I2C PROTOCOL IN FPGA USING VHDL

MAIN PROJECT INTERIM REPORT

LITERATURE SURVEY

I2C PROTOCOLS

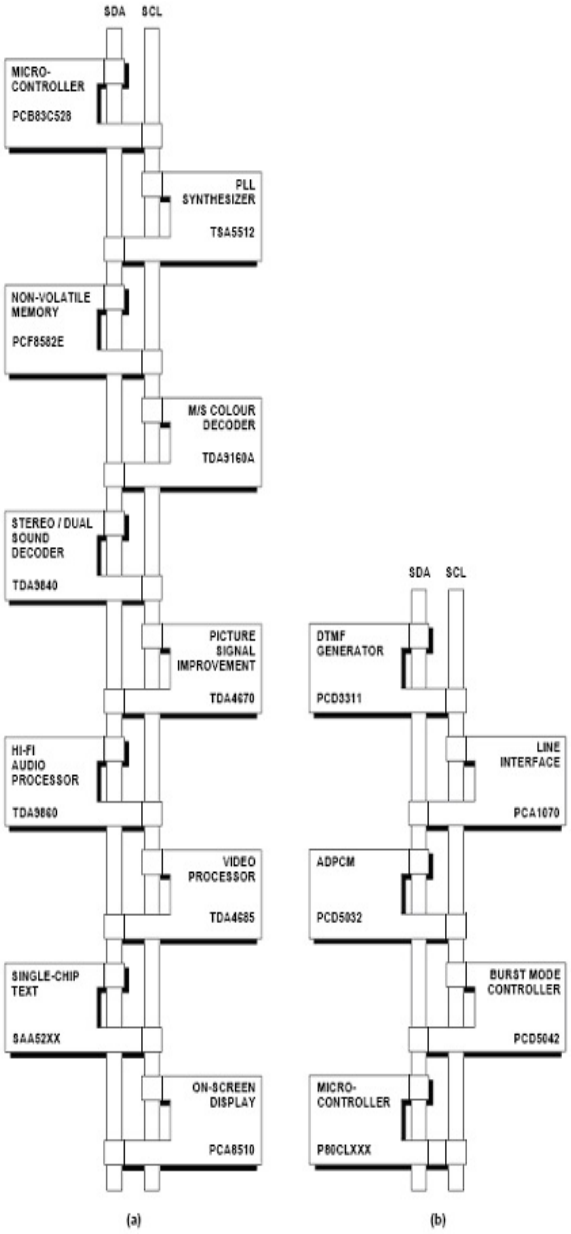
The I2C bus is a popular serial, two-wire interface used in many systems because of its low overhead. The two-wire interface minimizes interconnections so ICs have fewer pins, and the number of traces required on printed circuit boards is reduced. Capable of 100 KHz operation, each device connected to the bus is software addressable by a unique address with a simple Master/Slave protocol.

The I2C bus consists of two wires, serial data (SDA) and serial clock (SCL), which carry information between the devices connected to the bus. Both the SDA and SCL lines are bidirectional lines, connected to a positive supply voltage via a pull-up resistor. 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 in order to perform the wired-AND function.

Each device on the bus has a unique address and can operate as either a transmitter or receiver. In addition, devices can also be configured as Masters or Slaves. A Master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any other device that is being addressed is considered a Slave. The I2C protocol defines an arbitration procedure that insures that if more than one Master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted. The arbitration and clock synchronization procedures defined in the I2C specification are supported by the I2C Controller. The I2C-bus supports any IC fabrication process (NMOS, CMOS, bipolar). Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

Data transfers on the I2C bus are initiated with a START condition and are terminated with a STOP condition. Normal 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 SCL is Low. The START condition is a unique case and is defined by a High-to-Low transition on the SDA line while SCL is High. Likewise, the STOP condition is a unique case and is defined by a Low-to-High transition on the SDA line while SCL is High. The definitions of data, START, and STOP insure that the START and STOP conditions will never be confused as data.

Each data packet on the I2C bus consists of eight bits of data followed by an acknowledge bit so one complete data byte transfer requires nine clock pulses. Data is transferred with the most significant bit first (MSB). The transmitter releases the SDA line during the acknowledge bit and the receiver of the data transfer must drive the SDA line low during the acknowledge bit to acknowledge receipt of the data. If a Slave-receiver does not drive



Two examples of I²C-bus applications: (a) a high performance highly-integrated TV set (b) DECT cordless phone base-station.

