The I^2C Bus Operation

The I^2C (Inter IC) bus is a simple bidirectional serial bus that supports multiple masters and slaves. It consists of only two lines; a serial bidirectional data line (SDA) and a serial bidirectional clock line (SCL). Within the I^2C bus specifications, a standard mode with a maximum clock rate of 100k Hz and a fast mode with a maximum clock rate of 400k Hz are defined.

Each device connected to the I^2C bus is software addressable by a unique address, and a simple master/slave relationship exists at all times among the devices. The device that controls the sending and receiving of messages by controlling the bus access is the *master*. Devices that are controlled by the master are the *slaves*. Both the master and the slave can send and receive messages. A device that sends data onto the bus is referred to as the *transmitter* and a device receiving data is referred to as the *receiver*.

More than one master and more than one slave can co-exist on the same I^2C bus. The bus, however, is always controlled by a single master at any one time, and is responsible for generating the serial clock (SCL) and controlling the bus access by initiating and terminating a message transfer.

Our system consists of only one master (the controller that is implemented on the Arduino) and one slave (the Maxim DS1307 real-time clock chip) as shown in Figure 1. Both the SDA and SCL lines connecting between the master and the slave are open-drained, and must be pulled up to Vcc with a 5.6 kΩ resistor. Regardless of how many devices are connected to the bus, only one pull-up resister is needed per line. This implementation of the master controller does not follow the full specifications of the I^2C protocol, so if you use this for other I^2C connections, the master controller might halt in the error state.

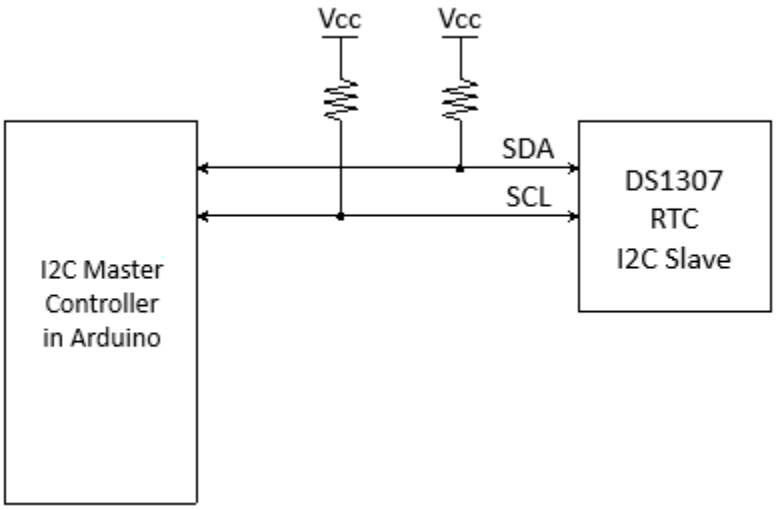

Figure 1 I^2C bus system with the I^2C master controller implemented in an Arduino and a real-time clock chip acting as the slave.

The I^2C bus is idle when both SCL and SDA are at a logic 1 level. The master initiates a data transfer by issuing a START condition, which is a high to low transition on the SDA line while the SCL line is high as shown in Figure 2(a). The bus is considered to be busy after the START condition. After the START condition, a slave address is sent out on the bus by the master. This address is 7 bits long followed by an eighth bit that is a data direction bit $(R/\overline{W})$, where a 0 indicates a write from the master to the slave, and a 1 indicates a read from the slave to the master. The master, who is controlling the SCL line, will send out the bits on the SDA line, one bit per clock cycle of the SCL line, with the most significant bit sent out first. The value on the SDA line can be changed only when the SCL line is at a low.

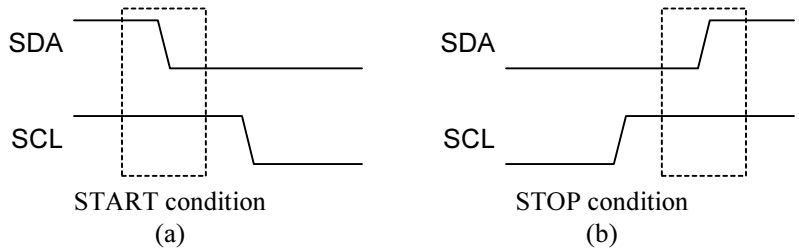

Figure 2 The START (a) and STOP (b) conditions are both initiated by the master. The START condition happens when the SDA line changes from a high to a low while the SCL line is at a high. The STOP condition happens when the SDA line changes from a low to a high while the SCL line is at a high. These are the only two situations where the SDA line can change when SCL is at a high.

The slave device whose address matches the address that is being sent out by the master will respond with an acknowledgment bit on the SDA line by pulling the SDA line low during the ninth clock cycle of the SCL line as shown in Figure 3. The direction bit $(R/\overline{W})$ determines whether the master or the slave will be the transmitter in the subsequent data transmission after the sending of the slave address.

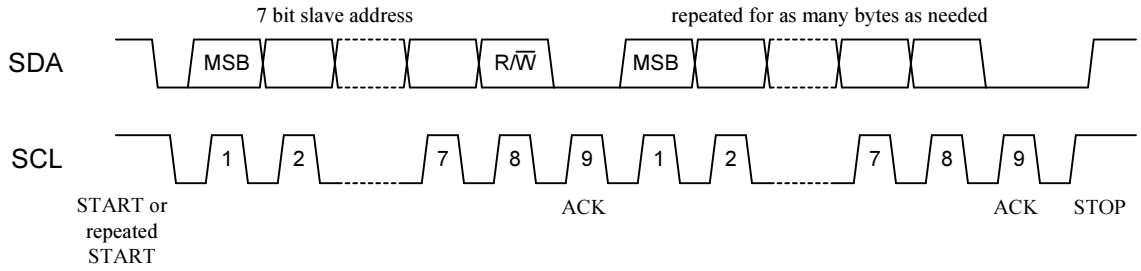

Figure 3 The master initiates data transmission by sending the START condition. For every 8 bits of data transmitted, the receiver sends an acknowledgment during the ninth clock cycle by pulling SDA low. The only exception is when the master-receiver wants to end the data transmission in which case the master (who is the receiver) will not acknowledge by keeping SDA high. Data transmission is terminated by the master sending the STOP condition.

