
- The device must also be able to sense the logic level on these pins
- All devices have a common ground reference
- The serial clock and data lines are connected to Vdd(typically +5V) through pull up resistors
- At any given moment the I2C bus is:
  - Quiescent (Idle), or
  - in Master transmit mode or
  - in Master receive mode.

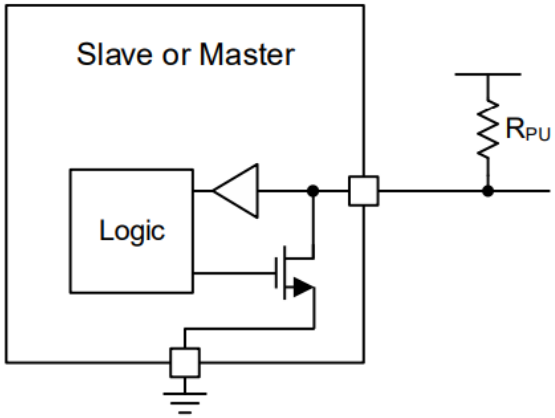
I²C Electrical Aspects



- I²C devices are wire ANDed together.
- If any single node writes a zero, the entire line is zero

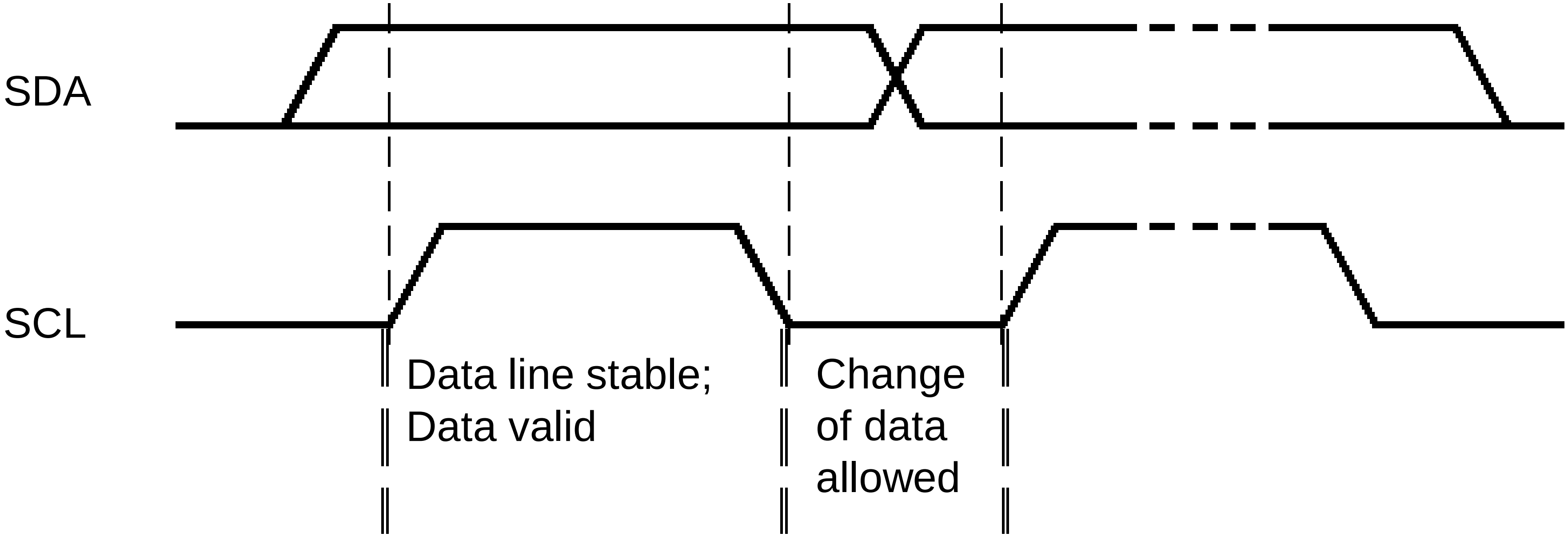
Open-Drain for Bidirectional Communication

- Open-drain is an output type that can either pull a communication bus down to a low voltage (usually ground) or allow it to be pulled up by a pull-up resistor.
- When a device releases the bus, a pull-up resistor (RPU) is responsible for raising the bus voltage to the power rail, preventing conflicts between devices.
- Open-drain ensures that no device can force a high signal on the bus, preventing communication issues like shorts between power and ground caused by conflicting signals.
- In I2C, when a master detects that the bus is low (pulled down by another device), it must halt communication to avoid conflicts, especially in multi-master environments.



