1. **I2C Interface**
2. **1 I2C BUS Protocol**

I2C Bus is a 2-wire serial bus protocol which is widely used in consumer electronics, telecommunications and industrial electronics.

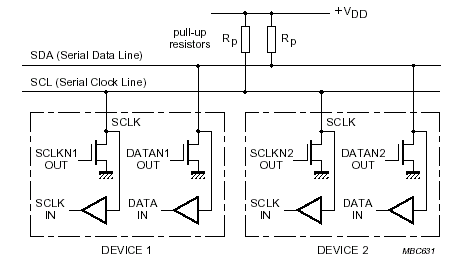
Here are the features of I2C.

* Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL).
* Each device connected to the bus is software addressable by a unique address and simple.
* Master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers.
* Serial, 8-bit oriented, bi-directional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, or up to 3.4 Mbit/s in the High-speed mode.
* On-chip filtering rejects spikes on the bus data line to preserve data integrity.

1. **2 Connection of I2C Bus**

Both SDA and SCL are bi-directional lines, connected to a positive supply voltage via a current-source or pull-up resistor (see Figure 3-2). When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function. Data on the I2C bus can be transferred at rates of up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, or up to 3.4 Mbit/s in the High-speed mode. The number of interfaces connected to the bus is solely dependent on the bus capacitance limit of 400 pF.

Figure 3-2 I2C BUS Connection



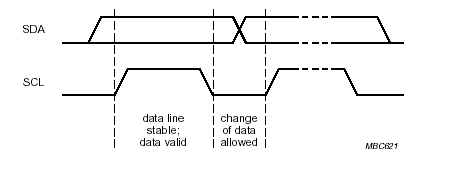
1. **3 Bit Transfer on I2C BUS**

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I2C bus, the levels of the logical ‘0’ (LOW) and ‘1’ (HIGH) are not fixed and depend on the associated level of VDD. One clock pulse is generated for each data bit transferred.

1. **3. 1 Data Validity**

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (Figure 3-3).

Figure 3-3 Bit Transfer on I2C BUS



1. **3. 2 START and STOP Conditions**

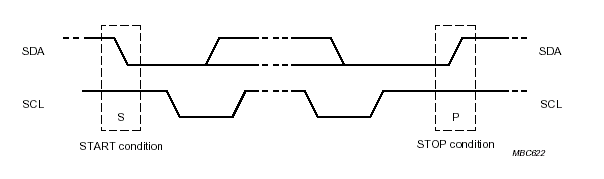
Within the procedure of the I2C-bus, unique situations arise which are defined as START (S) and STOP (P) conditions (Figure 3-4).

A HIGH to LOW transition on the SDA line while SCL is HIGH is one such unique case. This situation indicates a START condition.

A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition.

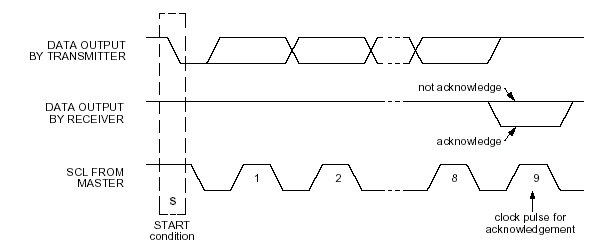
Figure 3-4 START and STOP Conditions



1. **3. 3 Acknowledge Bit on I2C BUS**

For the accuracy of I2C transfer, acknowledge bit is needed. Only the received side sends acknowledge bit, the next data’s transfer can be continued (Figure 3-5).

Figure 3-5 Acknowledge on I2C BUS



1. **4 Master Mode I2C Function Block**

The master mode I2C block is connected to the ISDA /ISCL pins. It’s speed can be selected to 10KHz-400KHz by S/W setting the clk\_div\_cnt（0x1f030034） control register. The software program can access the external I2C device through this interface. A summary of master I2C access is illustrated as follows.

1. **4. 1 Transmit Mode**

* Setting slave\_addr, if not its initial value is 0xf.
* Setting NWord (I2C master transmit data length)
* Setting transmit data to data\_2\_iicm0/1;
* Setting cpu\_command[3:0] = 0x01, I2C Master send Start condition and data transmit starts.
* Once finish the transmission of the last byte data, master\_nstop interrupt is occurred if master\_nstop\_en is set to “1”.

I2C Master transmit mode (Figure 3-9)

Figure 3-9 I2C Master Transmit Format

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | Slave\_addr | W | A | DATA0 | A |  | DATAn | A | P |

1. **4. 2 Receive Mode**

* Setting slave\_addr, if not its initial value is 0xf
* Setting NWord (I2C master Receive data length)
* Setting cpu\_command[3:0] = 0x02, I2C Master send Start condition and data receive starts.
* Every data received is sent to iicm\_2\_data0/1
* Once finish the reception of the last byte data, master\_nstop interrupt is occurred if master\_nstop\_en is set to “1”.