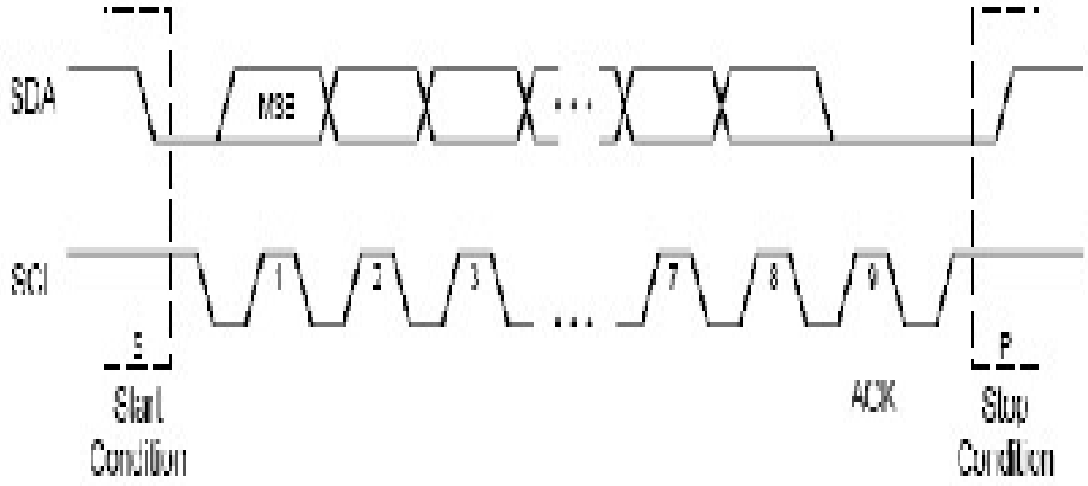


Figure 2

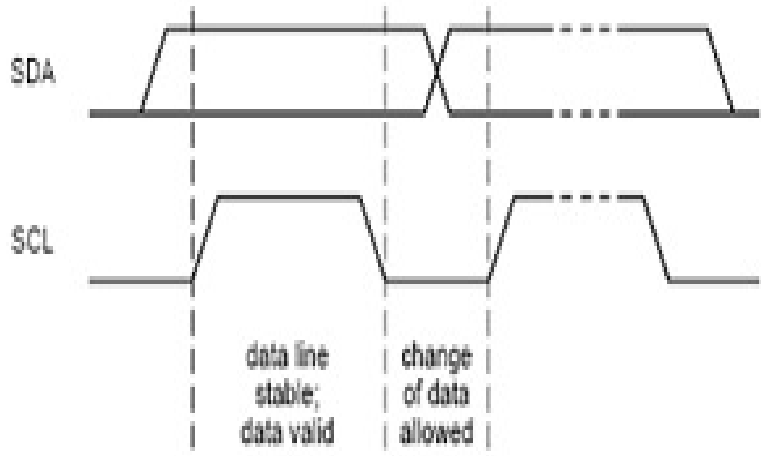
the SDA line Low during the acknowledge bit, this indicates that the Slave-receiver was unable to accept the data and the Master can then generate a STOP condition to abort the transfer. If the Master-receiver does not generate an acknowledge, this indicates to the Slave-transmitter that this byte was the last byte of the transfer.

Standard communication on the bus between a Master and a Slave is composed of four parts: START, Slave address, data transfer, and STOP. The I2C protocol defines a data transfer format for both 7-bit and 10-bit addressing. The implementation of the I2C controller supports the seven-bit address format. After the START condition, a Slave address is sent. This address is seven bits long followed by an eighth-bit which is the read/write bit. A 1 indicates a request for data (read) and a 0 indicates a data transmission (write). Only the Slave with the calling address that matches the address transmitted by the Master responds by sending back an acknowledge bit by pulling the SDA line Low on the ninth clock.

Once successful Slave addressing is achieved, the data transfer can proceed byte-by-byte as specified by the read/write bit. The Master can terminate the communication by generating a STOP signal to free the bus. However, the Master may generate a START signal without generating a STOP signal first. This is called a repeated START.

MASTERS AND SLAVES

The I2C-bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. To avoid the chaos that might ensue from such an event , an arbitration procedure has been developed. This procedure relies on the wired-AND connection of all I2C interfaces to the I2C-bus. If two or more masters try to put information onto the bus, the first to produce a one when the other produces a zero will lose the arbitration. The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired-AND connection to



Bit transfer on the I²C-bus.

Figure 3

the SCL line.

Generation of clock signals on the I2C-bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding-down the clock line or by another master when arbitration occurs.

