I2C Overview

- Multi-mastered
- Send and receive
- Two wire (plus ground)
- Serial
## Terminology

<table>
<thead>
<tr>
<th>TERM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter</td>
<td>The device which sends data to the bus</td>
</tr>
<tr>
<td>Receiver</td>
<td>The device which receives data from the bus</td>
</tr>
<tr>
<td>Master</td>
<td>The device which initiates a transfer, generates clock signals and terminates a transfer</td>
</tr>
<tr>
<td>Slave</td>
<td>The device addressed by a master</td>
</tr>
<tr>
<td>Multi-master</td>
<td>More than one master can attempt to control the bus at the same time without corrupting the message</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Procedure to ensure that, if more than one master simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Procedure to synchronize the clock signals of two or more devices</td>
</tr>
</tbody>
</table>
Physical Interface

Fig.3 Connection of Standard- and Fast-mode devices to the I2C-bus.

wired-and configuration
Interfacing to the Bus

Electrical connection is fine. But the software is complex.
An alternative

Memory mapped device that handles bit level interface.
Only do byte level stuff in software
Bit Transfer

Sampled on both edges

Transmitter (master or slave)

Master

Fig. 4 Bit transfer on the I²C-bus.
Who gets to be master

The one who initiates a frame:

A frame is:

<Start><addr><data>…<data><Stop> OR
<Start><addr><data>…<data><R_Start><addr><data>…<Stop>

Fig. 5 START and STOP conditions.
An I2C Byte Transfer

MSB First

SDA

START or repeated START condition

S or Sr

master

SCL

1 2 7 8 9

slave

byte complete, interrupt within slave
clock line held low while interrupts are serviced

ACK

slave

acknowledgement signal from slave

ACK

slave

acknowledgement signal from receiver

slave

STOP or repeated START condition

MSC028

MSB

LSB

Tx Device

Rx Device

Fig. 6 Data transfer on the I2C-bus.
Addresses are 7 bits, LSB of address byte is Read/Write control. Determines whether master or slave becomes the transmitter.
Arbitration

It is up to the failed master to try again when the bus is not busy. Seems like high address devices are low priority receivers…but I haven’t read this anywhere.
Using the Bus Interface

PCF8584 resets to slave receiver mode
parallel bus interface determined by PCF8584 (80XX/68XXX)

START

reset minimum 30 clock cycles

power-on address line A0

A0 = HIGH

A0 = LOW

send byte 80H

send byte 55H

send byte A0H

send byte 1CH

send byte C1H

delay: wait a time equal to the longest I²C message to synchronize B8-bit. (multimaster systems only)

A0 = HIGH

A0 = LOW

initialization of PCF8584 completed

END

A0 = HIGH enables data transfer to/from register S1

A0 = LOW Access to all other registers defined by the bit pattern in register S1

Loads byte 80H into register S1, i.e. next byte will be loaded into register S0 (own address register); serial interface off.

Loads byte 55H into register S0; effective own address becomes AAH.

Loads byte A0H into register S1, i.e. next byte will be loaded into the clock control register S2.

Loads byte 1CH into register S2; system clock is 12 MHz; SCL = 90 kHz.

Loads byte C1H into register S1; register enable serial interface, set I²C-bus into idle mode: SDA and SCL are HIGH. The next write or read operation will be to/from data transfer register S0 if A0 = LOW.

On power-on, if an PCF8584 node is powered-up slightly after another node has already begun an I²C-bus transmission, the bus busy condition will not have been detected. Thus, introducing this delay will insure that this condition will not occur.
Slave Flow

START

read byte from S1 register

addressed as slave (AAS = 1?)

no

A0 = HIGH

Check whether 'addressed as slave'

yes

Check that 'own address' has arrived correctly

read byte from S1 register

PIN bit = 0?

no

A0 = LOW

Read incoming address to determine if the R/W bit is 0 or 1 This will differentiate between slave receiver or slave transmitter modes.

yes

read byte from S0 register

R/W = 1

read or write? (LSB = 1 or 0?)

SLAVE TRANSMITTER MODE

A0 = HIGH

SLAVE RECEIVER MODE

R/W = 0
Slave Receiver Mode

- **Slave Transmitter Mode**:
  - Read byte from S1 register
  - PIN bit = 0?
    - yes
      - negative ACK received? (LRB = 1?)
        - yes
          - write data to S0 register
        - no
      - write last data byte to S0 register
    - no
      - END TX

- **Slave Receiver Mode**:
  - Read byte from S1 register
  - PIN bit = 0?
    - yes
      - STOP detected? (STS = 1?)
        - yes
          - END RX
        - no
      - read data from S0 register
    - no
      - read last data byte from S0 register

Read incoming address to determine if the R/W bit is 0 or 1. This will differentiate between slave receiver or slave transmitter modes.