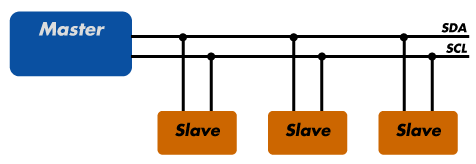
**1.Introduction**

I2C as known as inter-integrated circuit is a bus interface used to communicate between devices. The communication speed of a standard I2C can serve up to 100kbits per second. In fast mode, the speed can serve up to 400kbits per second. However as the technology became advanced nowadays, the some of the I2C interface can provide the communication speed in high speed mode which up to 3.4Mbits per second.

I2C interface used two bus line to communicate. One of the bus line is called SDL (Serial Data line) which is used to transfer data while the other one is called SCL (Serial Clock Line) is used to transfer the clock pulse to trigger the data. In the communication with I2C interface, the devices are connected with the relationship between master and slave. There can be multiple master devices and multiple slave devices connected with only one I2C interface bus line, but only one master and one slave can communicate at the same time.

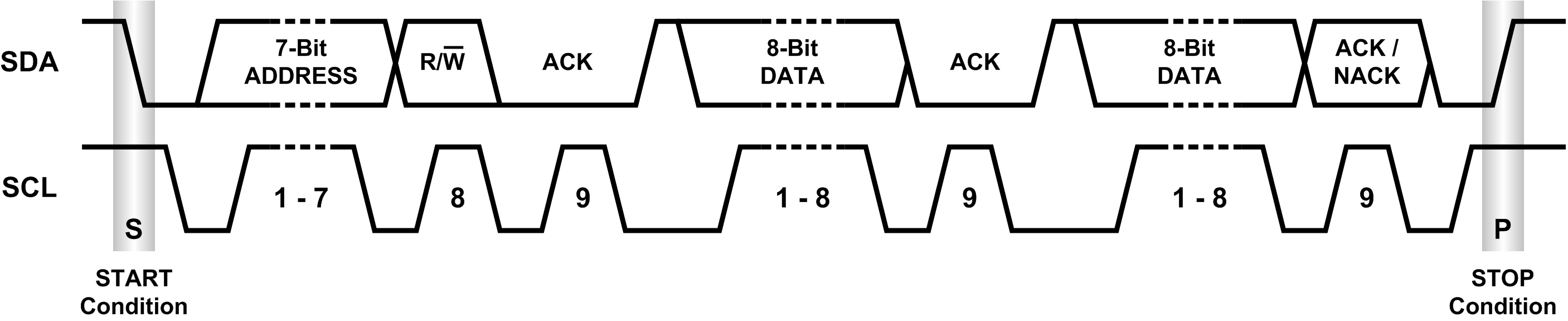
In master device, they can select one of the slave device to communicate if the device is connected in the same I2C interface bus line. Master devices can device whether send or receive data from a slave device. For a slave device, the interface are just waiting for the master to select them to get communication by matching the address sent from the master with their own address.



(Access from <http://www.totalphase.com/support/articles/200349156-I2C-Background>)

Protocol

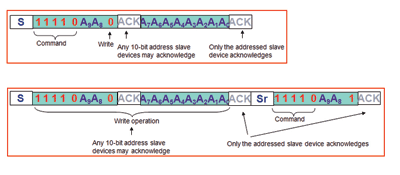
There is 7 bit address and 10 bit address mode in I2C slave device. Normally 7 bit address mode was used while the 10 bit address mode is the enhanced version of 7 bit address mode, so that the master device able to connected with more slave devices in a single connection.



(Access from <http://www.planetanalog.com/document.asp?doc_id=527900>)

In 7 bit address mode, the master will generate a start bit by set the START bit in CR1 followed by the address byte. The address byte contains the 7 bit address and the LSB bit which the whether the master is going to transmit or receive data. The acknowledge bit was set to low by the slave device, the master will follow by send data or receive data from the slave.

When the master want to stop the communication the master should generate a stop condition by set the STOP bit to 1 if the master is transmitting data. While for master receiver mode, the master should give a NACK signal after receive the last data in order to send a stop condition to the slave.