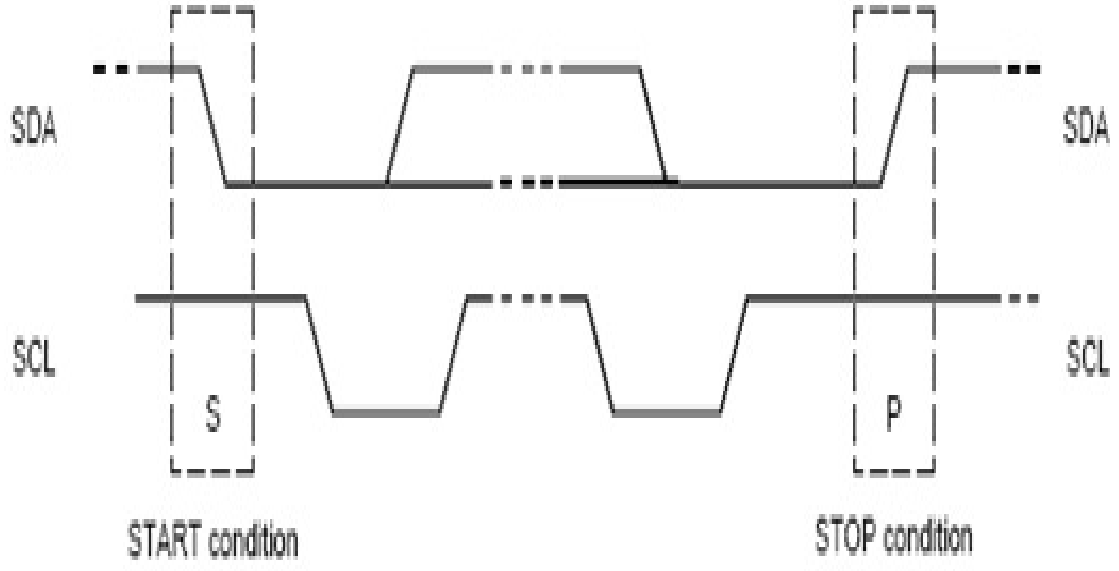
BIT TRANSFER

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.

START AND STOP CONDITIONS

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

The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical.



START and STOP conditions.

TRANSFERING DATA

BYTE FORMAT

Every byte put on the SDA line must be 8-bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first. If a slave can not receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL LOW to force the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCL.

ACKNOWLEDGE

Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse. The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse. Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received.

When a slave does not acknowledge the slave address (for example, it is unable to receive or transmit because it is performing some real-time function), the data line must be left HIGH by the slave. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer. If a slave-receiver does acknowledge the slave address but, some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave generating the not-acknowledge on the first byte to follow. The slave leaves the data line HIGH and the master generates a STOP or a repeated START condition. If a master-receiver is involved in a transfer, it must signal the end of data to the slave- transmitter

