Inter-Integrated Circuit Bus

(I^2C Serial Bus)

http://www.i2c-bus.org/

I^2C OVERVIEW

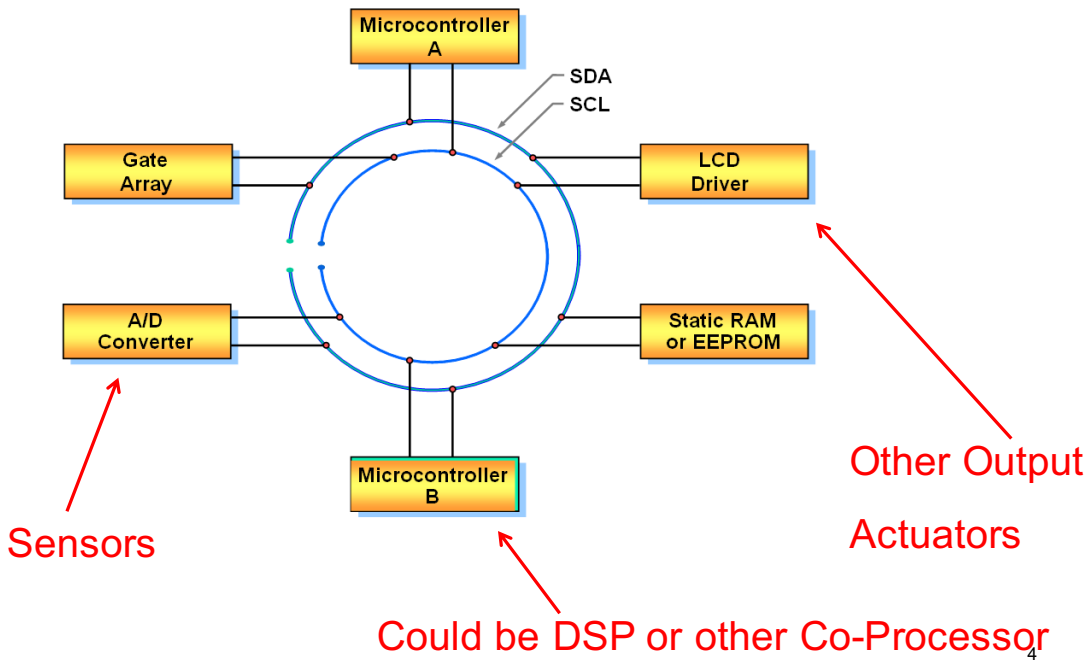
- The name stands for “Inter - Integrated Circuit Bus” (Developed by Philips in the early 1980s)
  - physical layer specification (see class webpage)
- A Small Area Network connecting ICs and other electronic systems
- Originally intended for operation on one single printed circuit board (PCB)
  - Synchronous Serial Signal
  - Two wires carry information between a number of devices
  - One wire use for the data: SDA (Serial DAta)
  - One wire used for the clock : SCL (Serial CLock)

I²C USAGE

- Examples of I²C-compatible devices found in embedded systems include:
  - Microcontrollers
  - EEPROMs
  - Real-Timers
  - interface chips
  - LCD drivers
  - Data A/D converters (A/D and D/A)
- Can be Found in the following:

– interconnecting subcircuits within a core
– interconnecting cores in SoC
– interconnecting ICs on a board
– interconnecting boards on a backplane
– in some cases interconnecting rack mounted chassis

I²C Bus Embedded System Example



I²C Characteristics

- 3 levels of data transfer rate
  - 100kbps in standard mode
  - 400kbps in Fast mode
  - 3.4 Mbps in high-speed mode
- Standard is open (not proprietary)
- Requires only 2 Signal Conductors (higher reliability uses dedicated ground)
- Many Supporting Chips and Cores Available
- Other Popular Serial Busses:
  - EIA-232C (commonly referred to by its older name RS-232)
  - Universal Serial Bus (USB)
  - Firewire (IEEE Standard 1394)

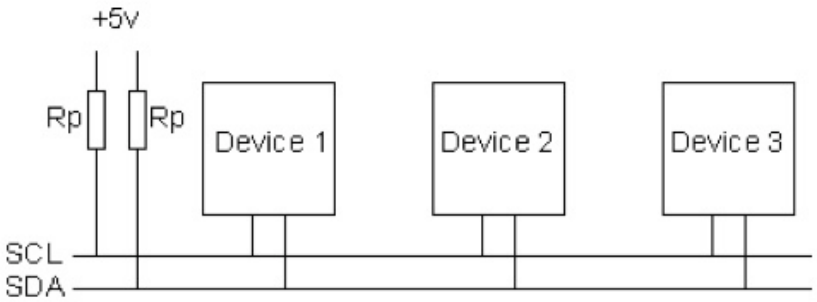
I²C Bus Characteristics

- 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
  - Shielded Twisted Pair Cabling
- 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 - Ramifications