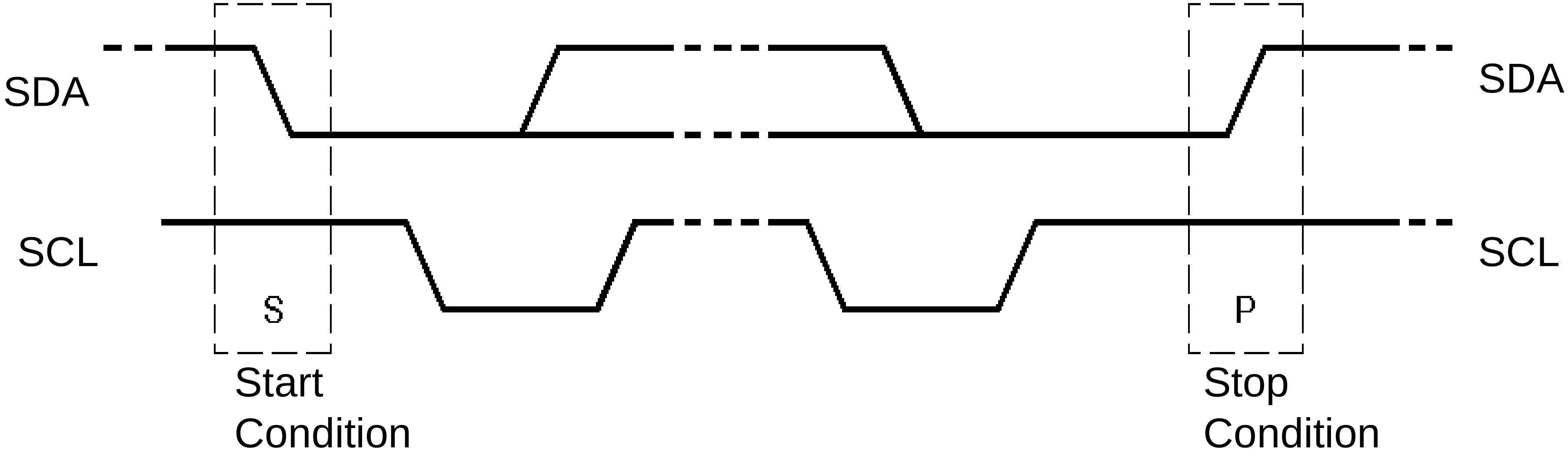
Bit Transfer on the I^{2}C Bus

- In normal data transfer, the data line only changes state when the clock is low



Start and Stop Conditions

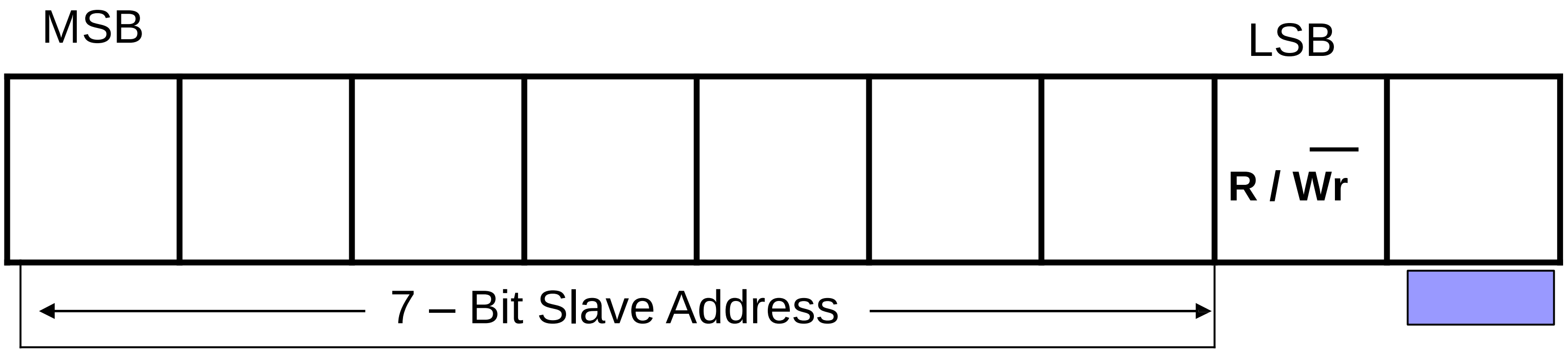
- A transition of the data line while the clock line is high is defined as either a start or a stop condition.
- Both start and stop conditions are generated by the bus master
- The bus is considered busy after a start condition, until a stop condition occurs