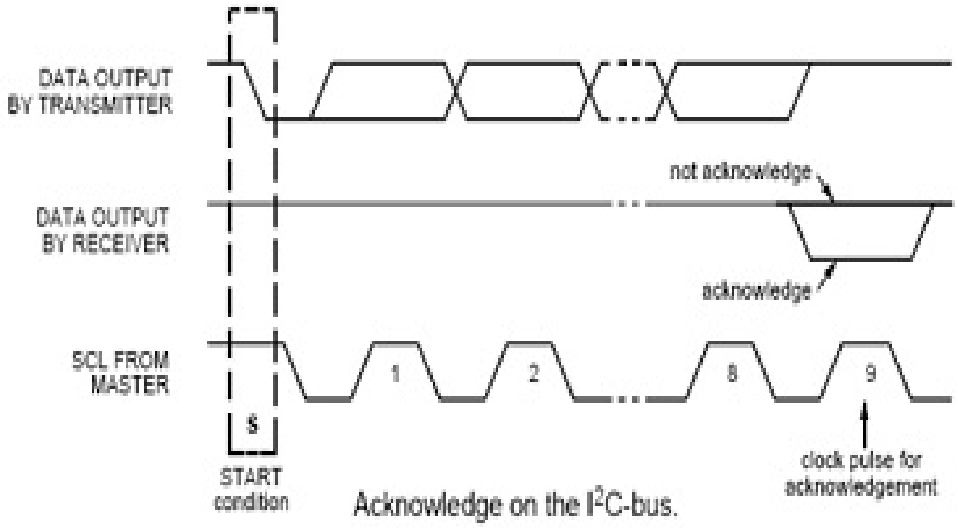


Figure 5

by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition.beginfigure

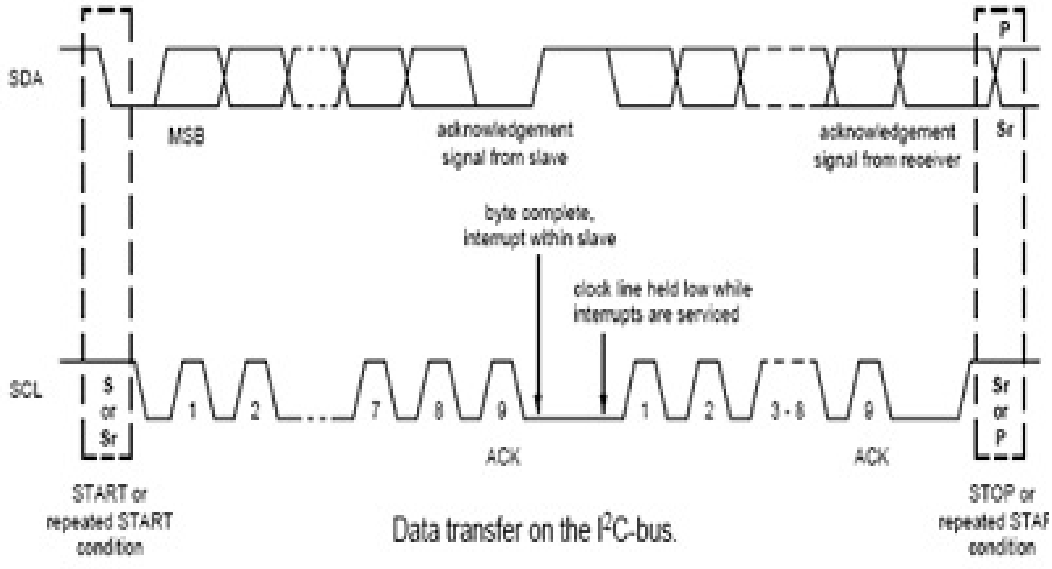
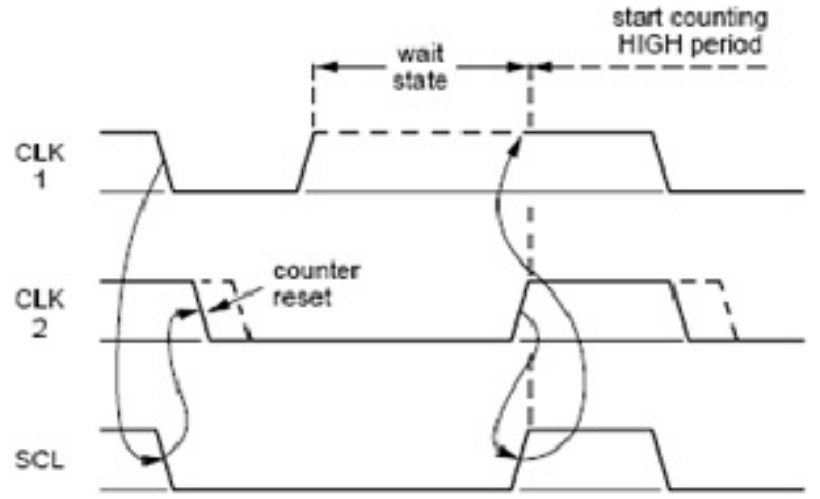


Figure 6

ARBITRATION AND CLOCK GENERATION

SYNCHRONIZATION

All masters generate their own clock on the SCL line to transfer messages on the I2C-



Clock synchronization during the arbitration procedure.

Figure 7

bus. Data is only valid during the HIGH period of the clock. A defined clock is therefore needed for the bit-by-bit arbitration procedure to take place. Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line will cause the devices concerned to start counting off their LOW period and, once a device clock has gone LOW, it will hold the SCL line in that state until the clock HIGH state is reached (see Fig.8). However, the LOW to HIGH transition of this clock may not change the state of the SCL line if another clock is still within its LOW period. The SCL line will therefore be held LOW by the device with the longest LOW period. Devices with shorter LOW periods enter a HIGH wait-state during this time. When all devices concerned have counted off their LOW period, the clock line will be released and go HIGH. There will then be no difference between the device clocks and the state of the SCL line, and all the devices will start counting their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronized SCL clock is generated with its LOW period determined by the device with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.When all devices concerned have counted off their LOW period, the clock line will be released and go HIGH. There will then be no difference between the device clocks and the state of the SCL line, and all the devices will start counting their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronized SCL clock is generated with its LOW period determined by the device with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.

