Talking to the I2C slave

The i2c slave is a read/write device that operates under the control of the i2c master. The slave implementation follows (loosely) the assembly code provided in Microchip application note (AN734). The slave code is written for the 16F877, but should be easily adapted to other similar devices. This i2c master functionality and code is as described in the documentation supplied with BoostC

The i2c slave appears to the master as a device address that has two internal buffers. The buffers function as follows:

> Message Buffer **(mesg)**: The master sends or receives the data to/from this location within the slave device. This buffer is 32 bytes long in the example.
>
> Address Buffer **(i2c_addr)**: This is the address of a location within the Message Buffer. i.e **i2c address** zero is the start of the Message Buffer. The master can only write to this location. This buffer is 32 bytes long in the example.

Additionally there are two parameters in the function call that specify the device identifier (address) as well as the size of the data to be placed into the message buffer These are as follows:

> Hardware address **(hw_address)**: This is the address of the particular I2C Device. Set to 0x0C for this slave example.
>
> String Length **strlen(mesg)**: Specifies how many bytes to read or write from the Message Buffer and typically uses the strlen() function, but it could also be a simple numeric value.

To write to a particular location in the slave Message Buffer:

```
char *mesg = "ABCD";
char hw_address = 0x0C;

write_XEE( mesg  hw_address, 3, strlen(mesg+1)
```

Master sends the mesg string ABCD plus it's terminating NULL to byte 3 of the slave device Message Buffer. The slave device address is 0x0C.

To read from a particular location in the slave Message Buffer:

```
char *data[5];
char hw_address = 0x0C;

read_XEE(data, hw_address, 3, 5)

Master reads the contents of the slave device 0x0C (ABCD)
starting at byte 3 of the slave Message Buffer and includes the
terminating NULL.
```

The size of the data sent to the slave should, of course, not exceed the size of the Slave Message Buffer. Du-uh!

In summary; the i2c slave looks like a chunk of memory to which data can be read or written. The slave receives data from the master and places it into the Message Buffer. Data to be sent back to the master is placed into the Message Buffer by the slave. The slave can not signal the master that data is available or that it has been acted upon.