What is I^2C (inter-IC)

- Originally designed by Philips to enable inter-IC communication using a minimum number of pins
- Build a simple universal bus for IC compatibility from different vendors
  - Simple hardware spec
  - Simple software protocol
  - Speed at 100kbps -> 400kbps -> 3.4Mbps
- Features
  - No special wiring or connector needed
  - A serial data line (SDA) and a serial clock line (SCL)
  - Each device has unique address, is software addressable
  - Master/slave relationship, can have multiple masters
  - Serial, 8-bit, bi-directional data transfer
  - Max bus capacity of 400pF

Serial Bus Comparison
[reference: Philips AN10216]

UART	CAN	USB	SPI	I2C
• Well known • Cost effective • Simple	• Secure • Fast	• Fast • Plug&Play HW • Simple • Low cost	• Fast • Universally accepted • Low cost • Large portfolio	• Simple • Well known • Universally accepted • Plug&play • Large portfolio • Cost effective
• Limited functionality • Point to point	• Complex • Automotive oriented • Limited portfolio • Expensive firmware	• Require powerful master • No plug&play software – drivers needed	• No plug&play HW • No fixed standard	• Limited speed

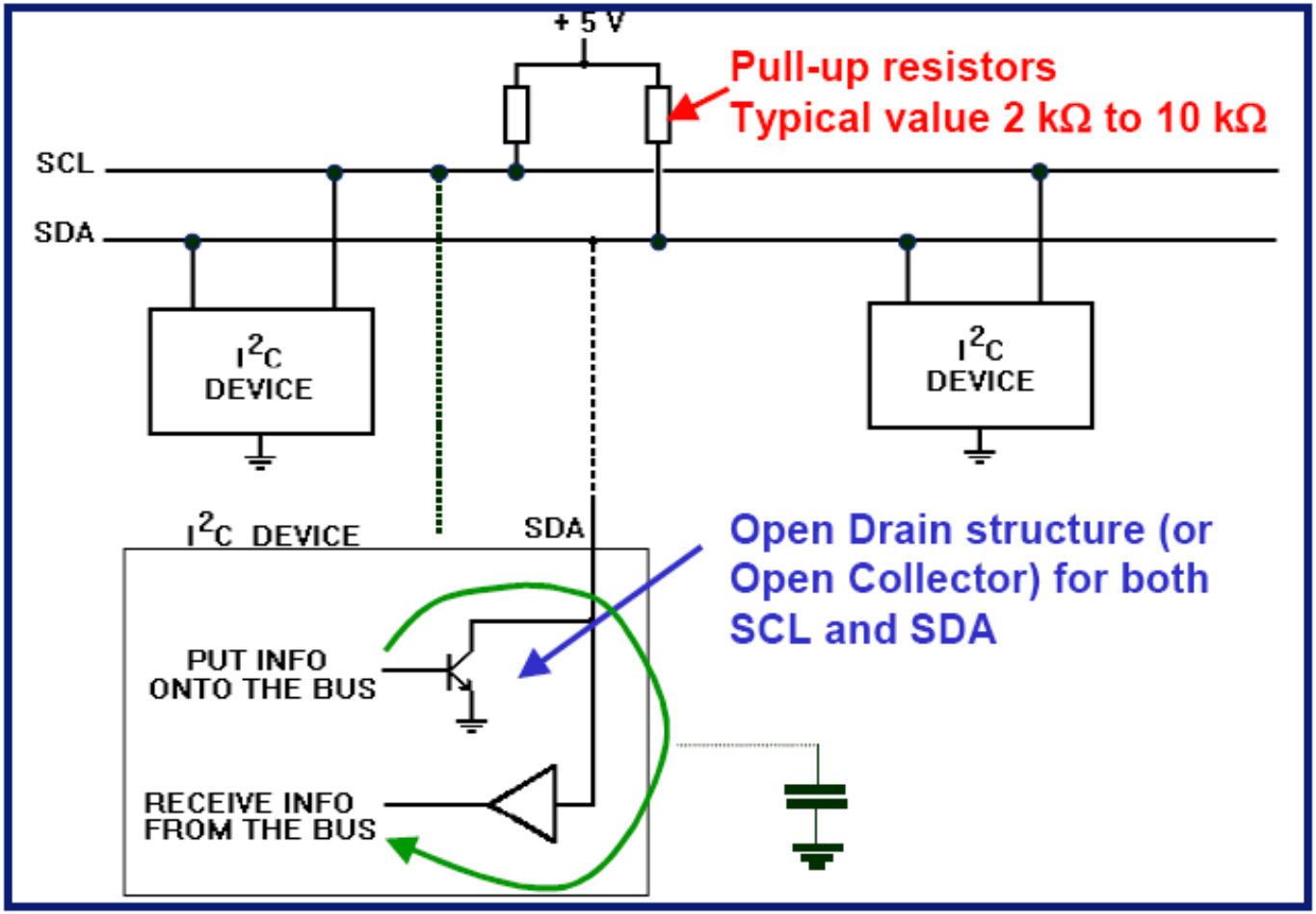
Software and Communication Procedure

- Software
  - Simple message format generated by microcontrollers
  - Devices have complete interfaces (built-in I^2C)
- Procedure for communication
  - Wait until I^2C bus is free: both SDA and SCL are high.
  - Put a "START" message on bus to claim bus (all other ICs will then listen)
  - Put a clock signal on SCL line for other ICs as reference time (the data on SDA wire must be valid when SCL switching from low to high)
  - Put binary address in series to identify target IC
  - Put one bit to identify direction (SEND or RECEIVE)
  - Ask other IC to acknowledge the address and readiness to transfer
  - Transfer (many) 8-bit data after receiving ack
  - The master send "STOP" message to free up the bus

Identify a Device

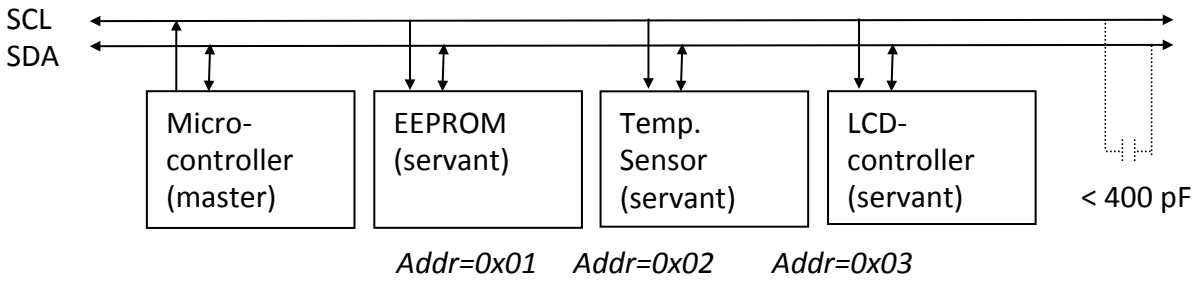
- Master device polls using a specific unique ID or "address"
  - If it knows the other ICs by design, it can talk directly
  - A master can also check the presence of other ICs at runtime (query/response)
- Devices with Master capability can identify themselves to other masters
  - Enable plug and play
  - Bus speed can be different
- Only one pair of devices can have a data transfer session at a time