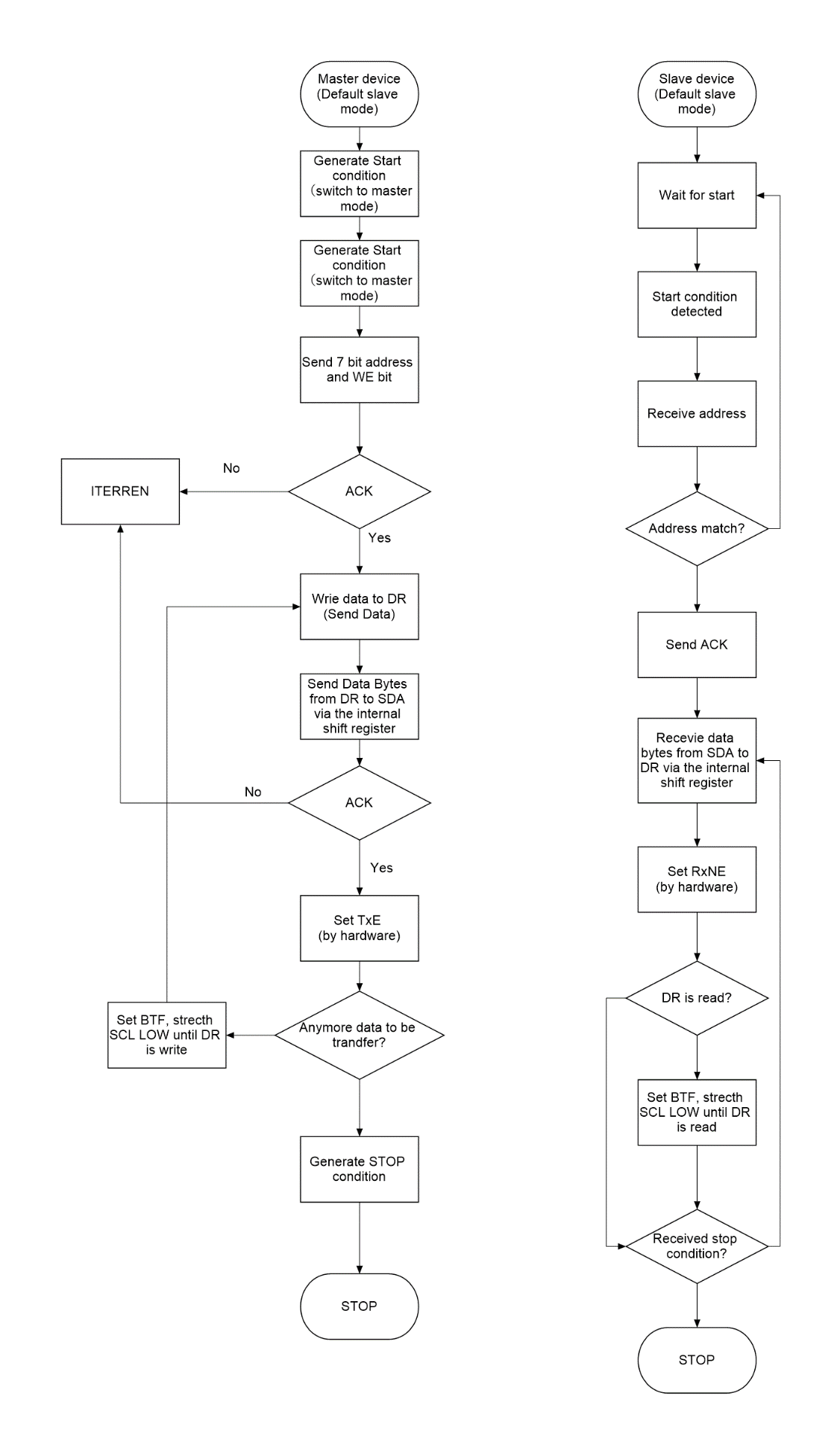
S is the start condition, while Sr is the restart condition

(Access from <http://blog.testequipmentconnection.com/using-an-oscilloscope-to-debug-the-i2c-protocol> )

In 10 bit mode, a header byte should be sent after the start bit was generated. The header bytes contains the header bits “11110XX0”, XX is the most two significant bits of the 10 bit address. Secondly, send the remaining 8 bit of the address. If the master decided to receive data from a slave device, the master required to repeat a start condition then send header bits “11110XX1” in order to change to receiver mode from transmitter mode.

**2. Methodology**

Flow chart



This is the code design flow for master transmit and slave receive mode. When the master does not receive acknowledge from the slave, the program counter will jump to interrupt (If ITERREN = 1).