ARBITRATION

A master may start a transfer only if the bus is free. Two or more masters may

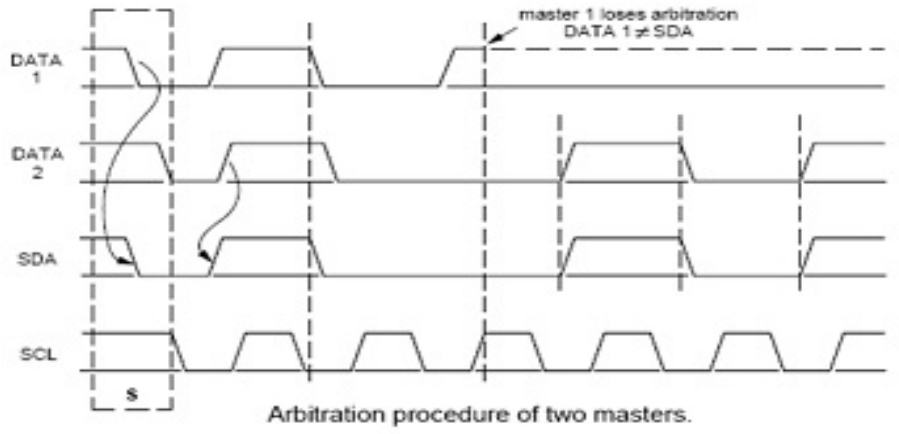


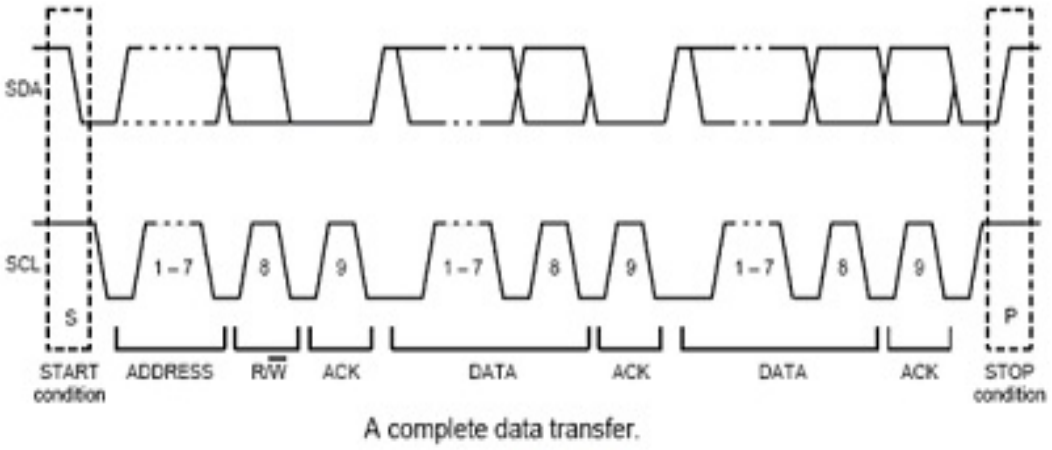
Figure 8

generate a START condition within the minimum hold time of the START condition which results in a defined START condition to the bus. Arbitration takes place on the SDA line, while the SCL line is at the HIGH level, in such a way that the master which transmits a HIGH level, while another master is transmitting a LOW level will switch off its DATA output stage because the level on the bus doesnot correspond to its own level. Arbitration can continue for many bits. Its first stage is comparison of the address bits. If the masters are each trying to address the same device, arbitration continues with comparison of the data-bits if they are master-transmitter, or acknowledge-bits if they are master-receiver. Because address and data information on the I2C-bus is determined by the winning master, no information is lost during the arbitration process. A master that loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration. If a master also incorporates a slave function and it loses arbitration during the addressing stage, it is possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave mode. Figure shows the arbitration procedure for two masters. Of course, more may be involved (depending on how many masters are connected to the bus). The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a HIGH output level is then connected to the bus. This will not affect the data transfer initiated by the winning master.

Since control of the I2C-bus is decided solely on the address or master code and data sent by competing masters, there is no central master, nor any order of priority on the bus. Special attention must be paid if, during a serial transfer, the arbitration procedure is still in progress at the moment when a repeated START condition or a STOP condition is transmitted to the I2C-bus.If it is possible for such a situation to occur, the masters involved must send this repeated START condition or STOP condition at the same position in the format frame.

In other words, arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition.



A complete data transfer.

Figure 9

Slaves are not involved in the arbitration procedure.

USE OF THE CLOCK SYNCHRONIZING AS A HANDSHAKE

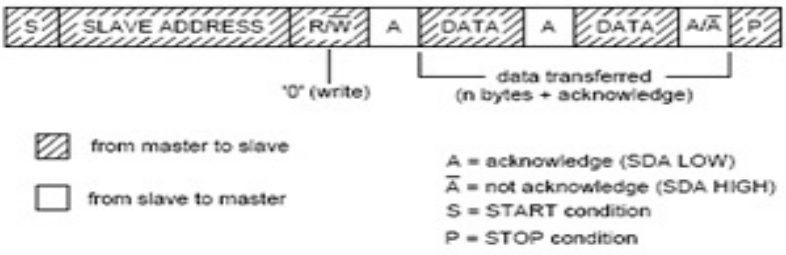
In addition to being used during the arbitration procedure, the clock synchronization mechanism can be used to enable receivers to cope with fast data transfers, on either a byte level or a bit level. On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Slaves can then hold the SCL line LOW after reception and acknowledgment of a byte to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure. On the bit level, a device such as a microcontroller with or without limited hardware for the I2C-bus can slow down the bus clock by extending each clock LOW period. The speed of any master is thereby adapted to the internal operating rate of this device.