I²C 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

Physical I²C Bus



- Pull-up Resistor Needed Since SCL/SDA are Open-Drain Drivers
  - can pull line low but not high w/o pull-up
  - 1.8k to 47k Ohm typical – depends on device specs
- Need Third Reference/GND Conductor
  - Shielded Twisted Pair for cabling works well

Why I2C has endured?

- Reliable performance using software-controlled collision detection and arbitration.
  - Collision detection/arbitration is ALWAYS a concern
  - Collisions can occur when two Masters attempt to broadcast within the same period of time
  - Some collision arbitration schemes can be quite complex such as the Ethernet CSMA (Carrier Sense Multiple Access) approach
  - I2C relies on hardware "wired-ANDing" of transmitted signals accomplished at the physical layer
- Ease of use. 2 lines connect all ICs in a system.
- Software controlled addressing scheme eliminating need for address-decoding hardware.
- Simple hardware for Collision Detection
- Does not support a huge number of devices on same bus

I^2C Bus Definitions

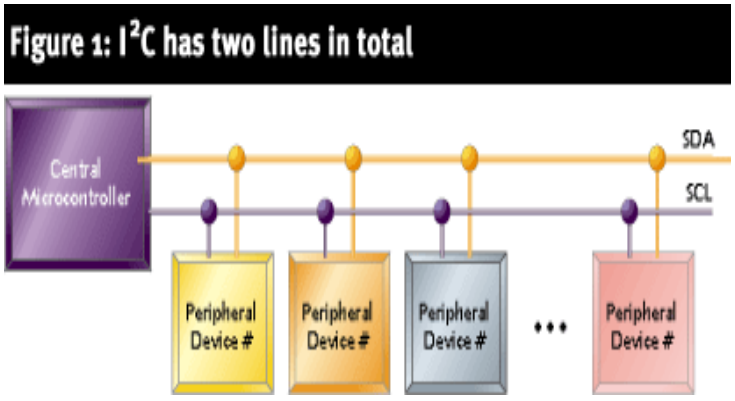
- **Master:**
  - Initiates a transfer by generating start and stop conditions
  - Generates the clock
  - Transmits the slave address
  - Determines data transfer direction
- **Slave:**
  - Responds only when addressed
  - Timing is controlled by the clock line

I²C Bus States

SCL and SDA are bi-directional !

- **F**(free) – Bus is idle or free. Both SDA and SCL are in a high state
- **S**(Start) or **R**(Restart) – SDA changes from high to low with the SCL line remaining high. All data transfers begin with **S**(Start) condition.
- **C**(Change) – SCL line is low. Data bit to be transferred is applied to the SDA line.
- **D**(Data) – high or low bit of information on SDA line is valid during the high level of the SCL line.
- **P**(Stop) – SDA line changes from low to high with SCL line remaining high. All data transfers end with **P**(Stop) condition.

I²C System Structure



I2C is a 2 wire serial bus as shown above. The 2 signals are

SDA→ Serial Data

SCL → Serial Clock

Together these signals make it possible to support serial transmission.

I²C Masters and Slaves

- The device that initiates the transaction on the I2C bus is termed the **master**. The master normally controls the clock signal.
- A device being addressed by the master is called the **slave**.
- There needs to be at least one master(usually a microcontroller or a DSP) on the bus, but there can be more than one master. All the masters on a bus have equal priority.
- Devices may be either masters, slaves, or both (master/slave)

I²C BUS ADDRESSING

- Every device on the I2C bus has a unique 7 bit (or 10 bit) I2C address
- Typically the 4 most significant bits are fixed and assigned to specific categories of devices. (eg. 1010 is assigned to serial EEPROM)
- The lower 3 bits are programmable allowing 8 devices of one kind to be present on a single I2C bus

How Many Devices will a Single I2C bus Support?