No

Yes

Yes

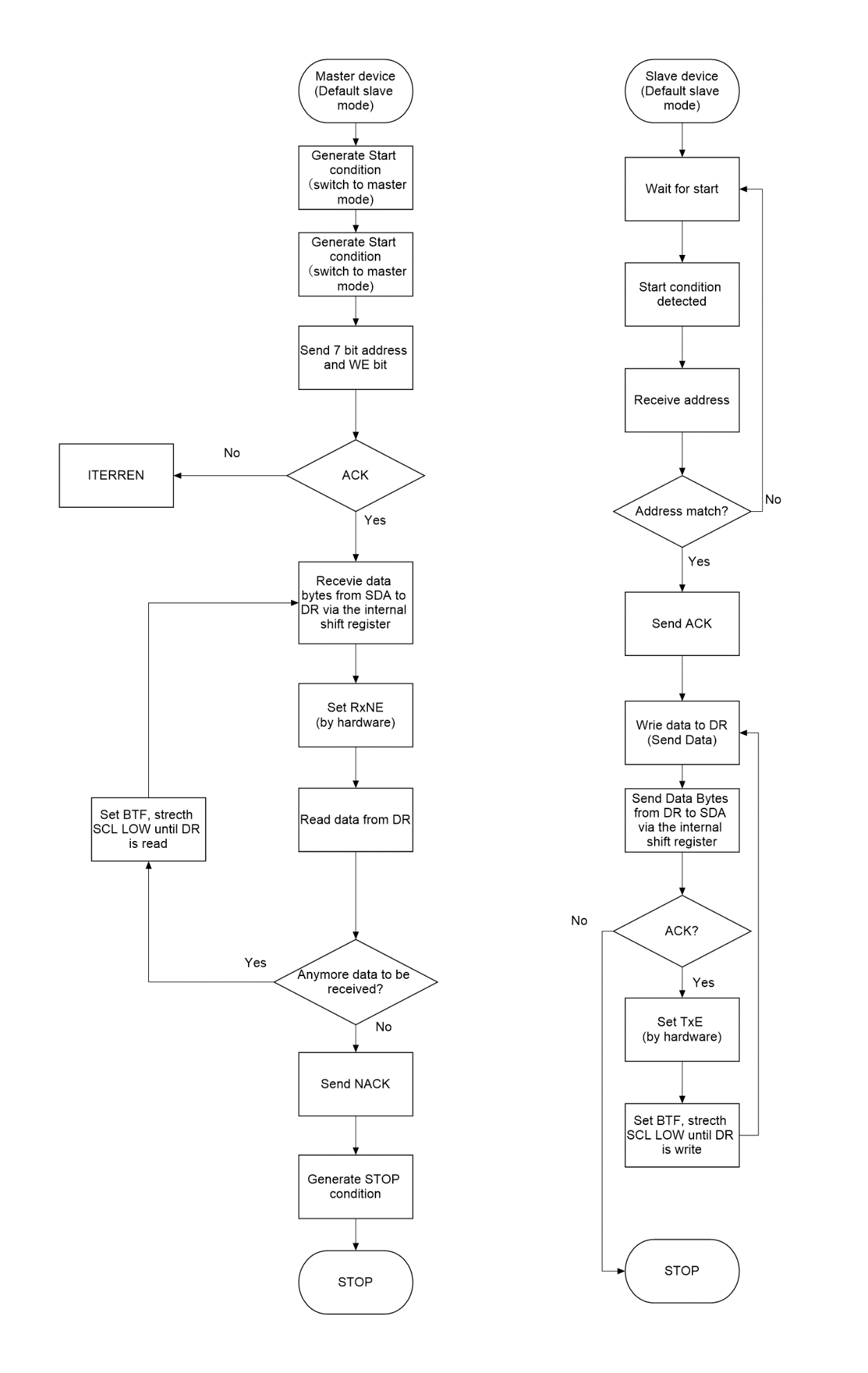
No

No

No

Yes

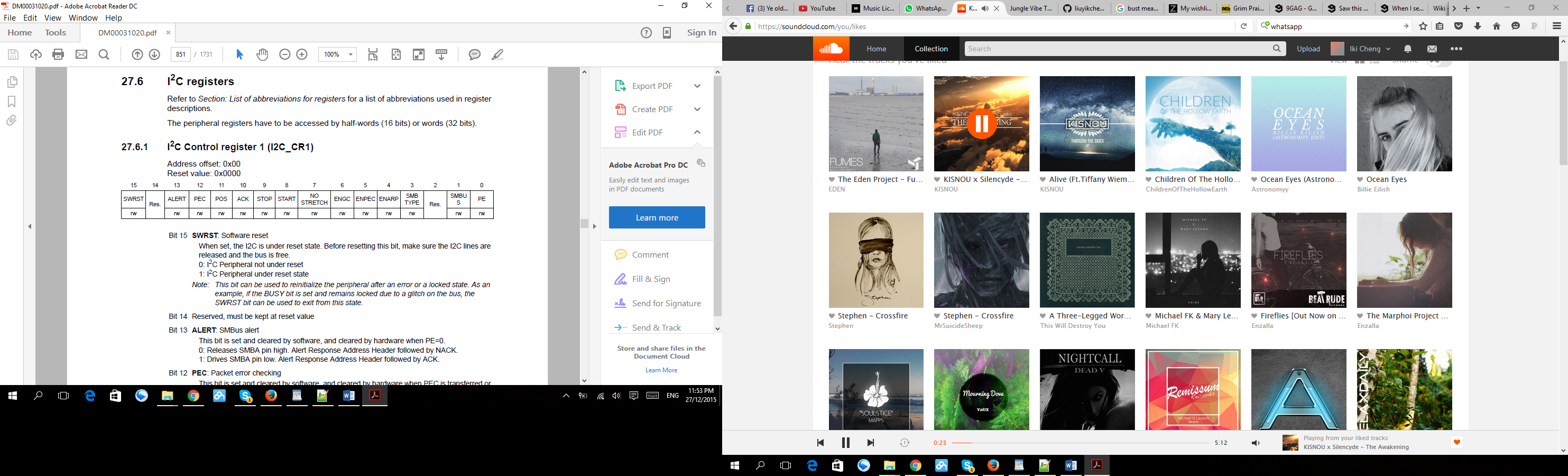
Yes



This is the code design flow chart for master receive and slave transmit mode. When the master want to stop, it should send acknowledge in order to let the slave stop the slave from transmitting data to the master and send a STOP condition to release the connection. However, if the slave receive any NACK, the slave will stop sending data.

**Configuration for I2C**

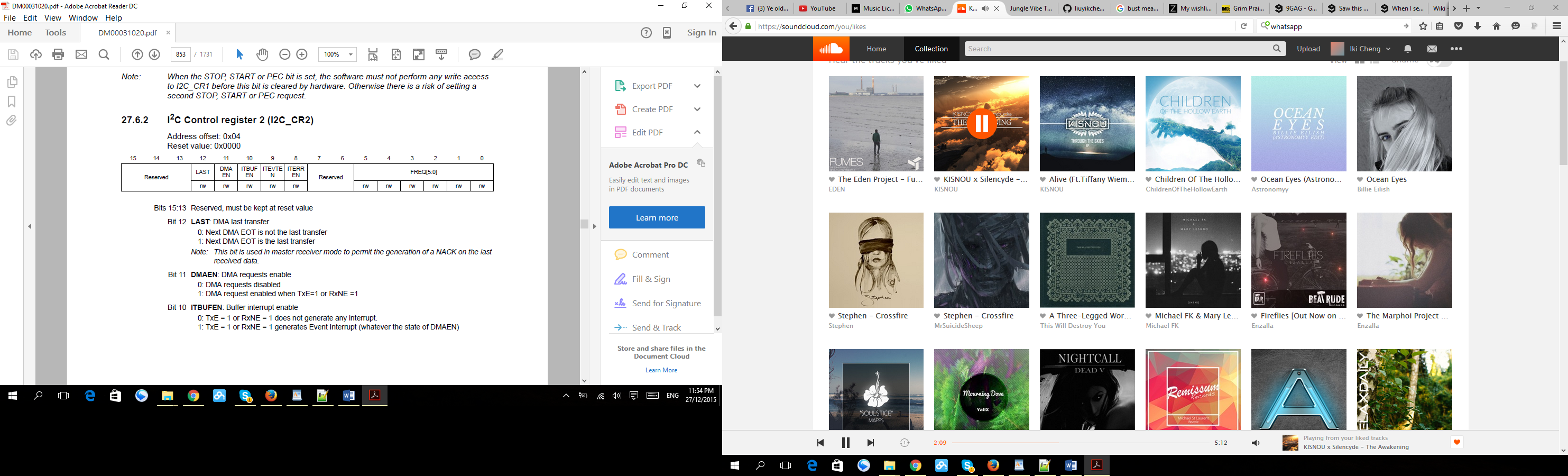
I2C Control Register 1



ACK was set to enable acknowledgement.