ADRESSING BY 7 BITS

Data transfers follow the format shown in Figure. After the START condition (S), a slave address is sent. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/W) - a zero indicates a transmission (WRITE), a one indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

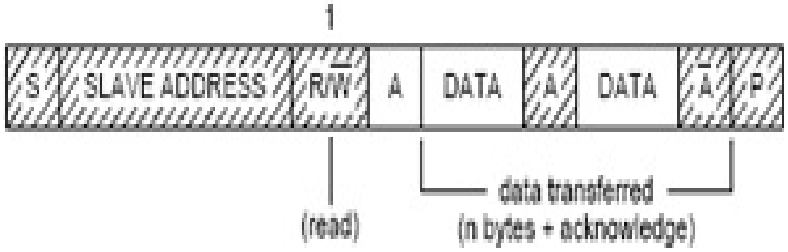
Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed.
- Master reads slave immediately after first byte. At the moment of the first acknowledge, the master- transmitter becomes a master- receiver and the slave-receiver becomes a slave-transmitter. This first acknowledge is still generated by the slave. The STOP condition is generated by the master, which has previously sent a not-acknowledge (A).



A master-transmitter addressing a slave receiver with a 7-bit address.
The transfer direction is not changed.

Figure 10



A master reads a slave immediately after the first byte.

Figure 11

- Combined format: During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/W bit reversed. If a master receiver sends a repeated START condition, it has previously sent a not-acknowledge.

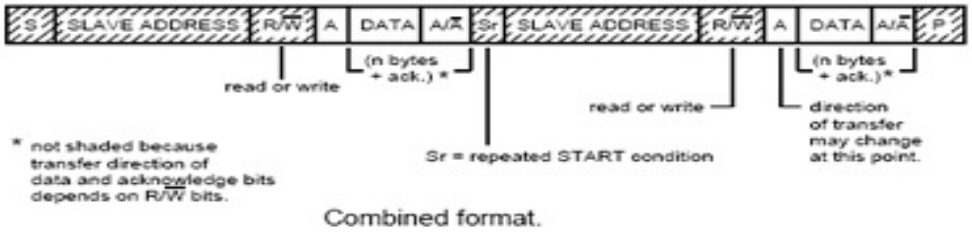


Figure 12

ADVANTAGES

- Simple
- Well Known
- Universally accepted
- Plug Play
- Cost Effective
- Large Portfolio

DISADVANTAGES

- Low Speed
- Complex Design
- Single Master Single Slave at a time

APPLICATIONS

- For adding UART / IrDA Functionality
- Cordless telephone
- Digital media adapters
- LCD display
- MP3 Players

ARCHITECTURE (DESIGN) OF I2C CONTROLLER

The I2C Controller design contains a microcontroller (mC) interface and provides I2C Master/Slave capability. It is intended to be used with a microcontroller (mC) or microprocessor (mP).