I2C Master receive mode (Figure 3-10)

Figure 3-10 I2C Master Receive Format

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | Slave\_addr | R | A | DATA0 | A |  | DATAn | A | P |

1. **4. 3 Restart Mode**

* Setting slave\_addr, if not its initial value is 0xf.
* Setting NWord (I2C master Receive data length).
* Setting mast\_read\_addr[15:0] to decide the start address that IICM received data is from.
* Setting cpu\_cmd[3:0] = 0x07 ( mast\_read\_addr low 8-bit active) or 0x0F (mast\_read\_addr 16-bit active), I2C Master send Start condition and data receive starts.
* Every data received is sent to iicm\_2\_data0/1
* Once finish the reception of the last byte data, master\_nstop interrupt is occurred if master\_nstop\_en is set to “1”.

I2C Master Restart mode 1 (Figure 3-11)

Figure 3-11 I2C Master Restart Format 1 (cpu\_command[3:0] = 0x07)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | Star\_addr | W | A | Mast\_ read\_addr[7:0] | A |
|  |  |  |  |  |  |  |  |  |  |  |  |
| RS | Star\_addr | R | A | DATAO | A |  | DATAn | A | P | | |

I2C Master Restart mode 2 (Figure 3-12)

Figure 3-12 I2C Master Restart Format 2 (cpu\_command[3:0] = 0x0F)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S | Star\_addr | W | A | Mast\_ read\_addr[7:0] | A | Mast\_ read\_addr[15:8] | A |
|  |  |  |  |  |  |  | |  | |  |  |  |
| RS | Star\_addr | R | A | DATAO | A |  | | DATAn |  | P | | |

1. **4. 4 I2C Master Interrupt**

The XBR820 has 2 kinds of I2C Master interrupt in I2C Master mode: master\_nstop, Master\_nack. The XBR820 generates master\_nstop interrupt at the end of the last data transmit/receive cycle, generates Master\_nack interrupt in transmit mode.

1. **5 SlaveB Mode**

The SlaveB mode I2C block is connected to HSDA and HSCL pins, receive/transmit data using I2C protocol.

1. **5. 1 Receive Mode**

Below shows the steps to perform I2C receive function:

A: one byte data receive mode

* Setting slavedev, if not its initial value is 0xaa.
* Setting en\_slaveb to “1”(initial value)
* The first data received is sent to SlaveB\_Data.

SlvB one byte data receive mode (Figure 3-6)

Figure 3-6 SlvB Receive Format (one byte)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S | SlvB ID | W | A | DATA |  | P |

B : sequence (more than one byte) data receive mode

* Setting slavedev, if not its initial value is 0xaa.
* Setting en\_slaveb to “1”(initial value)
* The first data received is sent to slaveb\_addr as the register address, slave\_addrb interrupt is occurred.
* The data from second byte received is sent to slaveb\_data, the end of every data receive cycle Slave\_rw\_int interrupt is occurred if rel\_slb\_rw\_int bit is cleared to “0”.

NOTE: Because of the constraint of the time sequence, in SlvB receive mode, the length of the running time of interrupt response program has some limitation. Please make sure that before next byte of data transmission is over, the interrupt response program for this byte of data is finished.

SlvB sequence data receive mode (Figure 3-7)

Figure 3-7 SlvB Receive Format

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | SlvB ID | W | A | slaveb\_addr | A | DATA0 | A |  | DATAn | A | P |

1. **5. 2 Transmit Mode**

Below shows the steps to perform I2C transmit function:

* Setting slavedev, if not its initial value is 0xaa.
* Setting en\_slaveb to “1”(initial value)
* Set SlvB transmit mode, data wrote in slaveb\_data\_2\_iic transmitted, and the end of every data transmit cycle Slave\_rw\_int interrupt is occurred if slave\_rw bit is set to “1”.

SlvB transmit mode (Figure 3-8)

Figure 3-8 SlvB Transmit Format

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | SlvB ID | R | A | DUMMY DATA | A | DUMMY DATA | A | DATA0 | A |  | DATAn | A | P |

Note: Please ignore the first and second transmitted data.

1. **5. 3 SlvB Interrupt**

The XBR820 has 4 kinds of SlvB interrupt in I2C SLAVEB mode : Slave\_rw\_int, Slave\_addr, Slave\_stop and Slave\_nack. The XBR820 generates Slave\_rw\_int interrupt at the end of every data transmission or reception cycle (in transmit mode Slave\_rw is set to 1, in receive mode Slave\_rw is cleared to 0), generates Slave\_addr interrupt at the end of first data receive cycle, generates Slave\_stop interrupt if HSCL, HSDA match stop condition, generates Slave\_nack interrupt in transmit mode.