PE was set to enable the peripheral. (This bit must be enable at last after configure the other register.)

I2C Control Register 2

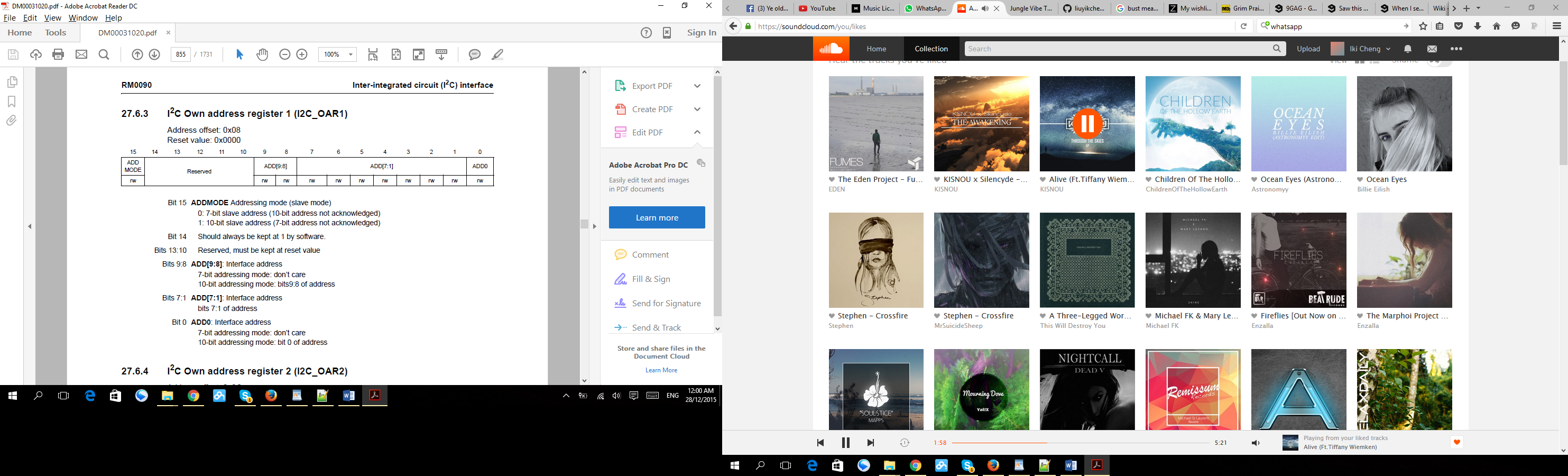


ITBUFEN, ITEVEN and ITERREN were set to enable all interrupts.

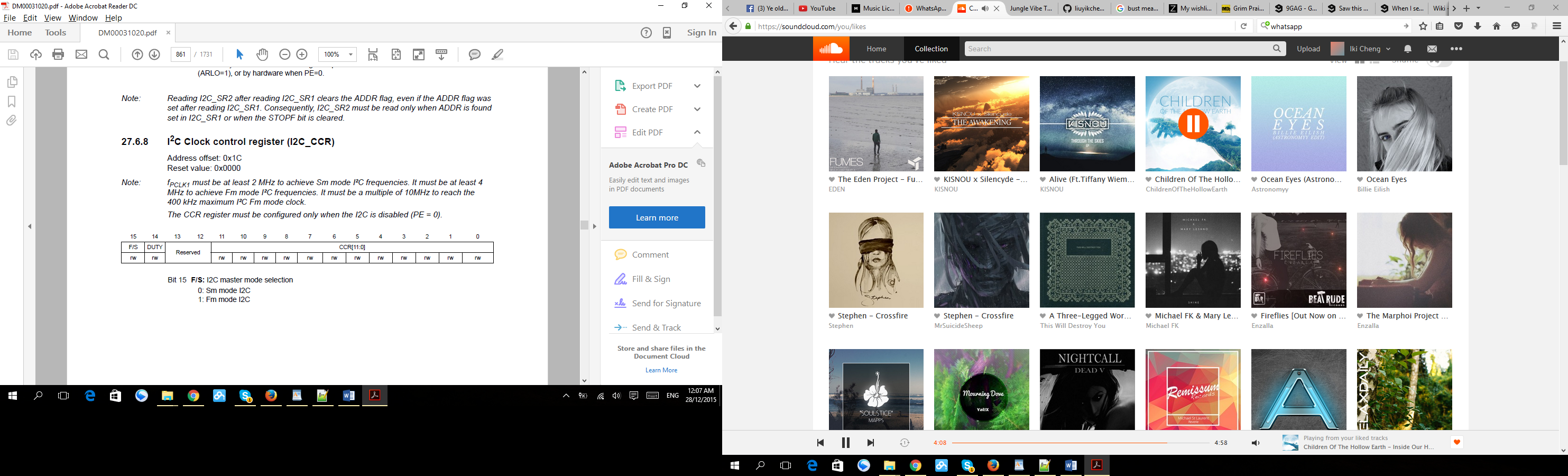
Set FREQ to 4MHz as this experiment was run in standard mode.

I2C Own Address Register 1

This register need to be configured for slave devices only.



ADD[7:1] indicated the slave address was assigned with a fixed value. Only ADD[7:1] was assign for 7 bit address mode.



CRR[11:0] was set with value 0x100 to generate about 8k of SCL frequency.

**Configuration for GPIO**

PB6 and PB7 were configured as SCL and SDA of I2C1.