I²C Addressing

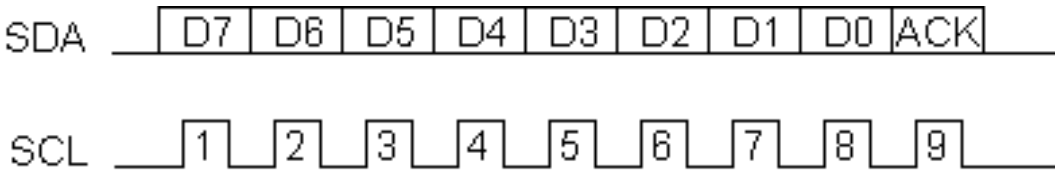
- **Each node has a unique 7 (or 10) bit address**
- **Peripherals often have fixed and programmable address portions**
- Typically the 4 most significant bits are fixed and assigned to specific categories of devices. (eg. 1010 is assigned to serial EEPROM)
- **Addresses starting with 0000 or 1111 have 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

I²C device addressing

- All I2C devices are either 7 (or 10) bits. Yet we send out 8 bits when trying to address a device.
- This extra bit is used to inform the slave whether the master is writing to it or reading from it.
- The 7 bit address are placed in the upper 7 bits and the R/W bit is placed in the LSb.
- So if LSb is 1, master wants to read from a slave. Else if LSb is 0, master wants to write to a slave.

DATA TRANSFER CHARACTERISTICS

- Data is transferred in sequences of 8 bits
- Bits are placed on SDA line with MSb first
- SCL line is then pulsed high and then low
- For every 8 bits transferred, the device receiving the data sends back an acknowledge bit



Data transfer(contd)

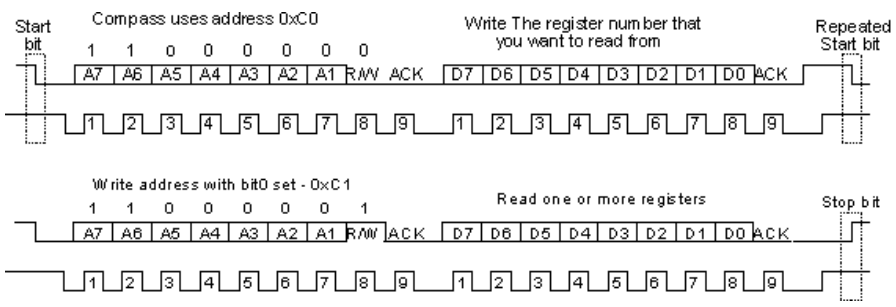
- If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte.
- If it sends back a high ACK, then the device is indicating that it cannot accept any further data and the master should terminate the transfer by sending a stop sequence.

Writing to a slave device

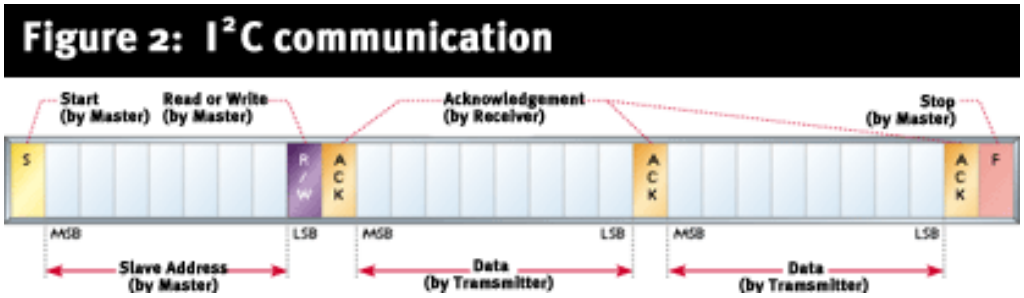
- Master sends a start sequence. This alerts all the slave devices to an impending transaction and they should listen, in case it is for them.
- Next the master sends out the device address with read/write bit low. The slave that matches this address will continue the transaction , while others ignore.
- Master can now send data byte(s).
- Master sends the stop sequence.

Reading from a slave

- Master sends a start sequence.
- Master sends the device address with read/write bit high.
- Master reads data from the device.
- Master sends the stop sequence



DATA TRANSFER EXAMPLE



Write Data

F	S	Slave Address	$\overline{W}$	A	Data	A	Data	$\overline{A}$	P	F
					< n data bytes >		last data bytes			