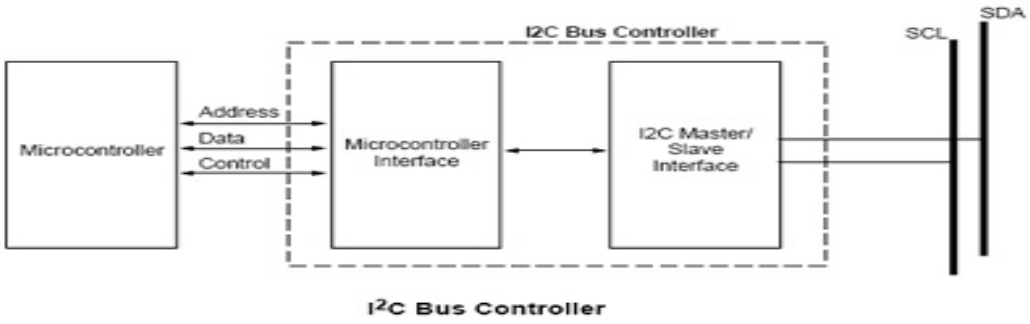
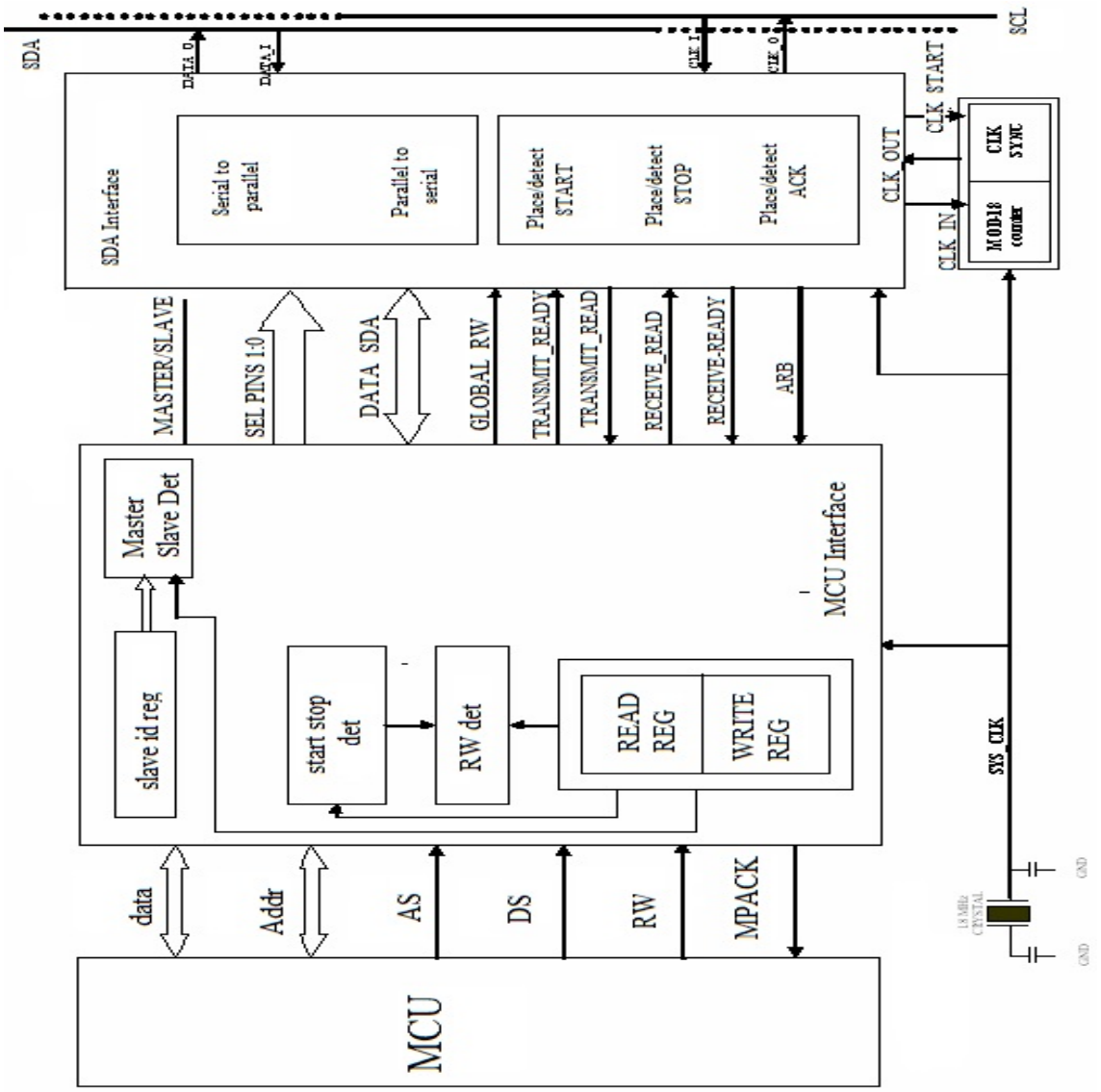


Figure 15

Name	Direction	Description
RESET	Input	**Reset pin**
SYS_CLK	Input	**System Clock**
DATA[7:0]	Bidirectional	**µC Data Bus.**
ADDR[7:0]	Bidirectional	**µC Address Bus.**
AS	Input	**Address Strobe**
DS	Input	**Data Strobe.**
RW	Input	**Read/Write**
MPACK	Output	**Microprocessor/Controller Acknowledge.**
DATA_I	Input	**I²C Serial Data Input**
CLK_I	Input	**I²C Serial Clock Input**
DATA_O	Output	**I²C Serial Data Output**
CLK_O	Output	**I²C Serial Clock Output**
PROCESSOR_WAIT	Input	**Microprocessor/Controller wait pin**