PA8 and PC9 were configured as SCL and SDA of I2C3.

Set all the port as alternate function mode, open drain, no pull up pull down and high speed mode.

Set the alternative function register (AFRH or AFRL) according to the alternate function mapping table for STM32F429xx device which may found in <http://www.sciencezero.org/index.php?title=STM32F429_Microcontroller>.

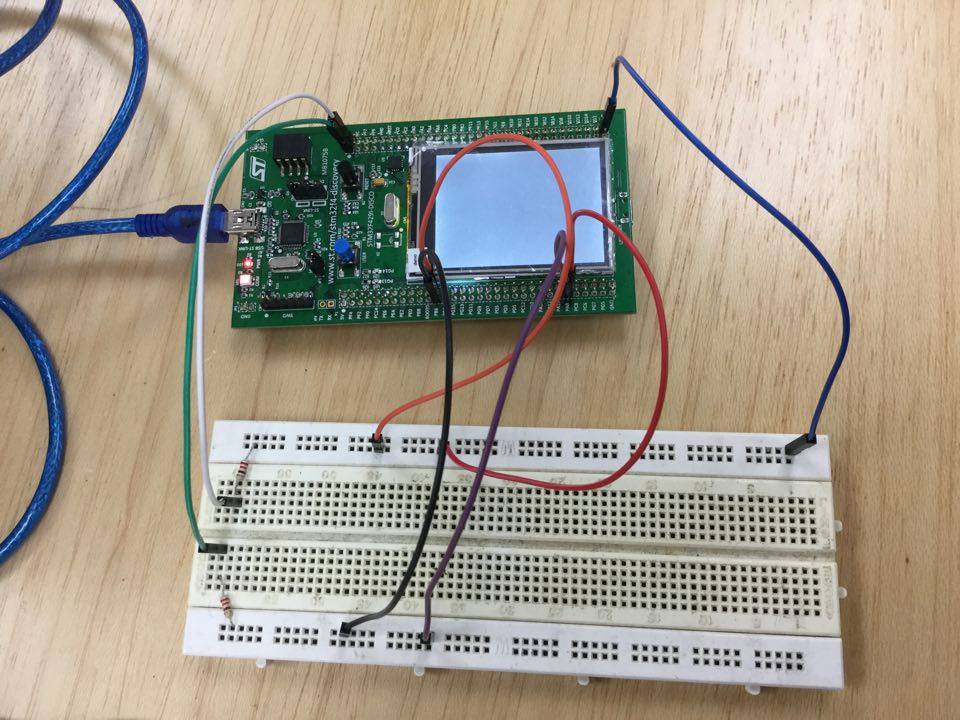
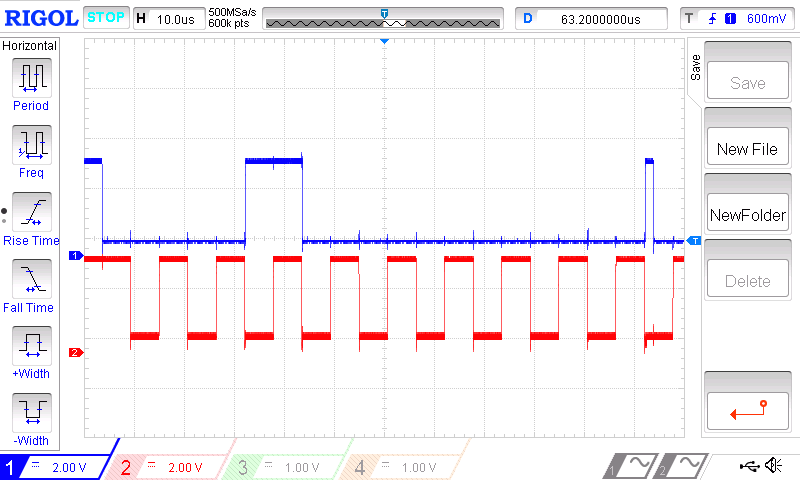
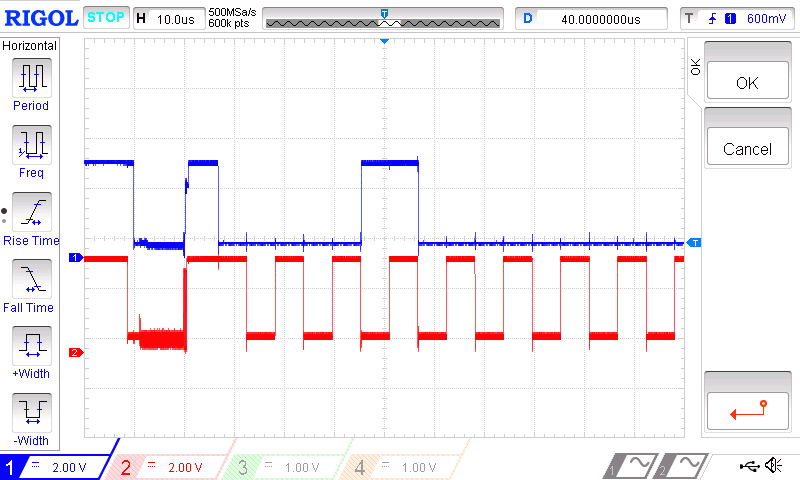


Figure: This figure shown the way of the connection of the external pull-up voltage and the two I2C interfaces which were the I2C1 and I2C3.

**3.Result**

I2C3 was configured as the master device will the I2C1 was configured as slave device.