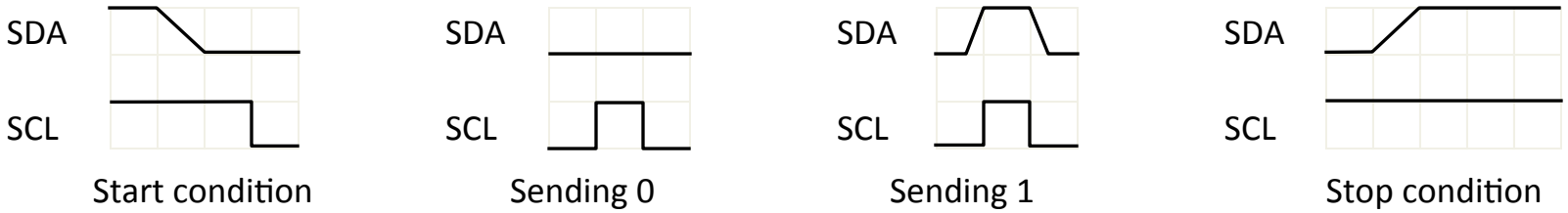
I²C Hardware Architecture

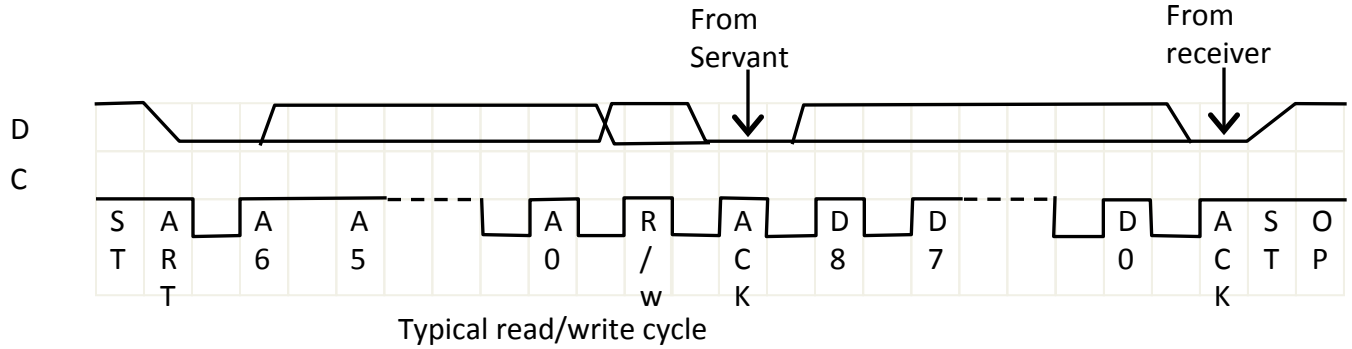


- Philips AN10216-01 I²C manual

START and STOP

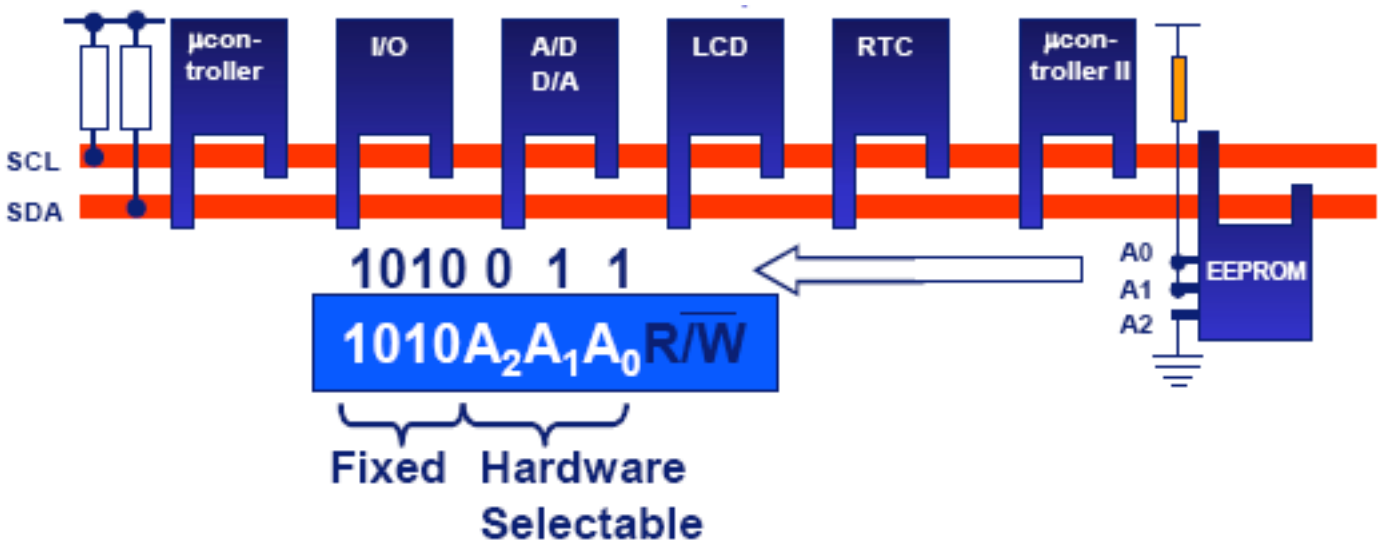






- Data on SDA must be stable when SCL is high
- Exceptions are the START and STOP conditions

I²C Addresses

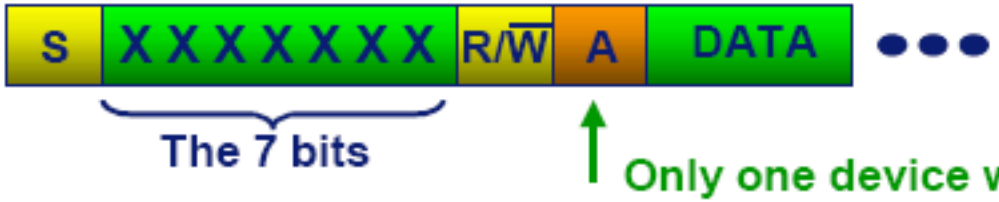


- Each device is addressed individually by software
- Unique address per device: fully fixed or with a programmable part through hardware pin(s).
- Programmable pins mean that several same devices can share the same bus
- Address allocation coordinated by the I²C-bus committee
- 112 different types of devices max with the 7-bit format (others reserved)

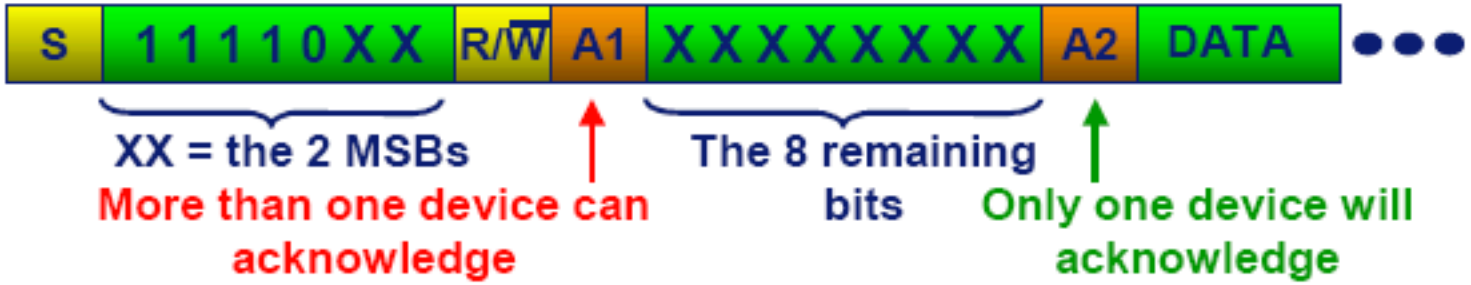
I²C Address, 7-bit and 10-bit formats

- The 1st byte after START determines the Slave to be addressed
- Some exceptions to the rule:
  - "General Call" address: all devices are addressed : 0000 000 + R/W =
  - 10-bit slave addressing : 1111 0XX + R/W = X