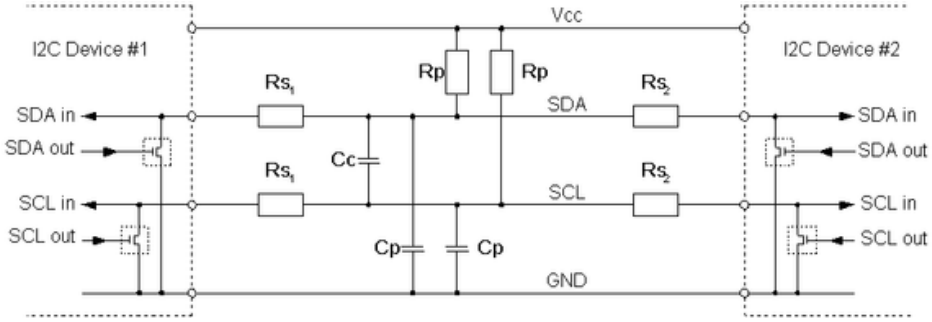
SLOW PERIPHERALS – CLOCK STRETCHING

- Master Device Determines Clock Speed
- I2C Provides an Explicit Clock Signal
  - relieves master and slave from synchronizing exactly to predefined baud rate
- Slow Peripherals Cannot Co-operate with Given Clock Speed from Master
- Slave Holds Down Clock if Needs to Reduce Bus Speed
- Master is Required to Read Back Clock Signal After Releasing to High State
  - Must Wait Until Line Has Actually Gone High
- Allows Slave to Actually Reduce Bus Bandwidth
- Slowest Slave on Bus can Impact Bus Bandwidth

I²C 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.

EQUIVALENT CIRCUIT

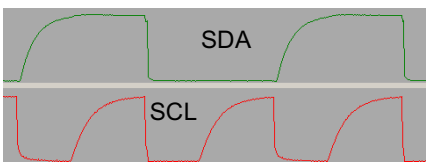


- VCC – I2C Voltage, Typical=1.2V to 5.5V
- GND – Common Ground
- SDA – Serial Data (I2C Data Line)
- SCL – Serial Clock (I2C Clock Line)
- Rp – Pull-up Resistance (I2C Termination)
- Rs – Serial Resistance
- Cp – Wire Capacitance
- Cc – Cross Channel Capacitance