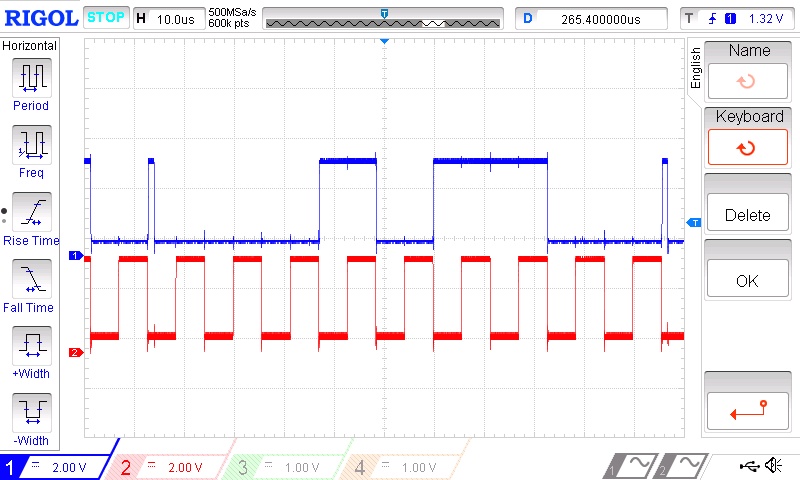
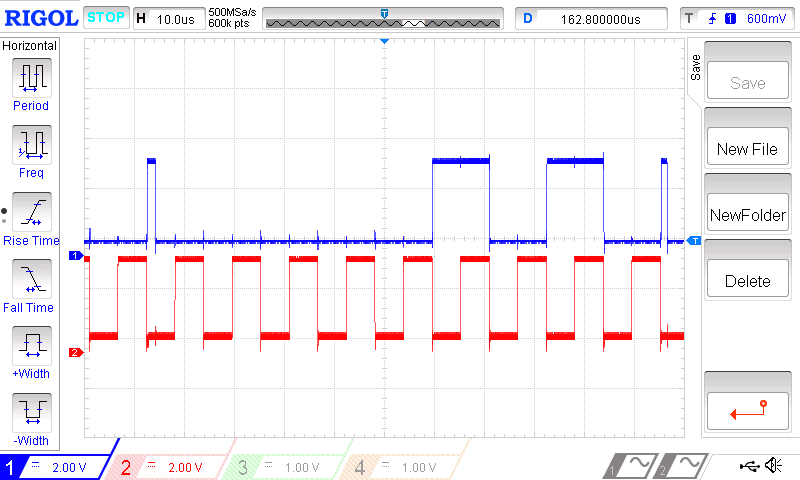
Send 0x5 and 0x16 from master to slave with address 7 bit 0x20



ACK

Slave address 0x20

Start condition

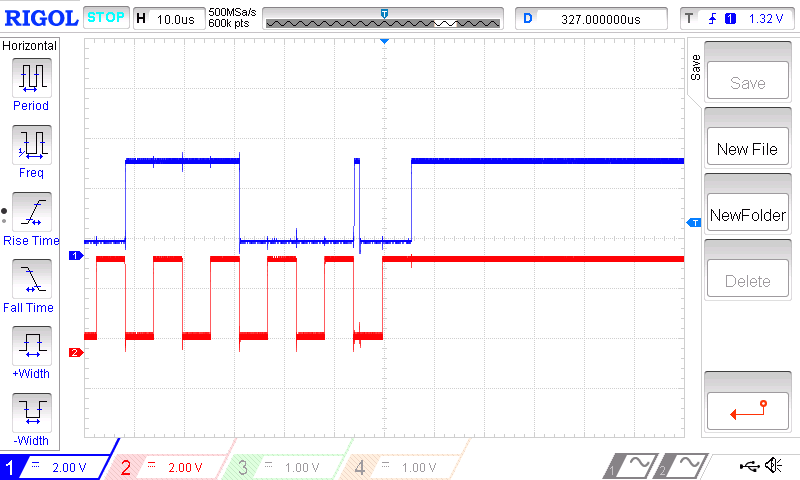


ACK

0x5

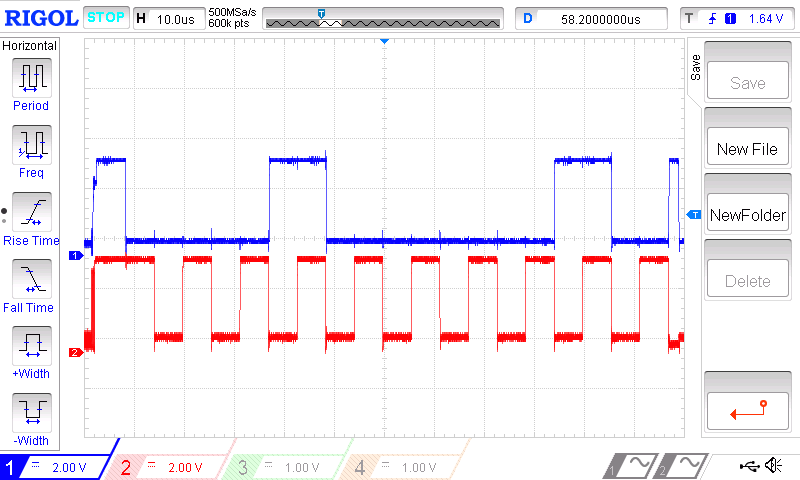
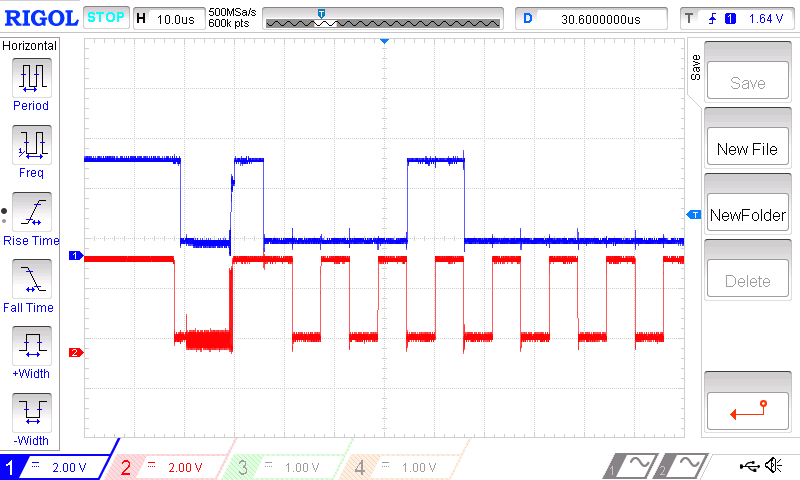
0x16