•7-bit addressing



• 10-bit addressing

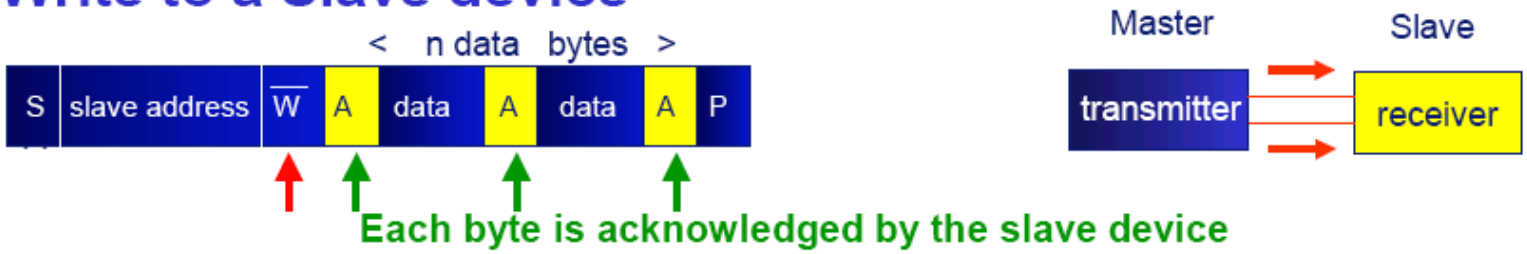


Bus Communication

- Data transfer is 8-bit bytes,
- Each byte of transfer is acknowledged with a 9th data bit generated by the receiver
- Apart from START and STOP, no device is allowed to change the state of the SDA bus line
- Multiple masters “compete” but only one wins the bus
- No minimum clock speed: masters allow the slower ICs to hold the SCL low until finish the transaction (i.e. clock stretching)

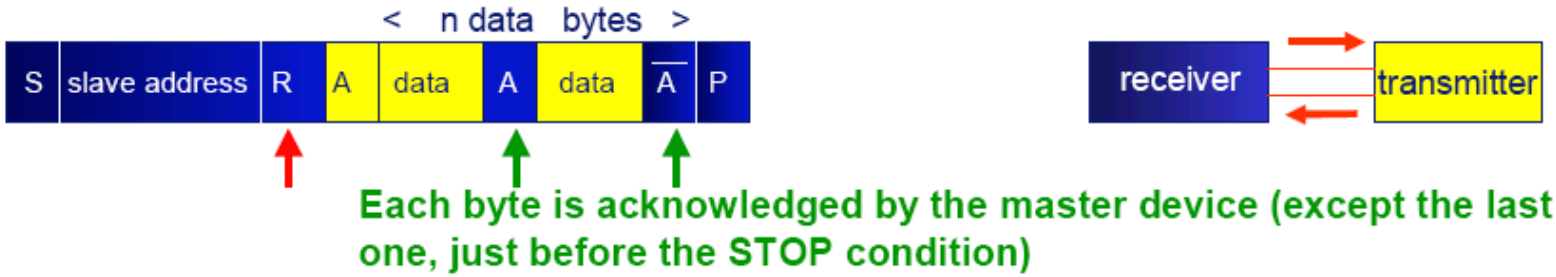
Single Read/Write

- **Write to a Slave device**



The master is a "MASTER - TRANSMITTER":
–it transmits both Clock and Data during the all communication

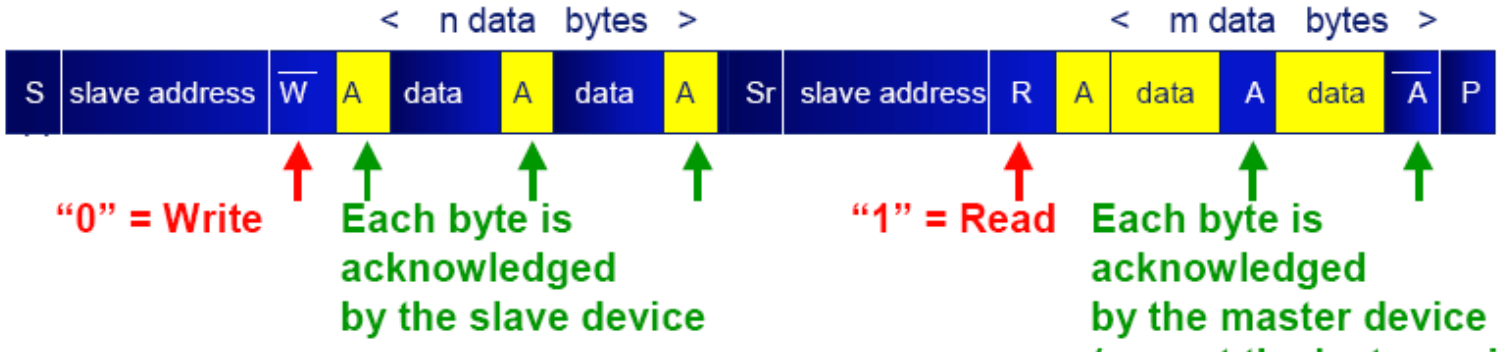
- **Read from a Slave device**



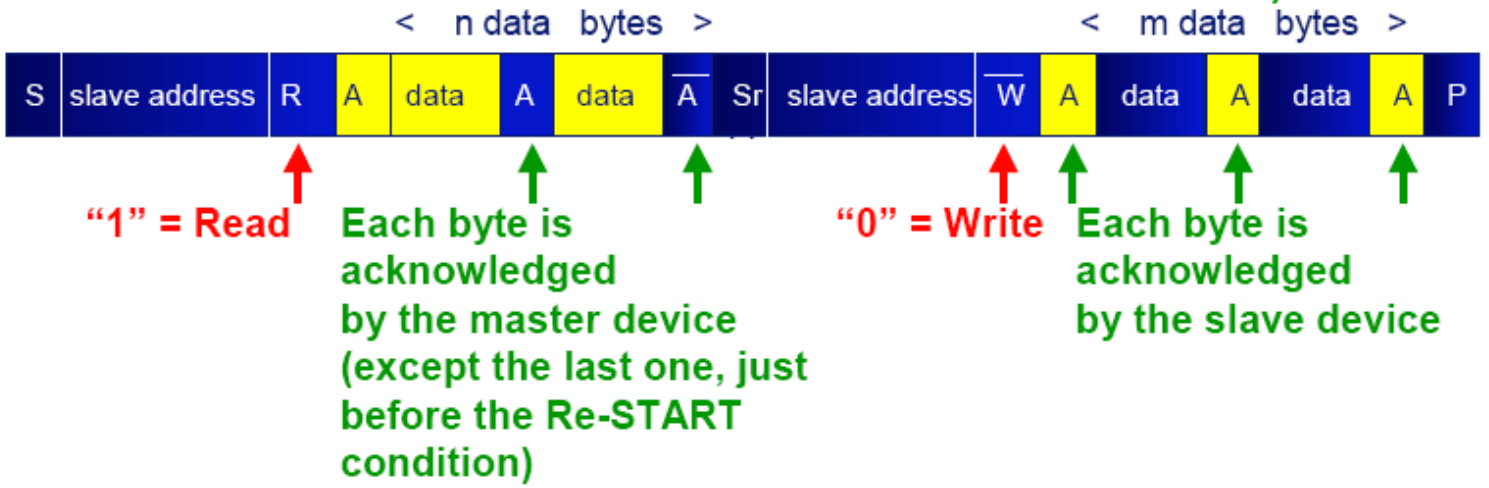
The master is a "MASTER TRANSMITTER then MASTER - RECEIVER":
– it transmits Clock all the time
– it sends slave address data and then becomes a receiver