Every byte put on the SDA line for transmission must be 8-bits long with the most significant bit first. Except for the START and STOP conditions, the SDA line must not changed when the SCL line is high. The number of bytes that can be transmitted is unrestricted. Each byte has to be followed by an acknowledge bit. The master generates the acknowledge-related clock pulse. The transmitter releases the SDA line (sets it to high impedance) during the acknowledge clock pulse, and the receiver must pull down the SDA line during the acknowledge clock pulse to acknowledge the receipt of the byte. The one exception is when a master-receiver is involved in a transfer. In this case the master-receiver must signal the end of data to the slave-transmitter by not generating an acknowledgement on the last byte clocked out of the slave.

To signal the end of data transfer, the master sends a STOP condition by pulling the SDA line from low to high, while the SCL line is at a high as shown in Figure 2(b). Alternatively, instead of sending a STOP condition, the master can send a repeated START condition so that it can change the direction of the data transmission without having to release the bus.

Figure 4(a) shows the scenario in which the master writes 1 byte of data to the slave (i.e., the master is the transmitter and the slave is the receiver). The master initiates the data transfer by first issuing the START condition followed by the 7-bit slave address plus the write (0) bit. After receiving an acknowledgment from the slave, the master sends the register number to let the slave know which register the following data should be written into. The slave responds with an acknowledgment, and the master sends the data byte to the slave. After the slave acknowledges receipt of the data byte, the master sends the STOP condition.

Figure 4(b) shows the scenario in which the master reads 1 byte of data from the slave (i.e., the master is the receiver and the slave is the transmitter). The master initiates the data transfer by issuing the START condition

followed by the 7-bit slave address plus the write (0) bit. Although the master wants to receive a byte, it first needs to send the register address byte to let the slave know which register the master wants to read from. After receiving an acknowledgment from the slave, the master sends the register number to the slave, and the slave responds again with an acknowledgment. This time the master has to do a repeated START condition because it needs to change the data direction from a write to a read. The repeated START is followed by the slave address again, but this time with the read (1) bit instead. The slave acknowledges and then sends the data byte from the addressed register to the master. This time, because the master is the receiver, the master has to acknowledge receipt of the data byte. If the master wants to receive more data bytes from the slave, it sends a 0 to acknowledge it. If the master doesn't want to receive any more bytes, it won't acknowledge by keeping SDA at a high. Finally, the master sends the STOP condition.

To summarize the master-transmitter and master-receiver scenarios, for both cases, the master first sends the 7-bit slave address and the write bit, followed by the register number to access. In the master-transmitter scenario, the master can immediately send out the data byte because the data direction is still a write. For the master-receiver scenario, the master has to do a repeated START and resend the slave address with the read bit in order to change the direction of the data transmission from a write to a read. After this, the master can receive a byte of data from the slave from the given register number.

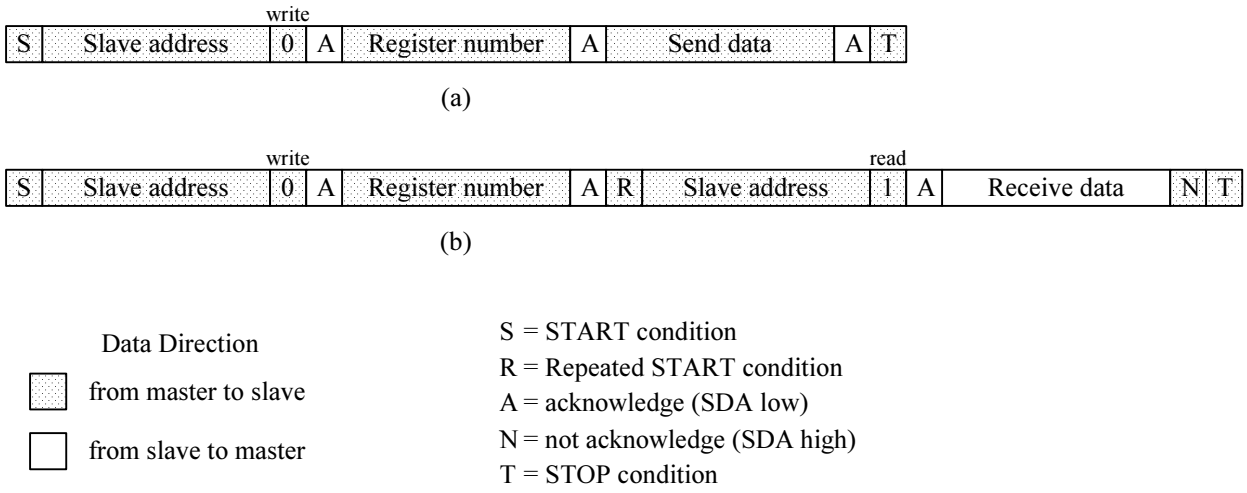

Figure 4 (a) The master transmits 1 byte of data to the slave. (b) The master receives 1 byte of data from the slave.