ACK



Stop condition

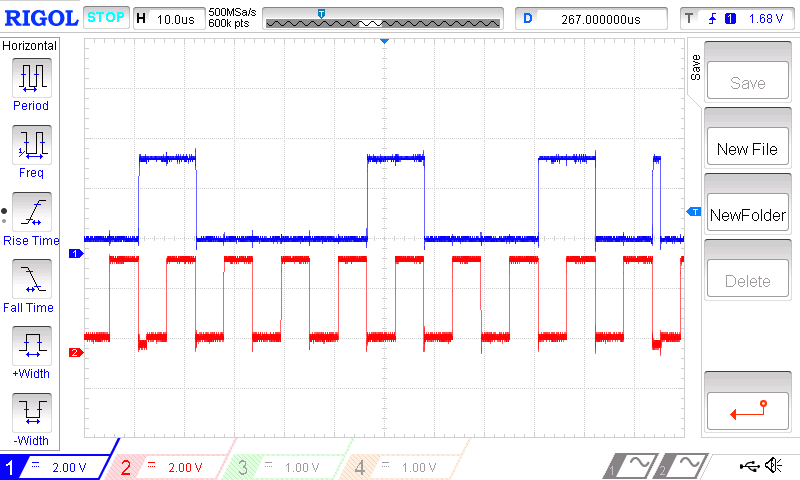
Receive 0x89, and 0x56 from slave with 7 bit address 0x21 to master



ACK

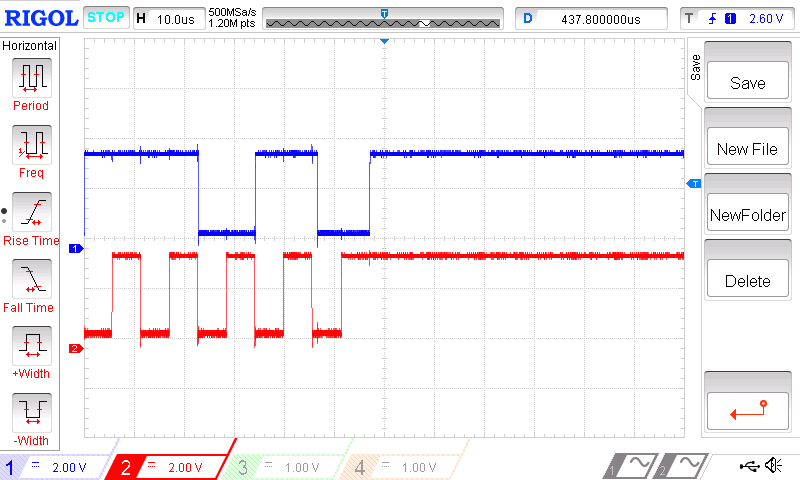
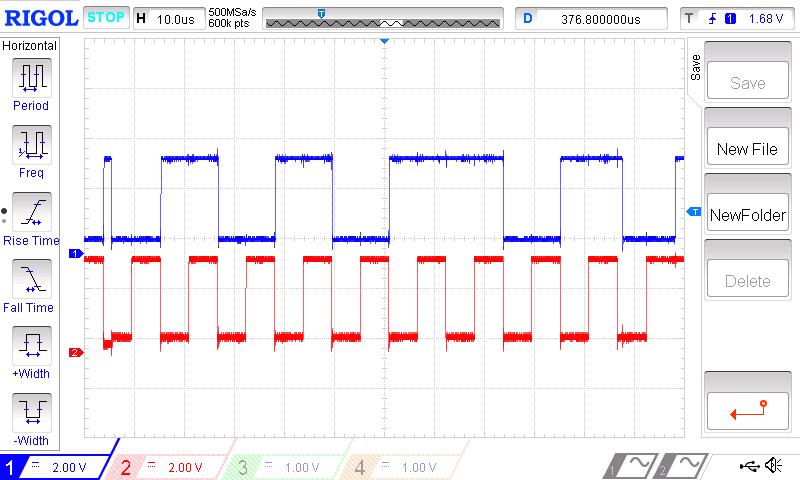
Slave address 0x21