Combined Read and Write

- **Combined Write and Read**



- **Combined Read and Write**



Acknowledge and Clock Stretching

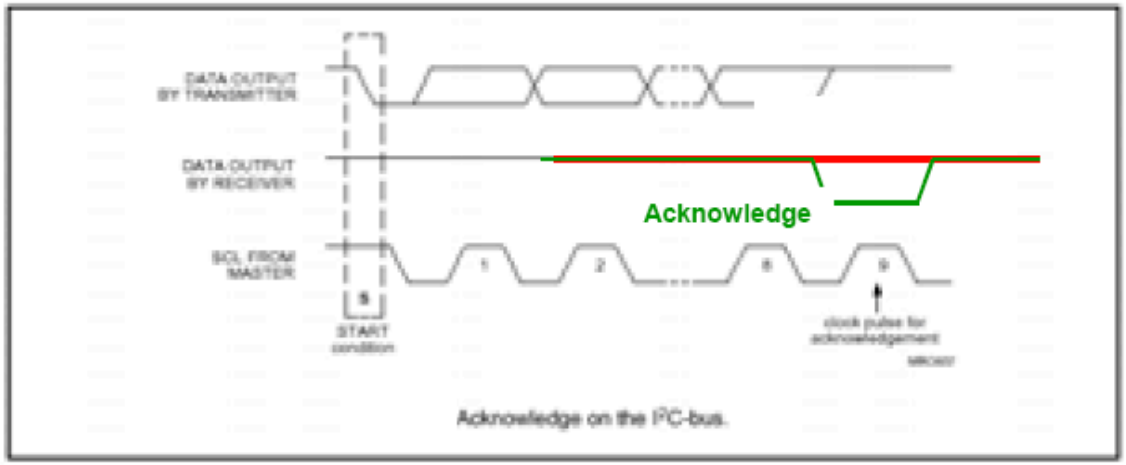
- **Acknowledge**

  Done on the 9th clock pulse and is mandatory

  → Transmitter releases the SDA line

  → Receiver pulls down the SDA line (SCL must be HIGH)

  → Transfer is aborted if no acknowledge



Acknowledge on the I²C-bus.

- **Clock Stretching**

  - Slave device can hold the CLOCK line LOW when performing other functions
  - Master can slow down the clock to accommodate slow slaves

Voltage Level Translation

- Traditionally I^2C devices are 5V, but new devices are 3.3V and becoming lower
  - Requires expensive 5V-tolerate devices
  - Devices with <2V supply voltage do not meet the 0.1Vdd noise margin, causing mismatch logic levels
- Voltage (Logic) level shifter
  - MOSFET based
  - Bi-directional level shifting without a direction control signal
  - Isolating a powered-down bus section from the rest
  - Protect low voltage side against voltage spikes at high voltage side

[ref: Philips AN97055]