I^{2}C Addressing

- **Each node has a unique 7 (or 10) bit address**
- **Peripherals often have fixed and programmable address portions**
- **Addresses starting with 0000 or 1111 have special functions:-**
  - 0000000 Is a General Call Address
  - 0000001 Is a Null (CBUS) Address
  - 1111XXX Address Extension
  - 1111111 Address Extension – Next Bytes are the Actual Address

First Byte in Data Transfer on the I²C Bus



MSB

LSB

R / $\overline{\text{Wr}}$

7 – Bit Slave Address

R/Wr

0 – Slave written to by Master

1 – Slave read by Master

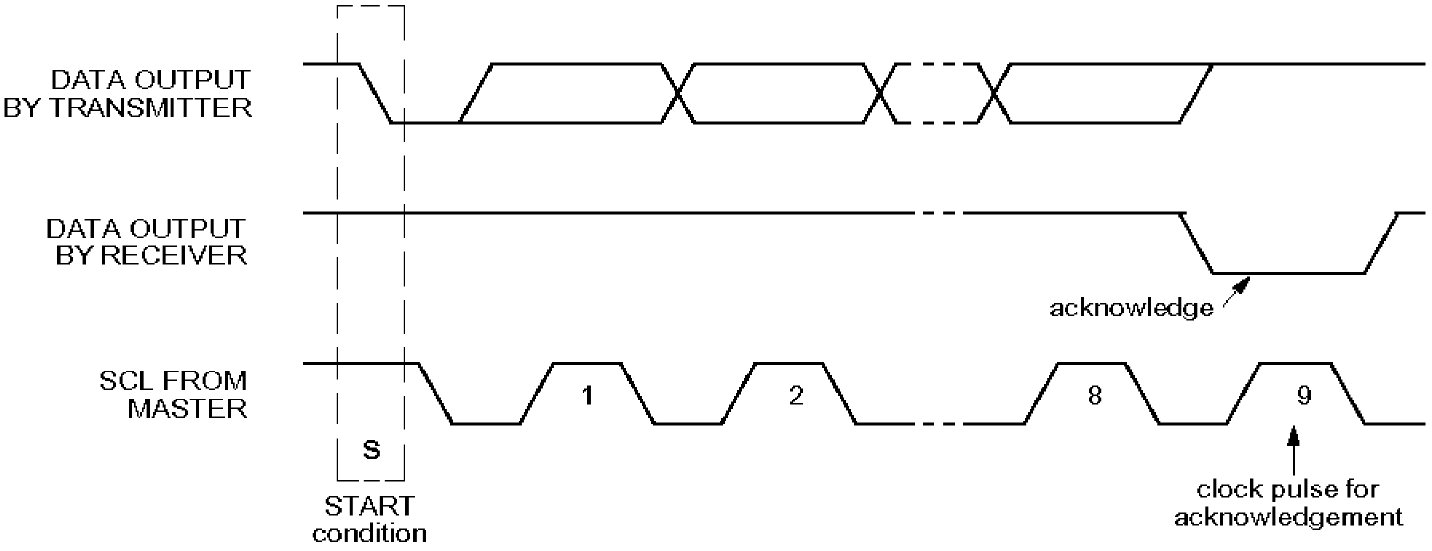
ACK – Generated by the slave whose address has been output.