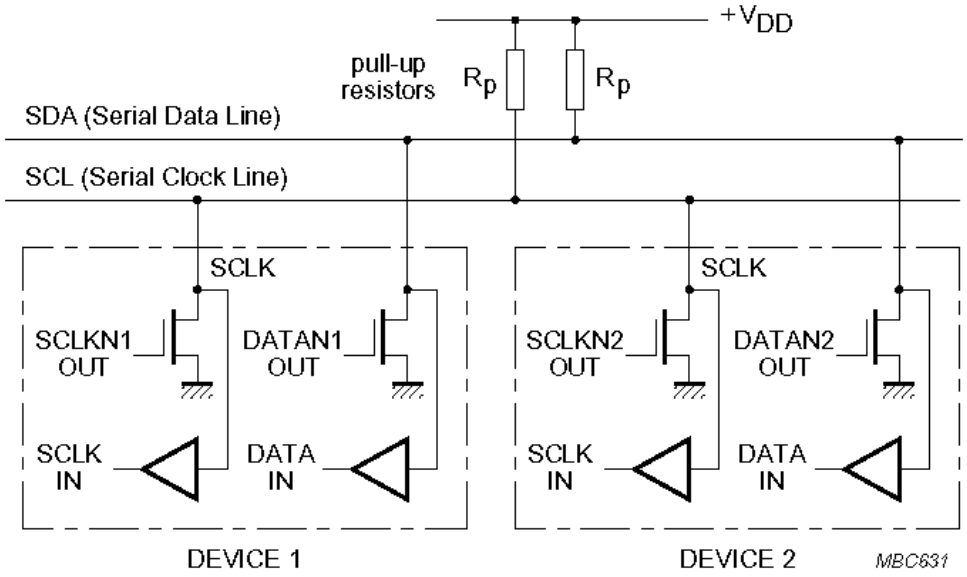
Rp=10kΩ

Cp=300pF

SCL Clock=100kHz

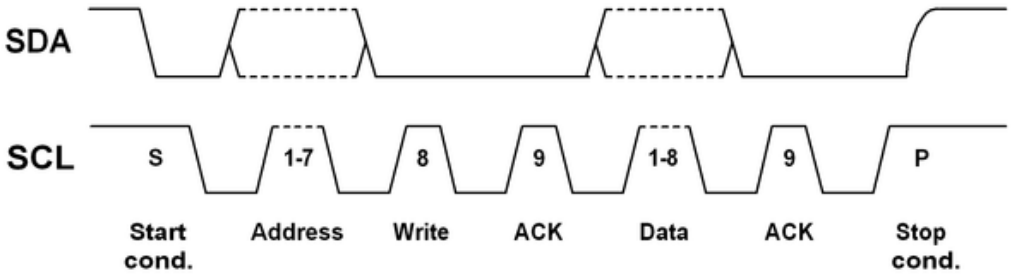


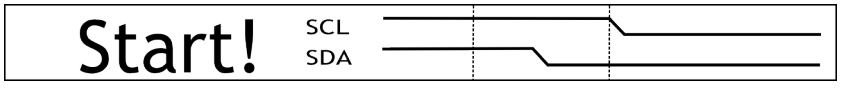
I²C Electrical Aspects

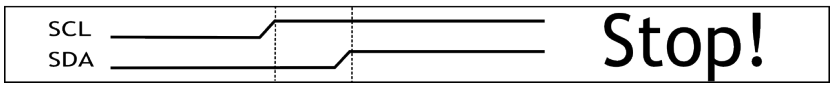


- I²C devices are wire ANDed together.
- If any single node writes a zero, the entire line is zero