I²C devices with different logic levels

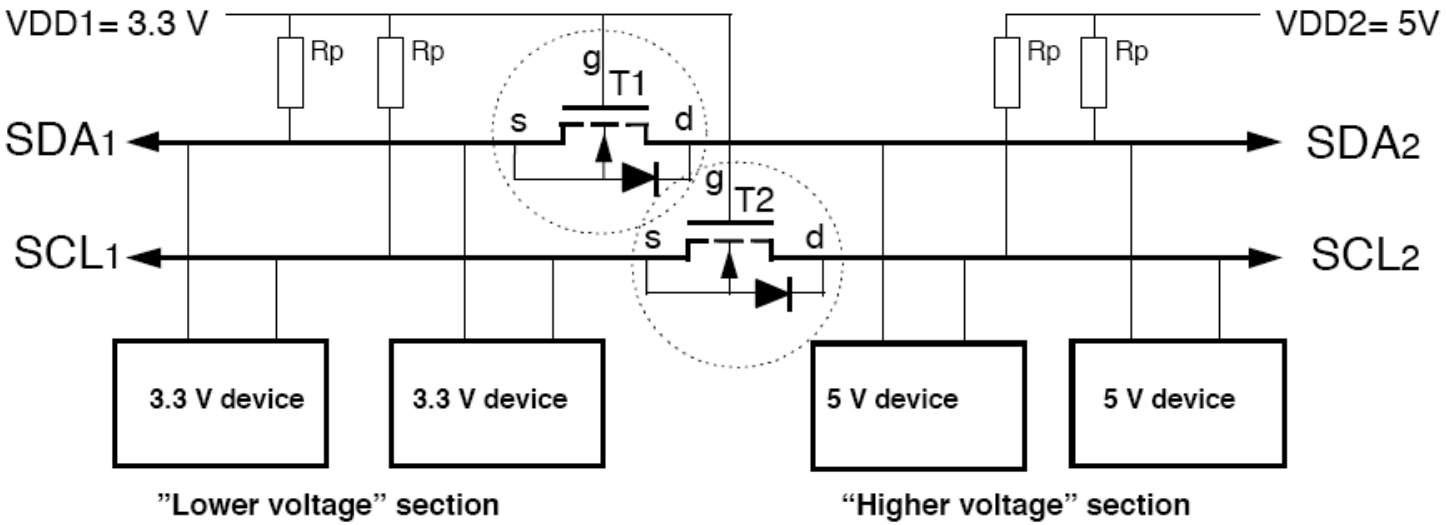


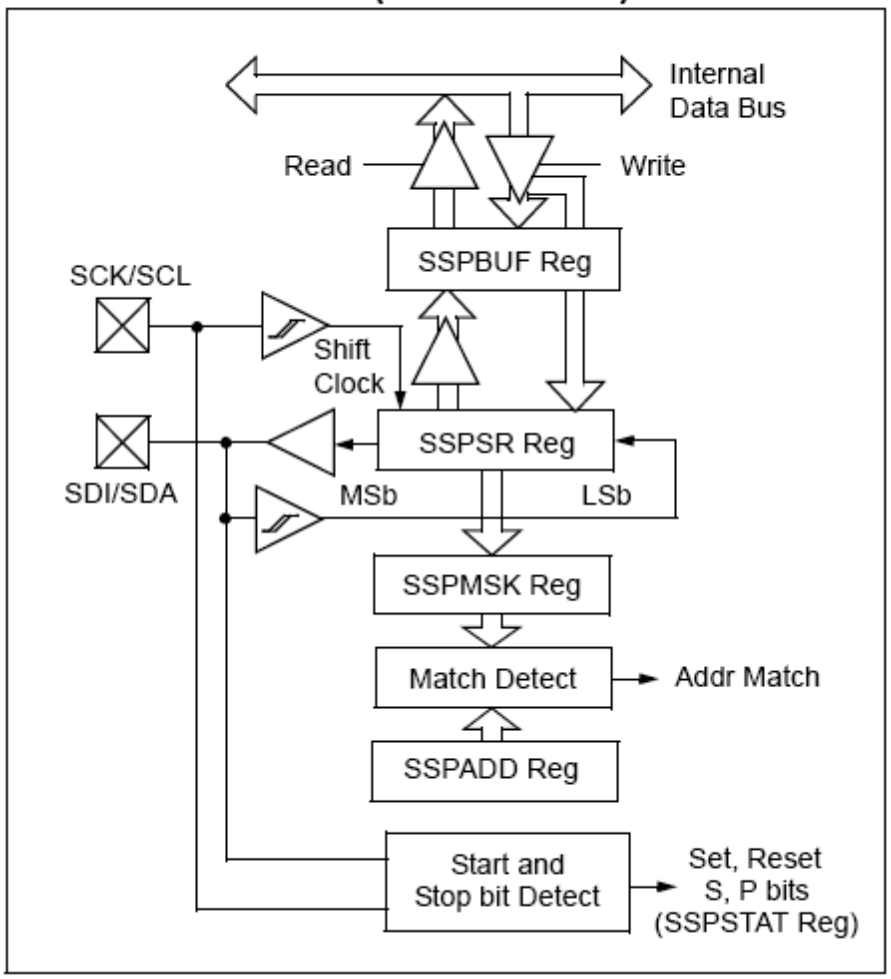
Figure 2. Bi-directional level shifter circuit connects two different voltage sections of an I²C-bus system.

- Each voltage section has pull-up resistors
- Each voltage section has I/Os with supply voltage related logic input levels
- Each bus line (SDA and SCL) has identical level shifter (N-channel enhancement MOS-FET)
- Gates(g) connected to low supply voltage, source (s) to bus lines of low voltage section, drains (d) to the bus line of high voltage section
  - Case 1: no device is pulling down, g=s=3.3V, MOSFET not conducting, high voltage section pulled up to 5V
  - Case 2: a 3.3V device pulls down the bus to LOW, s=LOW, g=3.3V, MOSFET conducting, high voltage section pulled down as well
  - Case 3: a 5V device pulls down the bus to LOW, drain-substrate diode cause Vgs pass threshold, MOSFET conducting, low voltage section pulled down

I²C with PIC18F45K20

- MSSP module in I2C mode fully implements master and slave functions
- Supports both 7-bit and 10-bit addressing
- Must configure SCL and SDA pins as inputs with the corresponding TRIS bits

FIGURE 17-7: MSSP BLOCK DIAGRAM (I²C™ MODE)



REGISTER 17-3: SSPADD: MSSP ADDRESS AND BAUD RATE REGISTER (I^2C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Master mode

bit 7-0 **ADD<7:0>:** Baud Rate Clock Divider bits
SCL pin clock period = ((ADD<7:0> + 1) *4)/FOSC

10-Bit Slave mode: Most significant address byte

bit 7-3 Not used: Unused for most significant address byte. Bit state of this register is a don't care. Bit pattern sent by master is fixed by I^2C specification and must be equal to '11110'. However, those bits are compared by hardware and are not affected by the value in this register.

bit 2-1 **ADD<9:8>**: Two most significant bits of 10-bit address

bit 0 **Not used**: Unused in this mode. Bit state is a "don't care".

10-Bit Slave mode: Least significant address byte

bit 7-0 **ADD<7:0>**: Eight least significant bits of 10-bit address

7-Bit Slave mode

bit 7-1 **ADD<7:1>:** 7-bit address

bit 0 **Not used**: Unused in this mode. Bit state is a "don't care".

REGISTER 17-4: SSPSTAT: MSSP STATUS REGISTER (I^2C MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P(1)	S(1)	R/$\overline{W}$(2, 3)	UA	BF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SMP:** Slew Rate Control bit

In Master or Slave mode:

1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high-speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit

In Master or Slave mode:

1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit

In Master mode:

Reserved.

In Slave mode:

1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit(1)

1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

bit 3 **S:** Start bit(1)

1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit (I^2C mode only)(2, 3)

In Slave mode:

1 = Read
0 = Write

In Master mode:

1 = Transmit is in progress
0 = Transmit is not in progress

- d

REGISTER 17-5: SSPCON1: MSSP CONTROL 1 REGISTER (I^2C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started (must be cleared by software)

0 = No collision

In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared by software)

0 = No collision

In Receive mode (Master or Slave modes):

This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared by software)

0 = No overflow

In Transmit mode:

This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins. When enabled, the SDA and SCL pins must be configured as inputs.

0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCK Release Control bit

In Slave mode:

1 = Release clock

0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:

Unused in this mode.

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits

1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled

1011 = I^2C Firmware Controlled Master mode (Slave Idle)

1000 = I^2C Master mode, clock = Fosc/(4 * (SSPADD + 1))

0111 = I^2C Slave mode, 10-bit address

0110 = I^2C Slave mode, 7-bit address

Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

- d

17.4.2 OPERATION

The MSSP module functions are enabled by setting SSPEN bit of the SSPCON1 register.

The SSPCON1 register allows control of the I^2C operation. Four mode selection bits of the SSPCON1 register allow one of the following I^2C modes to be selected:

- I^2C Master mode, clock = (FOSC/(4 x (SSPADD + 1))
- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I^2C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I^2C Firmware Controlled Master mode, slave is Idle

Selection of any I^2C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRIS bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

- d

17.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I^2C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