I2C CONTROLLER SIGNAL DESCRIPTION



I2C CONTROLLER BLOCK DIAGRAM

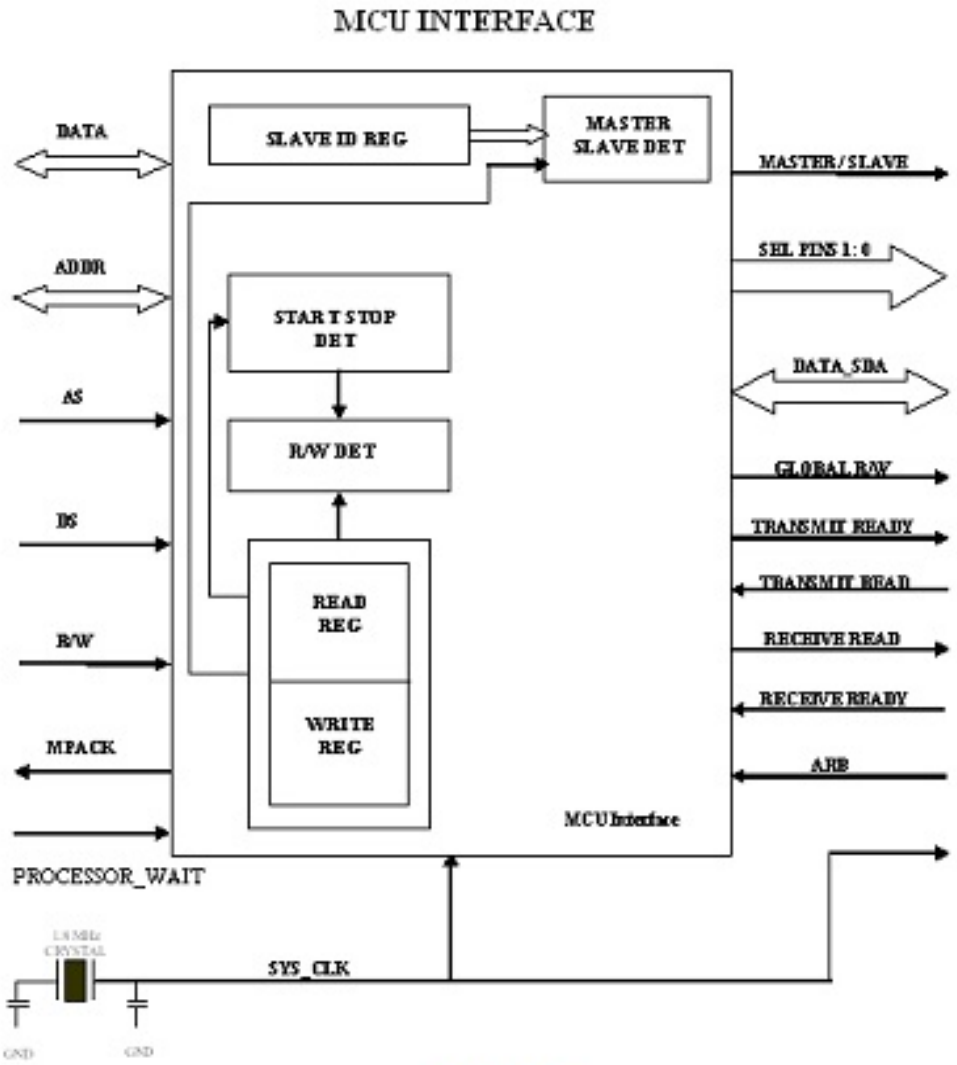


Figure 17

MICROCONTROLLER LOGIC

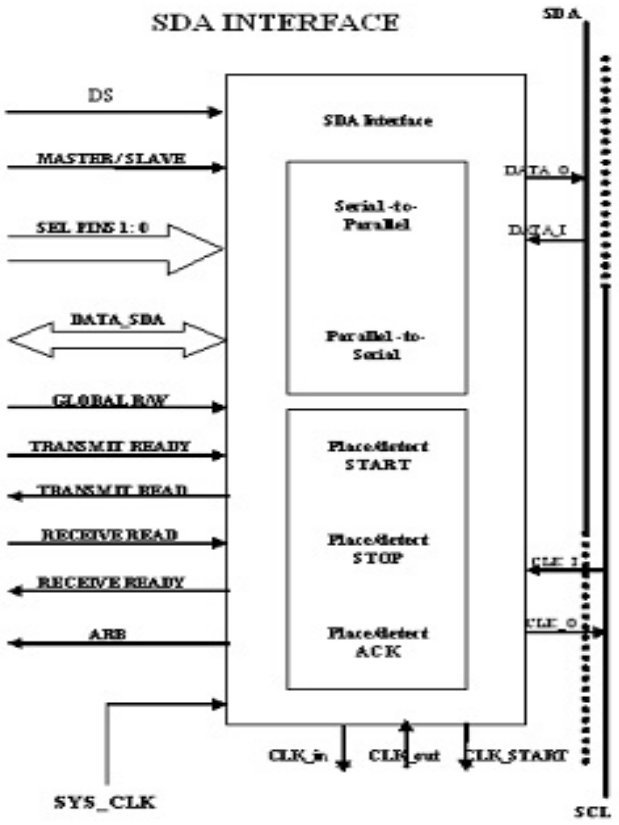


Figure 18

SDA INTERFACE LOGIC