I²C Bus Connections

- **Masters can be**
  - Transmitter only
  - Transmitter and receiver
- **Slaves can be**
  - Receiver only
  - Receiver and transmitter

Acknowledgements

- Master/slave receivers pull data line low for one clock pulse after reception of a byte
- Master receiver leaves data line high after receipt of the last byte requested
- Slave receiver leaves data line high on the byte following the last byte it can accept

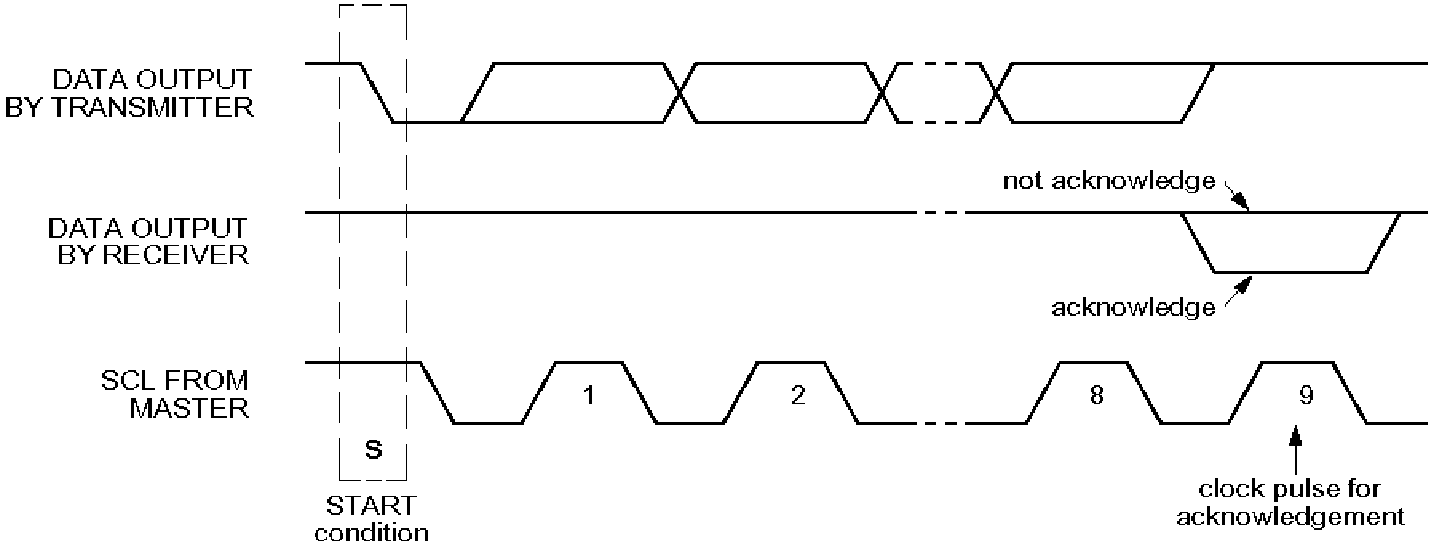


Transmitter releases SDA line during 9th clock pulse.