TIMING DIAGRAMS

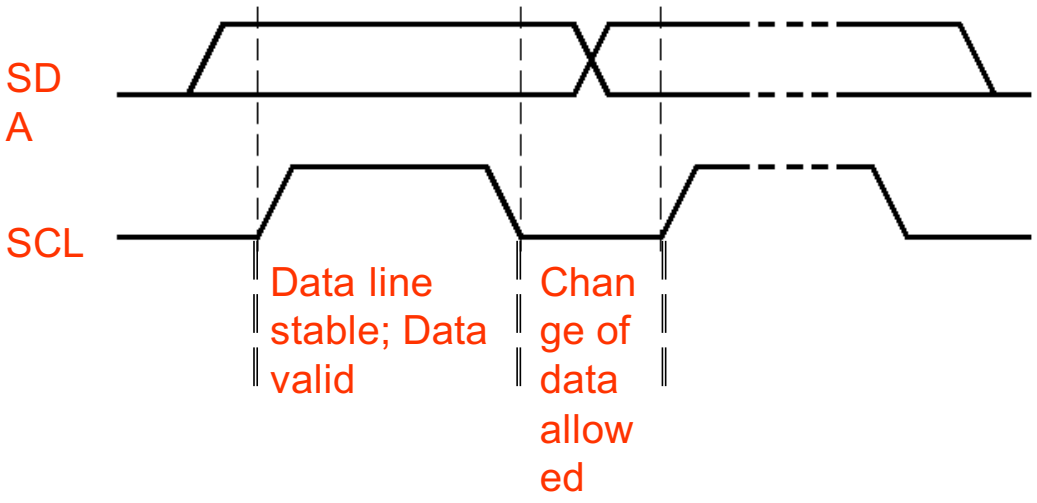






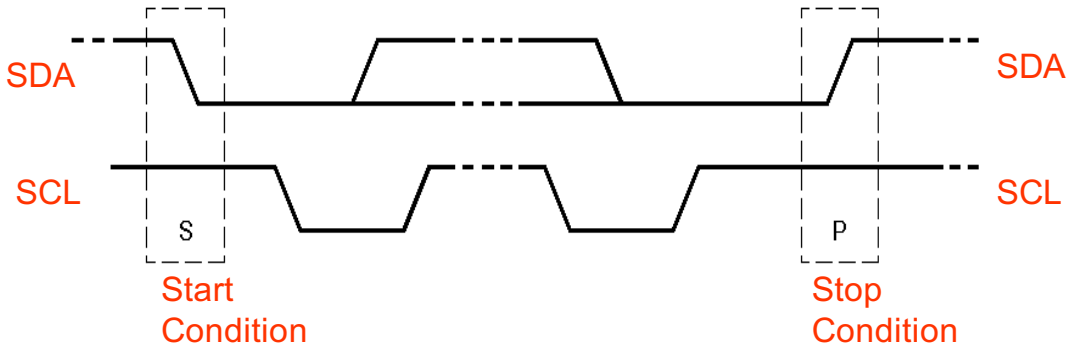
Bit Transfer on the I²C Bus

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

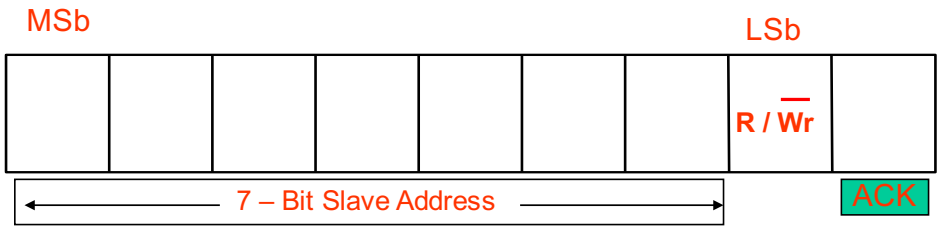


Start and Stop Conditions

- 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



First Byte in Data Transfer on the I²C Bus



R/$\overline{Wr}$

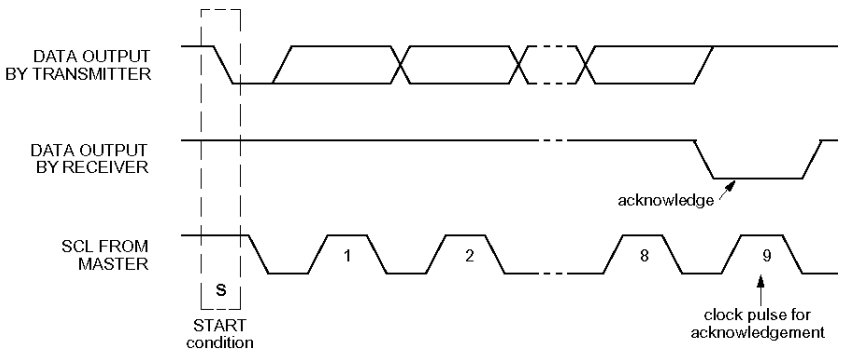
0 – Slave written to by Master

1 – Slave read by Master

ACK – Generated by the slave whose address has been output

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.

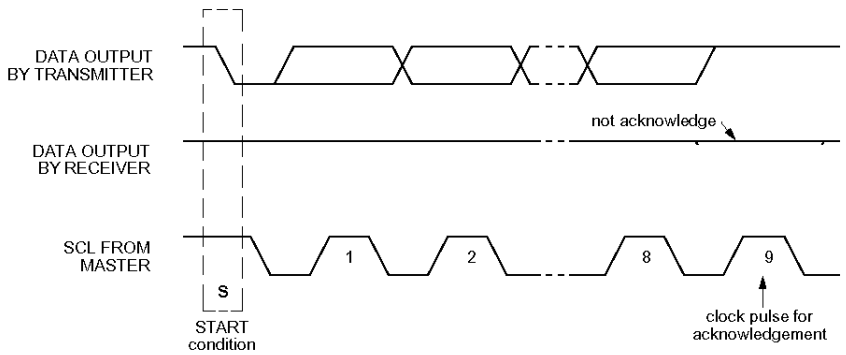
Acknowledgement from receiver

Acknowledgements

- **From Slave to Master Transmitter:**
  - After address received correctly
  - After data byte received correctly
- **From Slave to Master Receiver:**
  - Never (Master Receiver generates ACK)
- **From Master Transmitter to Slave:**
  - Never (Slave generates ACK)
- **From Master Receiver to Slave:**
  - After data byte received correctly

Negative Acknowledge

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



Transmitter releases SDA line during 9th clock pulse.

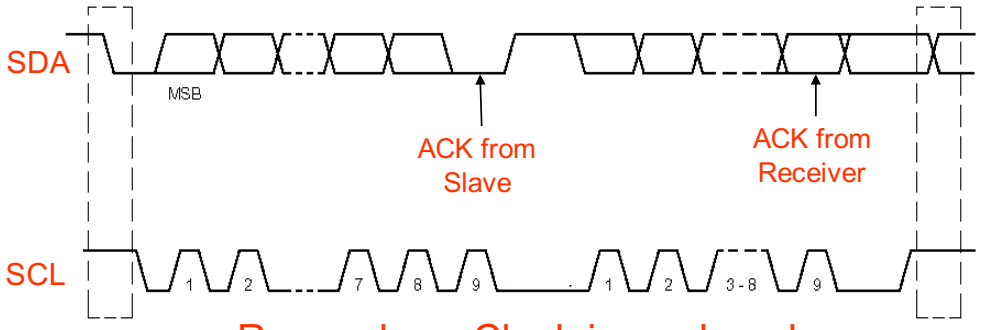
Not acknowledgement (NACK) from receiver

Negative Acknowledge (Cont'd.)