- d

I^2C Master Mode Operation

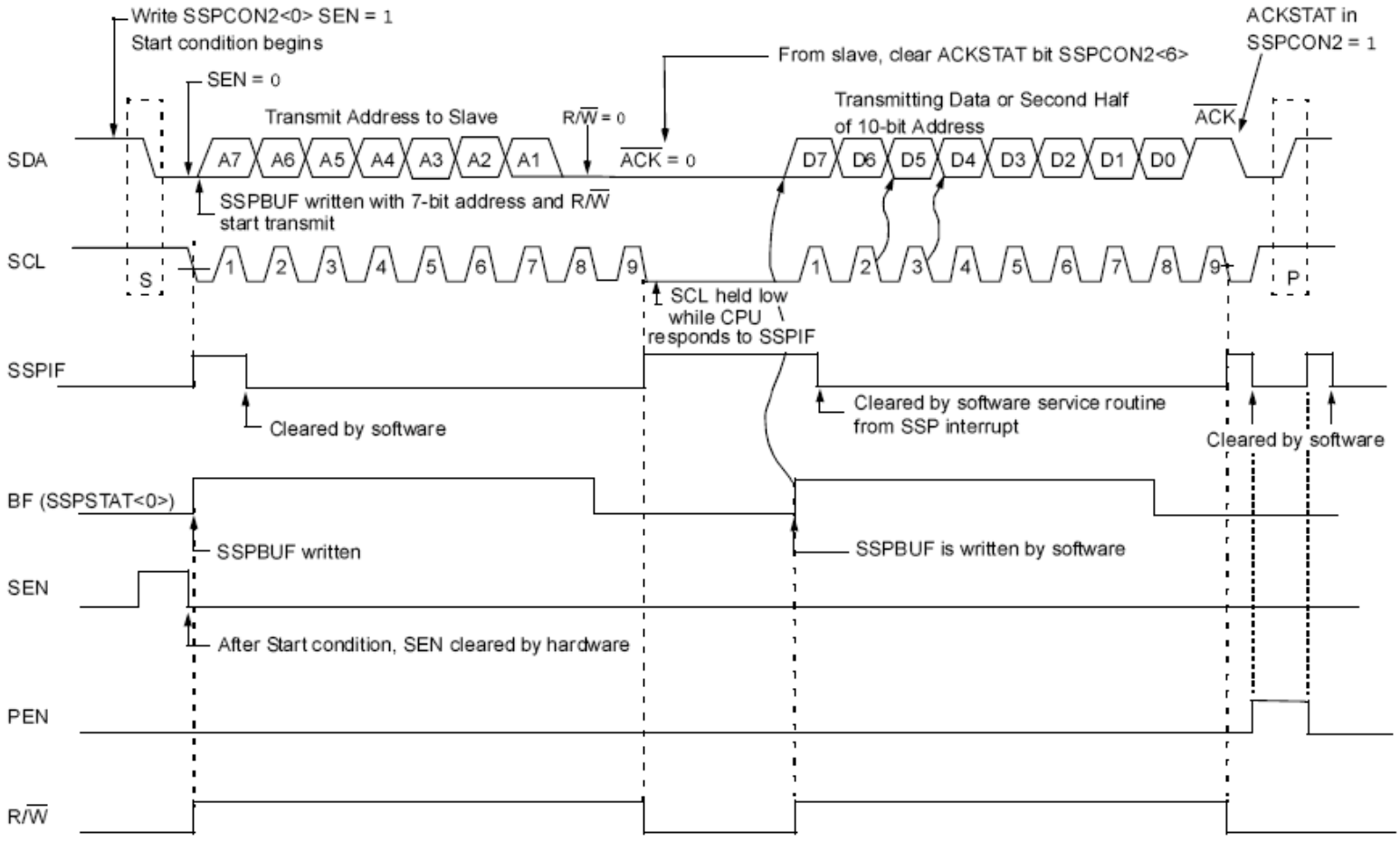
- Master device generates all the serial clock pulses and START and STOP conditions
- A transfer is ended with a STOP (releasing the bus), or a Repeated START (not releasing the bus)
- First byte transmitted = slave address (7bit) and R/W* bit (‘0’ for transmit, ‘1’ for receive)
- Data are 8-bits, followed by a ACK bit
- Baud Rate Generator use to set SCL clock frequency for 100KHz, 400KHz, or 1MHz. (check datasheet section 17.4.7)

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
2. SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the PEN bit of the SSPCON2 register.
12. Interrupt is generated once the Stop condition is complete.

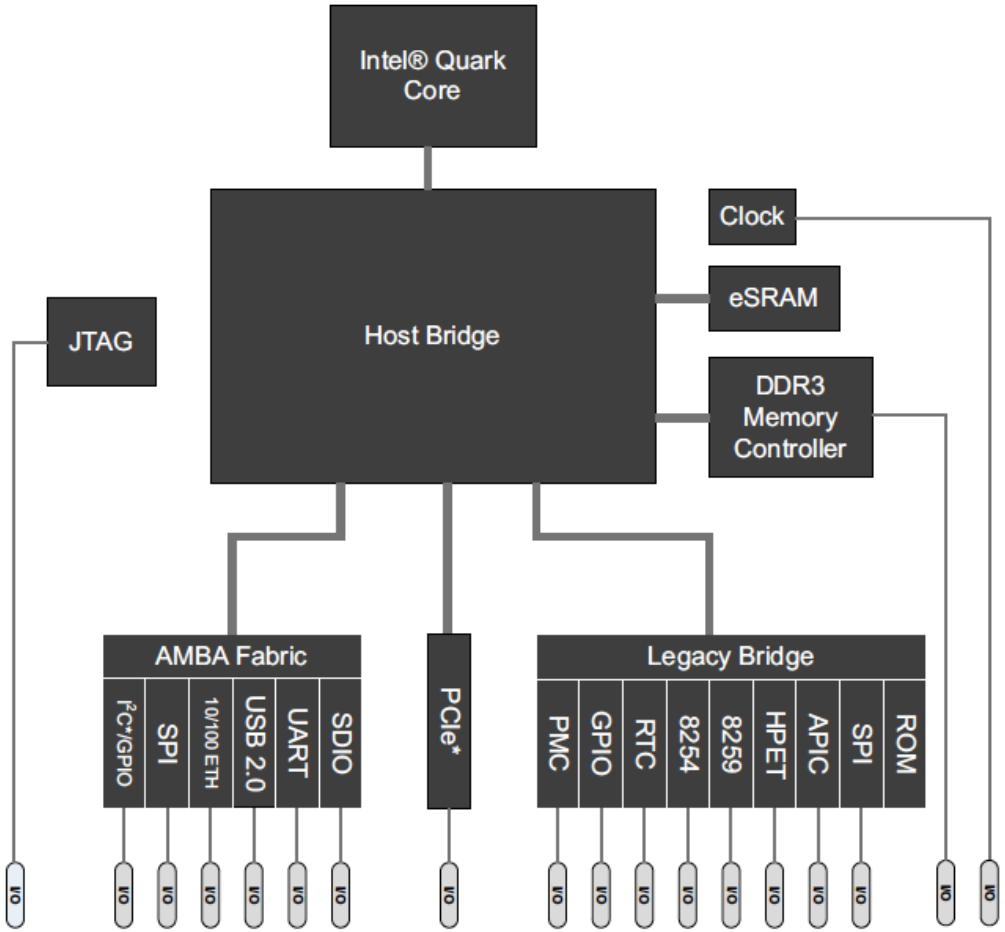
- d

I²C Master Mode Transmission Waveform



I²C on Intel Galileo

- Intel Galileo Development board supports I²C as Quark has built-in I²C controller (master)



I^2C Jumper on Galileo

- I^2C* Address Jumper
  - To prevent a clash between the I2C* Slave address of the on board I/O expander and EEPROM with any external I2C Slave devices, jumper J2 can be used to vary the I2C* address of the on-board devices.
  - With J2 connected to pin 1 (marked with white triangle), the 7-bit I/O Expander address is 0100001 and the 7- bit EEPROM address is 1010001. Changing the jumper position changes the I/O Expander address to 0100000 and the EEPROM address to 1010000.

I²C on Linux (Intel Galileo)

- Quark's I²C is supported with standard Linux i2c driver
- Its I²C register interface is 100% compatible with the upstream `i2c-designware-core` driver
- Load the driver (`modprobe i2c-dev`), and the interface is /dev/i2c-0
- communicate with downstream i2c devices using standard Linux API (e.g. i2cdetect)
- To load I²C driver in isolation from GPIO, use

```
modprobe intel_qrk_gip gpio=0
modprobe intel_qrk_gip gpio=0 enable_msi=0
```

User space Interface for I²C/SMBus

- Documented in Documentation/i2c/dev-interface
- Source code of I²C master drivers are under drives/i2c/busses/ (i2c-designware.c)
- Load the i2c-dev kernel module to create device nodes
- Each I²C master gets a character device node at /dev/i2c-N, where N is the master ID number
- Programming interfaces
  - read() and write() can do single direction transfer,
  - ioctl() does combined transfers (read and write in one transfer)
  - include the i2c-dev.h