Acknowledgement from receiver

Negative Acknowledge

- Receiver leaves data line high for one clock pulse after reception of a byte

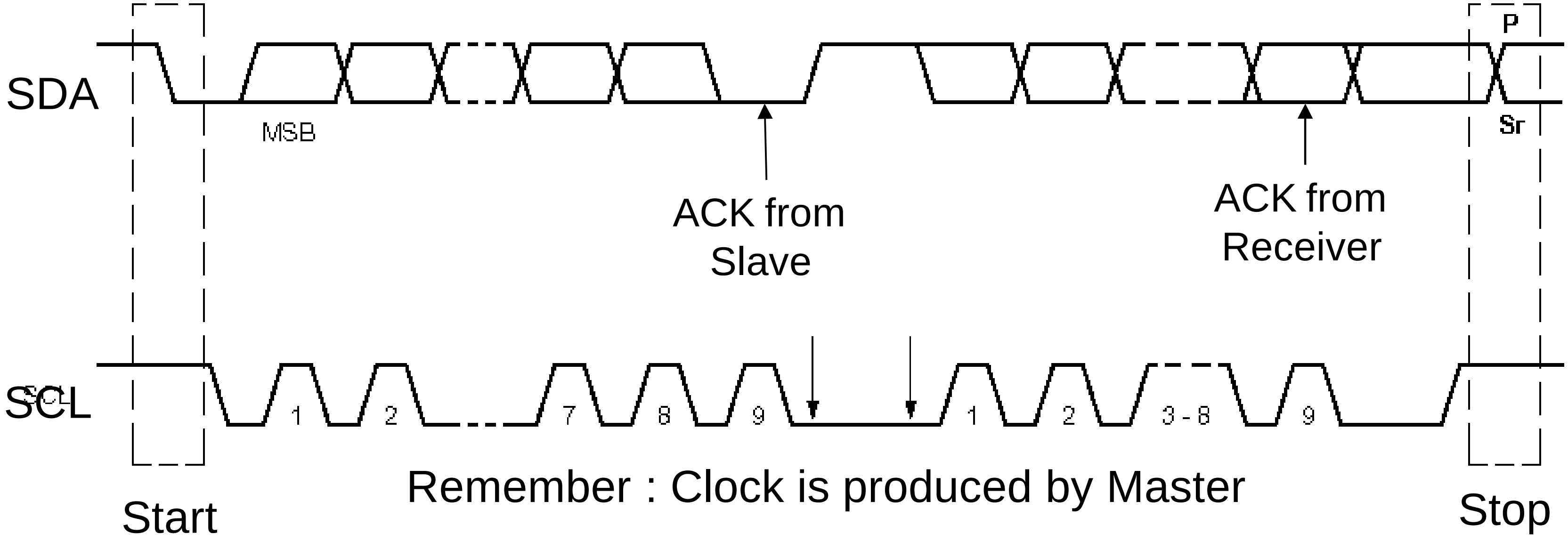


Transmitter releases SDA line during 9th clock pulse.

Not acknowledgement (NACK) from receiver

Data Transfer on the I2C Bus

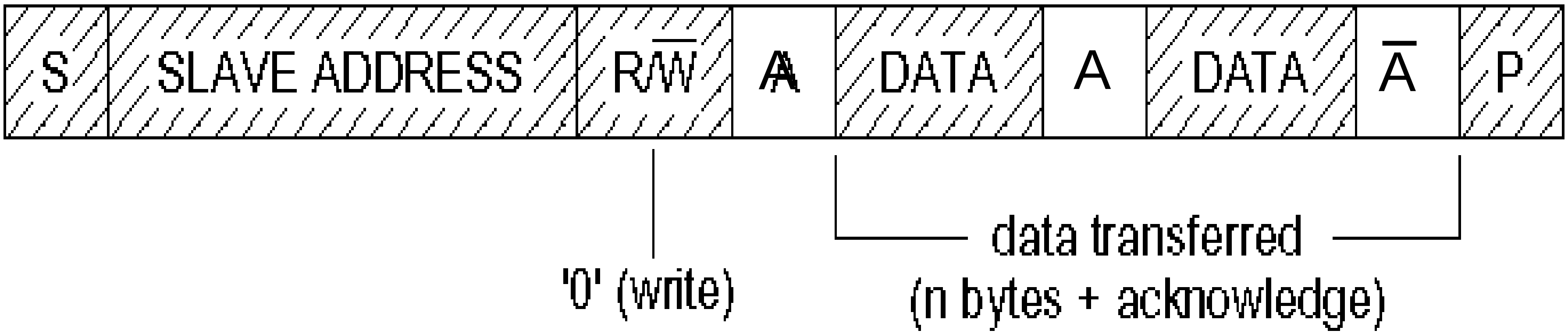
- **Start Condition**
- **Slave address + R/W**
  - Slave acknowledges with ACK
- **All data bytes**
  - Each followed by ACK
- **Stop Condition**



Remember : Clock is produced by Master

Data Formats

- **Master writing to a Slave**



S	SLAVE ADDRESS	R/$\overline{W}$	A	DATA	A	DATA	$\bar{A}$	P

'0' (write)

data transferred (n bytes + acknowledge)

▨ from master to slave

□ from slave to master

A = acknowledge (SDA LOW)