- **From Slave to Master Transmitter:**
  - After address not received correctly
  - After data byte not received correctly
  - Slave Is not connected to the bus
- **From Slave to Master Receiver:**
  - Never (Master Receiver generates ACK)
- **From Master Transmitter to Slave:**
  - Never (Slave generates ACK)
- **From Master Receiver to Slave:**
  - After last data byte received correctly

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**

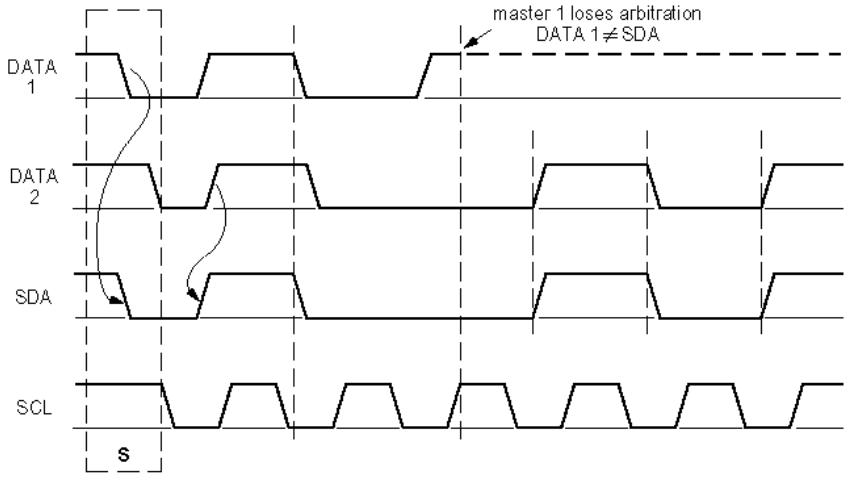


Remember : Clock is produced by Master

Multi-master I²C Systems

- **Multimaster situations require two additional features of the I²C 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

- **I²C 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**

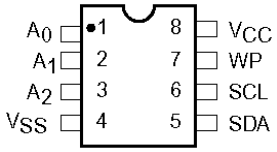
Bus Recovery

- **An I²C 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.**

I2C SERIAL MEMORY EXAMPLE

Serial EEPROM (Part 24WC32)

- **400 KHz I2C Bus Compatible***
- **1.8 to 6 Volt Read and Write Operation**
- **Cascadable for up to Eight Devices**
- **32-Byte Page Write Buffer**
- **Self-Timed Write Cycle with Auto-Clear**
- **Zero Standby Current**
- **Commercial, Industrial and Automotive Temperature Ranges**



- **Write Protection– Entire Array Protected When WP at VIH**
- **1,000,000 Program/Erase Cycles**
- **100 Year Data Retention**

Pin Name	Function
A0, A1, A2	Device Address Inputs
SDA	Serial Data/Address
SCL	Serial Clock
WP	Write Protect
V_{CC}	+1.8V to +6V Power Supply
V_{SS}	Ground

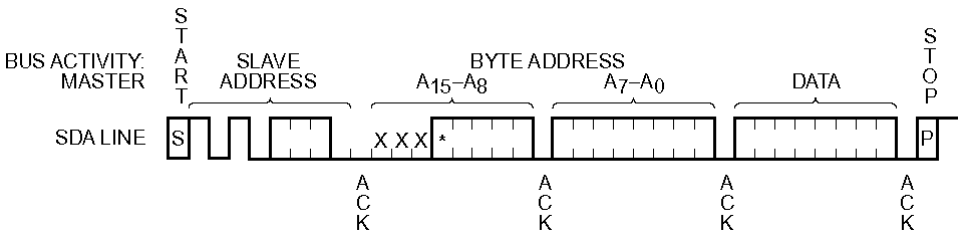
24WC32 Characteristics

- 32Kb memory organised as 4K x 8bit
- 12 address bits (2^{12} = 4K)
- Device Address :

1	0	1	0	A2	A1	A0	R/$\overline{W}$

- Writing
  - Byte Write
  - Page Write
  - Write time 10ms maximum
  - Write acknowledge Polling
- Reading
  - Immediate/Current address reading
  - Selective/Random Read
  - Sequential Read

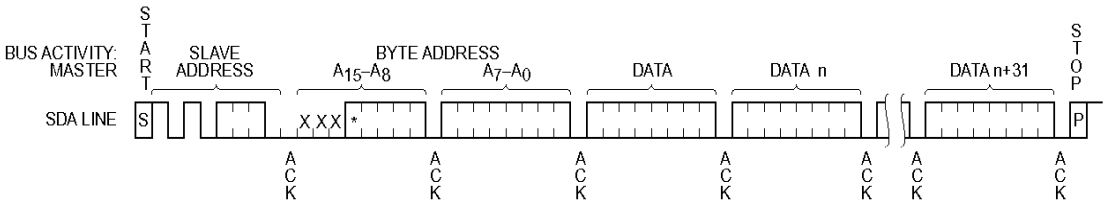
Writing a Single Data Byte



After the STOP bit is receive the device internally programs the EEPROM with the received data byte.

