The I2C Protocol

This document describes the I2C protocol. Or will, when it is finished :-)

Key to symbols

S	Start condition
P	Stop condition
Rd/Wr (1 bit)	Read/Write bit. Rd equals 1, Wr equals 0.
A, NA (1 bit)	Acknowledge (ACK) and Not Acknowledge (NACK) bit
Addr (7 bits)	I2C 7 bit address. Note that this can be expanded as usual to get a 10 bit I2C address.
Comm (8 bits)	Command byte, a data byte which often selects a register on the device.
Data (8 bits)	A plain data byte. Sometimes, I write DataLow, DataHigh for 16 bit data.
Count (8 bits)	A data byte containing the length of a block operation.
[..]	Data sent by I2C device, as opposed to data sent by the host adapter.

Simple send transaction

Implemented by i2c_master_send():

```
S Addr Wr [A] Data [A] Data [A] ... [A] Data [A] P
```

Simple receive transaction

Implemented by i2c_master_recv():

```
S Addr Rd [A] [Data] A [Data] A ... A [Data] NA P
```

Combined transactions

Implemented by i2c_transfer().

They are just like the above transactions, but instead of a stop condition P a start condition S is sent and the transaction continues. An example of a byte read, followed by a byte write:

```
S Addr Rd [A] [Data] NA S Addr Wr [A] Data [A] P
```

Modified transactions

The following modifications to the I2C protocol can also be generated by setting these flags for I2C messages. With the exception of I2C_M_NOSTART, they are usually only needed to work around device issues:

I2C_M_IGNORE_NAK:

> Normally message is interrupted immediately if there is [NA] from the client. Setting this flag treats any [NA] as [A], and all of message is sent. These messages may still fail to SCL lo->hi timeout.

I2C_M_NO_RD_ACK:

> In a read message, master A/NA bit is skipped.

I2C_M_NOSTART:

> In a combined transaction, no 'S Addr Wr/Rd [A]' is generated at some point. For example, setting I2C_M_NOSTART on the second partial message generates something like:
>
> ```
> S Addr Rd [A] [Data] NA Data [A] P
> ```
>
> If you set the I2C_M_NOSTART variable for the first partial message, we do not generate Addr, but we do generate the start condition S. This will probably confuse all other clients on your bus, so don't try this.
>
> This is often used to gather transmits from multiple data buffers in system memory into something that appears as a single transfer to the I2C device but may also be used between direction changes by some rare devices.

I2C_M_REV_DIR_ADDR:

> This toggles the Rd/Wr flag. That is, if you want to do a write, but need to emit an Rd instead of a Wr, or vice versa, you set this flag. For example:
>
> ```
> S Addr Rd [A] Data [A] Data [A] ... [A] Data [A] P
> ```

I2C_M_STOP:

Force a stop condition (P) after the message. Some I2C related protocols like SCCB require that. Normally, you really don't want to get interrupted between the messages of one transfer.