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

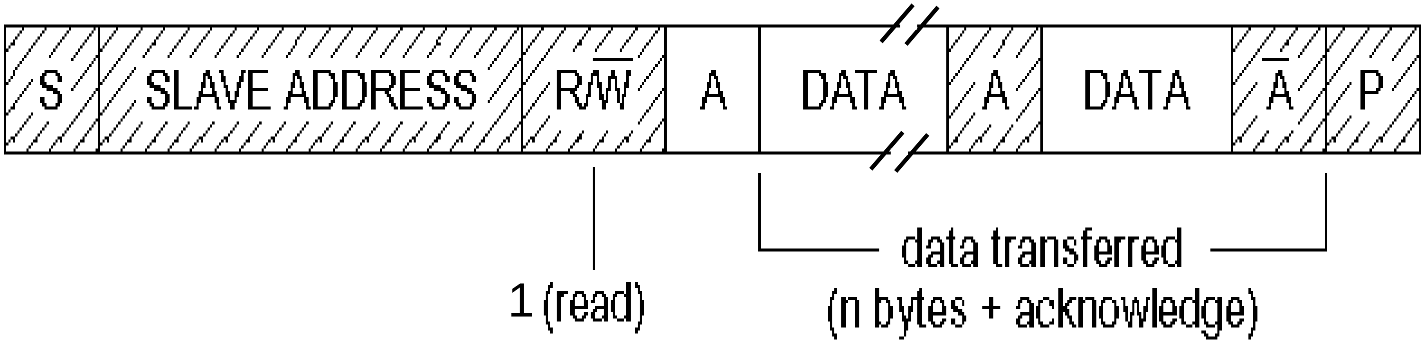
S = START condition

P = STOP condition

Data Formats Cont’d.

- **Master reading from a Slave** :

Master is Receiver of data and Slave is Transmitter of data.



▨ from master to slave

□ from slave to master

A = acknowledge (SDA LOW)