The programming can take up to 10ms (max.). The device will be busy during this period and will not respond to its slave address.

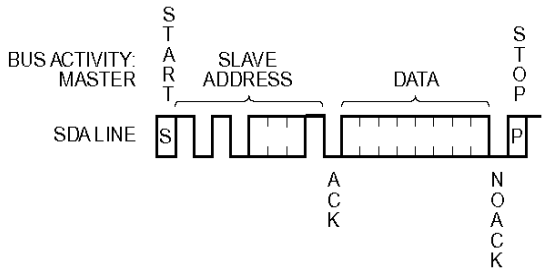
Writing Multiple Bytes (Page Write)



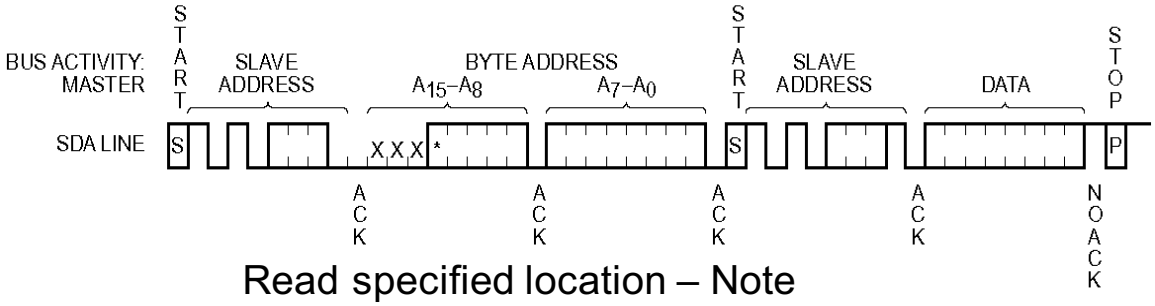
The bytes are received by the device and stored internally in a buffer before being programmed into the EEPROM.

A maximum of 32 bytes (one page = 32 bytes) may be written at one time for the 24WC32 device.

Reading EEPROM



Read current location



Read specified location – Note repeated start to prevent loss of bus during read process.

Type of I²C Implementations

- **Byte Oriented Interface**
  - Data is handled one byte at a a time
  - Processor interprets a status byte when an event occurs
  - For instance Philips 8xC554, 8xC591
- **Bit Oriented Interface**
  - Processor is involved in every bus event when the interface is not Idle
- **"Bit Banged"**
  - Implemented completely in software on 2 regular I/O pins of the microcontroller
  - Works for single master systems
  - Not recommended for Slave devices or Multimaster systems

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^2C 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

End use

- **Telecom:** Mobile phones, Base stations, Switching, Routers
- **Data processing:** Laptop, Desktop, Workstation, Server
- **Instrumentation:** Portable instrumentation, Metering systems
- **Automotive:** Dashboard, Infotainment
- **Consumer:** Audio/video systems, Consumer electronics (DVD, TV etc.)

Applications

- There are some specific applications for certain types of I^2C devices such as TV or radio tuners, but in most cases a general purpose I^2C device can be used in many different applications because of its simple construction.

I2C designer benefits

- Functional blocks on the block diagram correspond with the actual ICs; designs proceed rapidly from block diagram to final schematic
- No need to design bus interfaces because the I2C-bus interface is already integrated on-chip
- Integrated addressing and data-transfer protocol allow systems to be completely software-defined
- The same IC types can often be used in many different applications

I2C designer benefits

- Design-time improves as designers quickly become familiar with the frequently used functional blocks represented by I^2C-bus compatible ICs
- ICs can be added to or removed from a system without affecting any other circuits on the bus
- Fault diagnosis and debugging are simple; malfunctions can be immediately traced
- Software development time can be reduced by assembling a library of reusable software modules
- The simple 2-wire serial I^2C-bus minimizes interconnections so ICs have fewer pins and there are fewer PCB tracks; resulting in smaller and less expensive PCBs

I^2C Manufacturers benefits

- The completely integrated I^2C-bus protocol eliminates the need for address decoders and other 'glue logic'
- The multi-master capability of the I^2C-bus allows rapid testing/alignment of end-user equipment via external connections to an assembly-line
- Increases system design flexibility by allowing simple construction of equipment variants and easy upgrading to keep design up-to-date
- The I^2C-bus is a de facto world standard that is implemented in over 1000 different ICs (Philips has > 400) and licensed to more than 70 companies