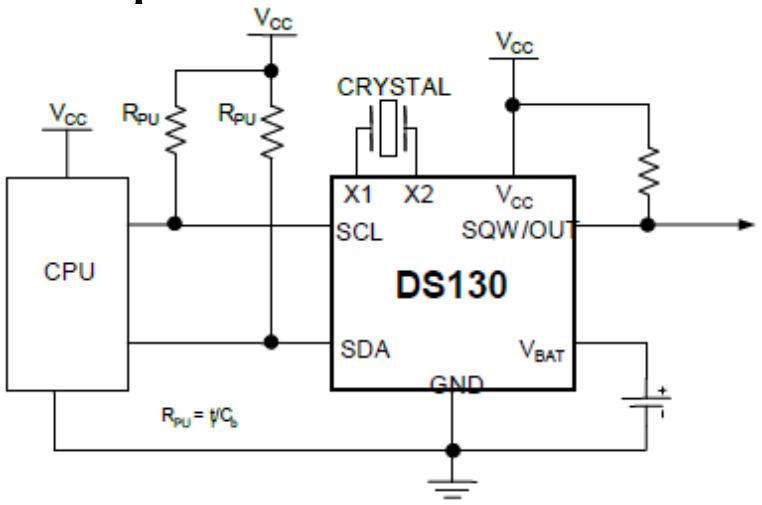
An Example I^2C device:
Real-Time Clock 1307

Introduction

- What is DS1307?
  - A real-time clock
  - Full binary-coded decimal clock/calendar
  - Transferred through an I2C serial interface
  - Automatically switch between external power and backup supply.
  - The end of month data is automatically adjusted.

Operating Circuit

- The RTC module only has 5 pins, the other 3 pins are interconnected.
  - VCC, GND are power and ground pin;
  - SCL, SDA are clock and data pin;
  - SQW/OUT is square wave output.



Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Control Register

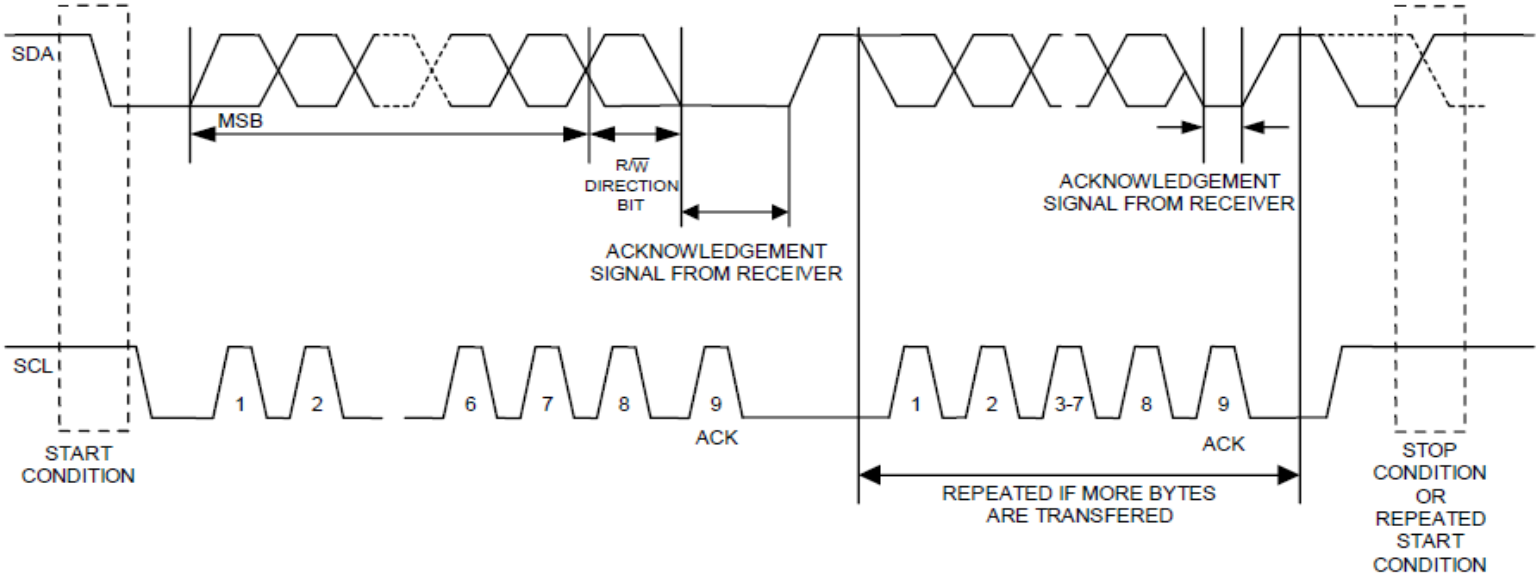
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

- OUT : Control the output level of SQW/OUT pin when SQWE is 0. The logic level on SQW/OUT pin is 1 when OUT is 1, and vice versa.
- SQWE : Enables the oscillator output when 1. The frequency of OUT is depends upon RS1 and RS0.
- RS1, RS0 : Control the frequency of square-wave output.

RS1	RS0	SQW/OUT OUTPUT	SQWE	OUT
0	0	1Hz	1	X
0	1	4.096kHz	1	X
1	0	8.192kHz	1	X
1	1	32.768kHz	1	X
X	X	0	0	0
X	X	1	0	1

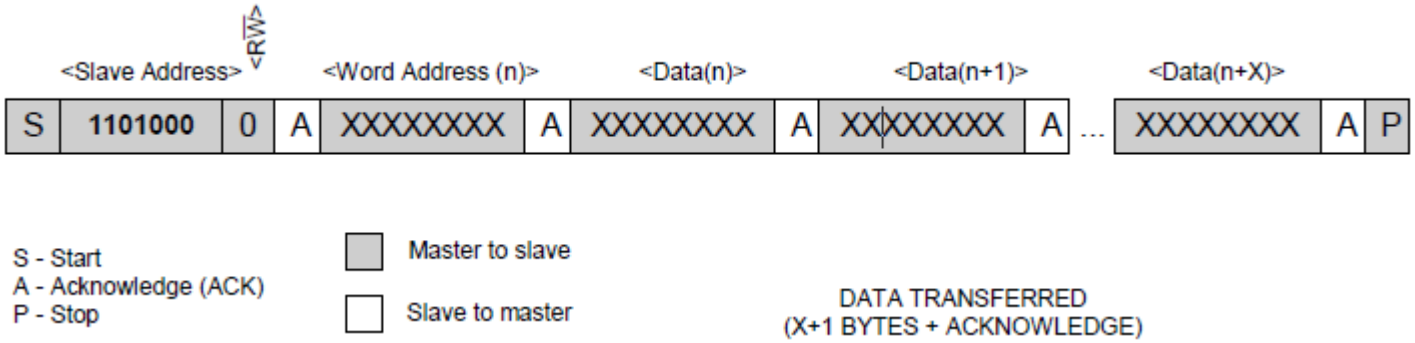
Timing Diagram

- Bus free : both SDA and SCL remains HIGH.
- START : SDA is on its falling edge, while SCL is HIGH.
- STOP : SDA is on its rising edge, while SCL is HIGH.
- Transfer : After a START condition, the data line should be stable for the duration of the HIGH period of SCL.
- Acknowledge: Each receiving device should generate an acknowledge after the reception of each byte on the extra 9 clock.