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

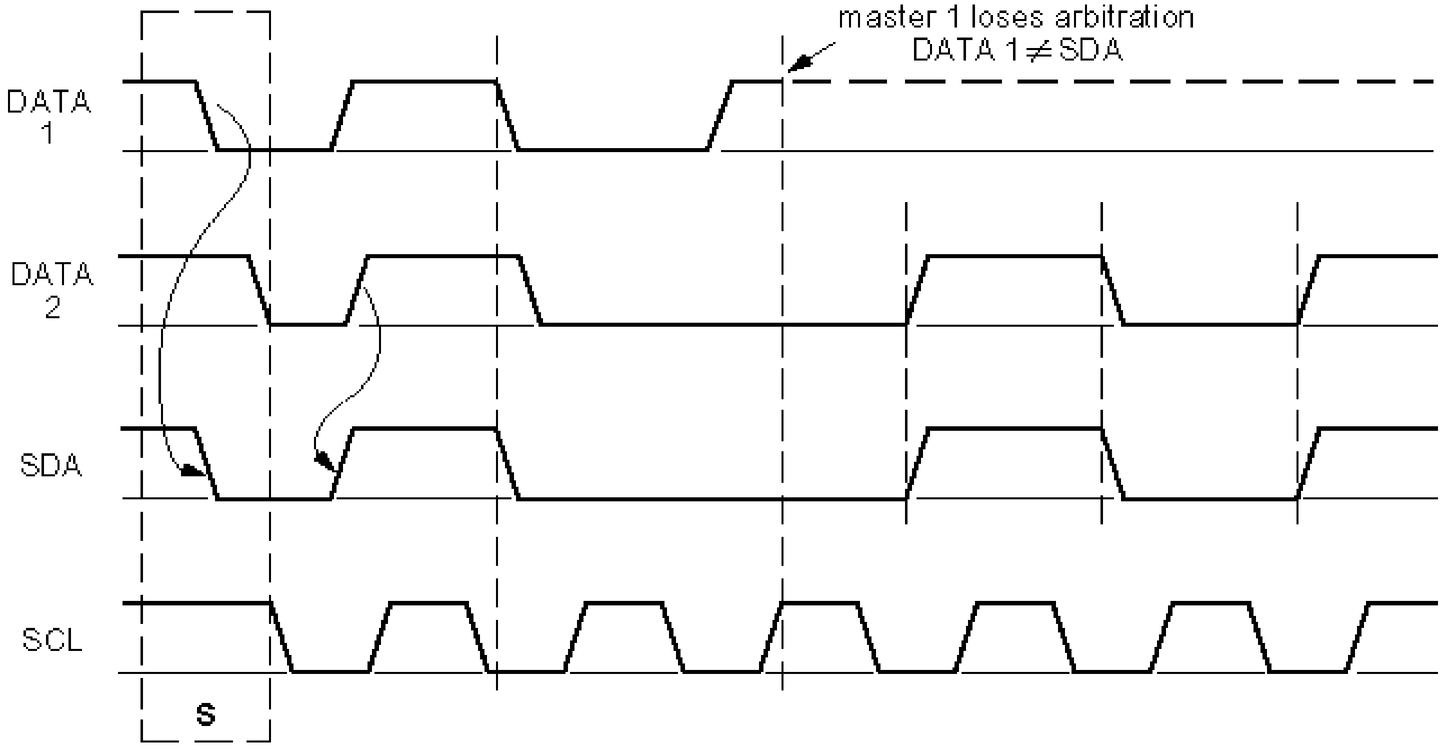
S = START condition

P = STOP condition

Multi-master I^{2}C Systems

- **Multimaster situations require two additional features of the I^{2}C protocol**
- **Arbitration:**
  - Arbitration is the procedure by which competing masters decide final control of the bus
  - I2C arbitration does not corrupt the data transmitted by the prevailing master
  - Arbitration is performed bit by bit until it is uniquely resolved
  - Arbitration is lost by a master when it attempts to assert a high on the data line and fails

Arbitration Between Two Masters



- As the data line is like a wired AND, a ZERO address bit overwrites a ONE
- The node detecting that it has been overwritten stops transmitting and waits for the Stop Condition before it retries to arbitrate the bus

Error Checking

- **I^{2}C defines the basic protocol and timing**
  - Protocol errors are typically flagged by the interface
  - Timing errors may be flagged, or in some cases could be interpreted as a different bus event
- **Glitches (if not filtered out) could potentially cause:**
  - Apparent extra clocks
  - Incorrect data
  - “Locked” bus
- **Microprocessors communicating with each other can add a checksum or equivalent**

Bus Recovery

- **An I^{2}C bus can be "locked" when:**
  - A Master and a Slave get out of synch
  - A Stop is omitted or missed (possibly due to noise)
  - Any device on the bus holds one of the lines low improperly, for any reason
  - A shorted bus line
- **If SCL can be driven, the Master may send extra clocks until SDA goes high, then send a Stop.**
- **If SCL is stuck low, only the device driving it can correct the problem.**

Available I²C Devices

- **Analog to Digital Converters (A/D, D/A):** MMI functions, battery & converters, temperature monitoring, control systems
- **Bus Controller:** Telecom, consumer electronics, automotive, Hi-Fi systems, PCs, servers
- **Bus Repeater, Hub & Expander:** Telecom, consumer electronics, automotive, Hi-Fi systems, PCs, servers
- **Real Time Clock (RTC)/Calendar:** Telecom, EDP, consumer electronics, clocks, automotive, Hi-Fi systems, FAX, PCs, terminals
- **DIP Switch:** Telecom, automotive, servers, battery & converters, control systems
- **LCD/LED Display Drivers:** Telecom, automotive instrument driver clusters, metering systems, POS terminals, portable items, consumer electronics

Available I^{2}C Devices

- **General Purpose Input/Output (GPIO) Expanders and LED Display Control:** Servers, keyboard interface, expanders, mouse track balls, remote transducers, LED drive, interrupt output, drive relays, switch input
- **Multiplexer & Switch:** Telecom, automotive instrument driver clusters, metering systems, POS terminals, portable items, consumer electronics
- **Serial RAM/ EEPROM:** Scratch pad/ parameter storage
- **Temperature & Voltage Monitor:** Telecom, metering systems, portable items, PC, servers
- **Voltage Level Translator:** Telecom, servers, PC, portable items, consumer electronics

Thank you