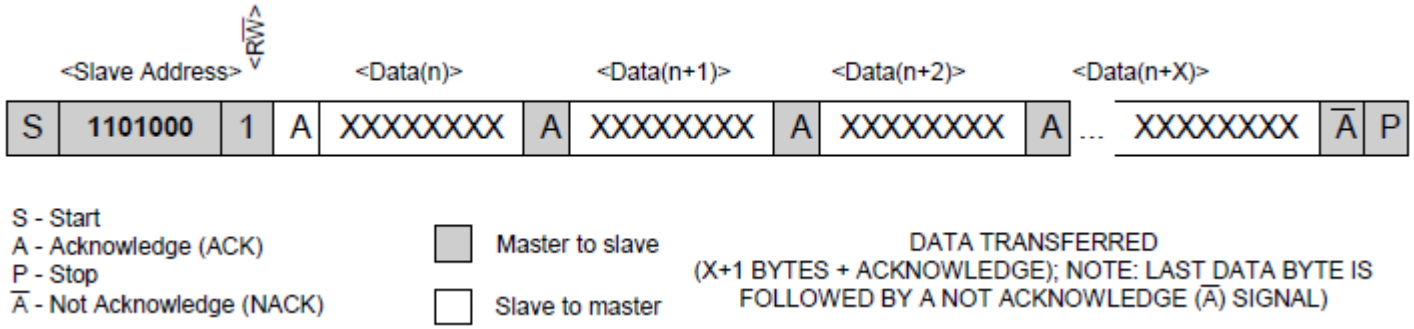


Transfer Mode

- Slave Receive Mode



- Slave Transmitter Mode



Backup Slides

Check list for a working I^2C in Lab 1

- Pull-up resistors
- Logic level shifter
- PIC registers related to I^2C
  - Set SCL, SDA pins as input
  - MSSP registers

Pull-up resistors

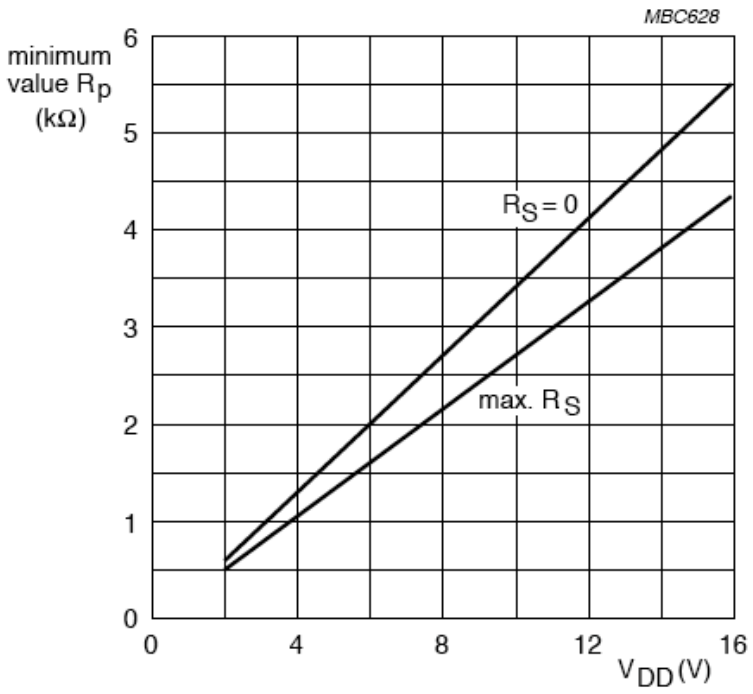


Fig.37 Minimum value of R_p as a function of supply voltage with the value os R_s as a parameter.

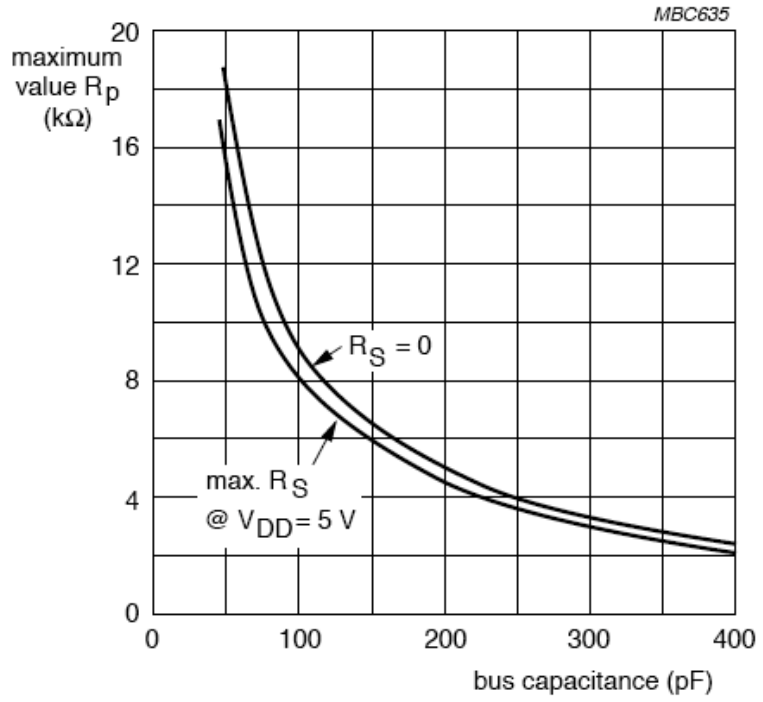
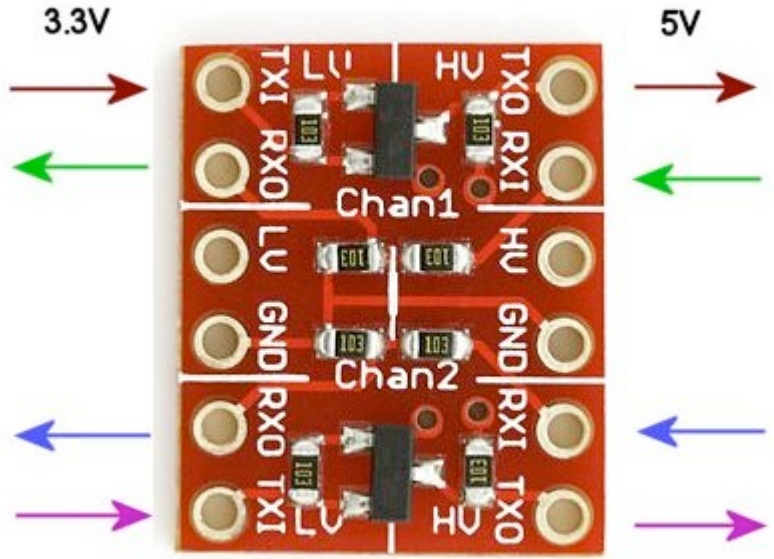


Fig.39 Maximum value of R_p as a function of bus capacitance for a Standard-mode I^2C-bus.

- I2C specification

Logic Level Shifter

- We need to use it as we have both a 5V (RTC) and a 3.3V (PIC) device



PIC registers

- PIC's oscillator configuration
  - OSCCON and OSCTUNE: to configure the PIC to run at 16MHz.
- TRISx (A,B,C, or D) registers to set the ports as inputs.
  - This is required by PIC's Master Synchronous Serial Port (MSSP)
  - MSSP supports both SPI and I^2C

PIC registers

- SSPCON1
  - SSPEN
  - SSPM
- SSPCON2
  - ACKSTAT
  - ACKDT
  - ACKEN
  - RCEN
  - PEN
  - SEN
- SSPSTAT
- SSPBUF
- SSPADD