Start condition



ACK

0x89

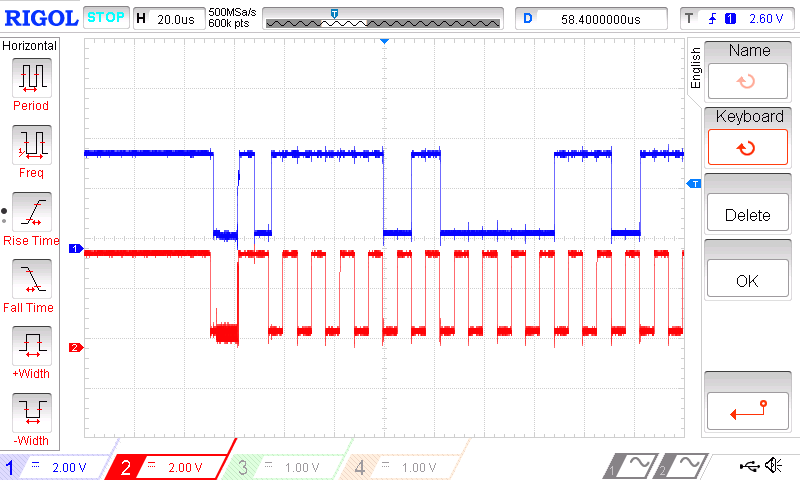


NACK

0x56

Stop condition

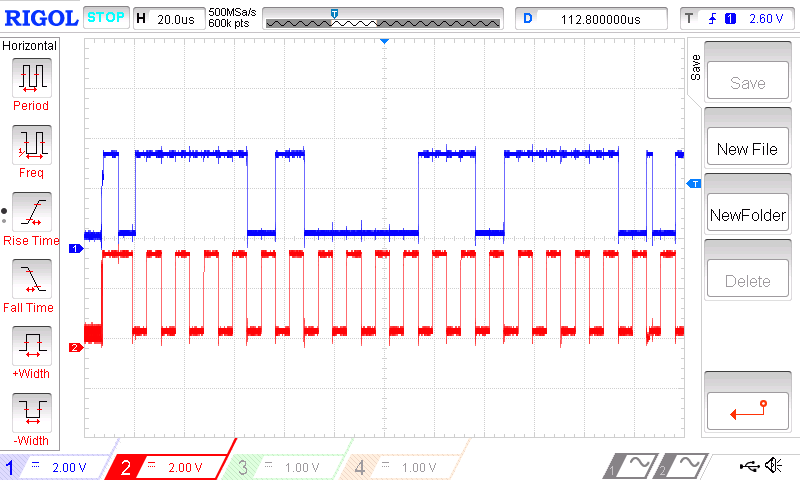
Send 0x62 and 0x21 from master device to slave device with 10 bit address mode.



ACK

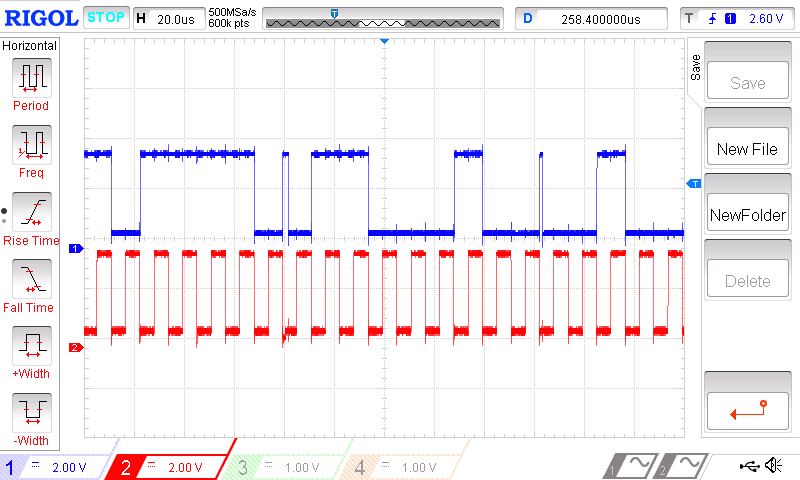
Header Byte

Start Condition



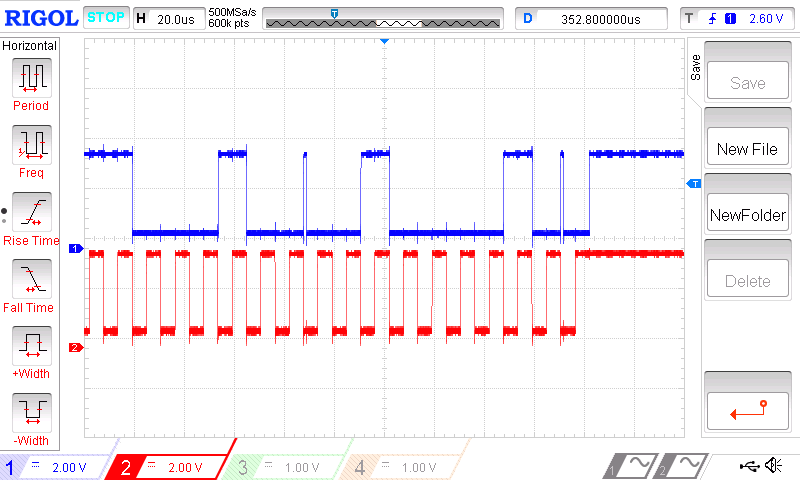
ACK

Remaining address byte



ACK

0x62

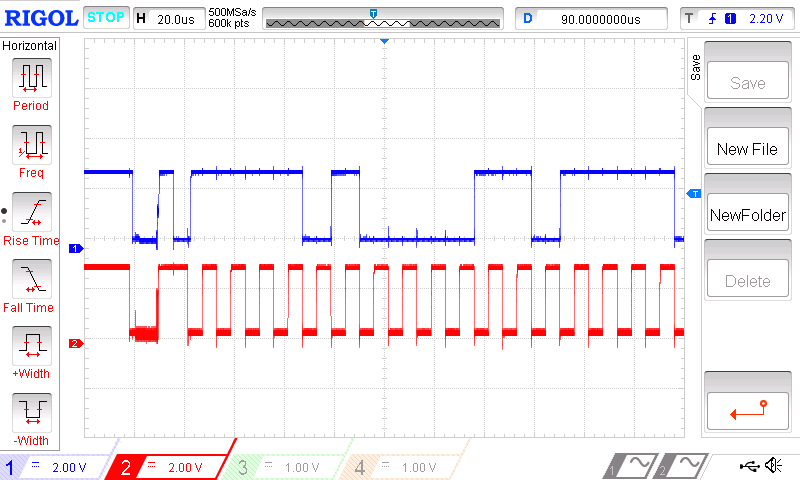


ACK

Stop Condition

0x21

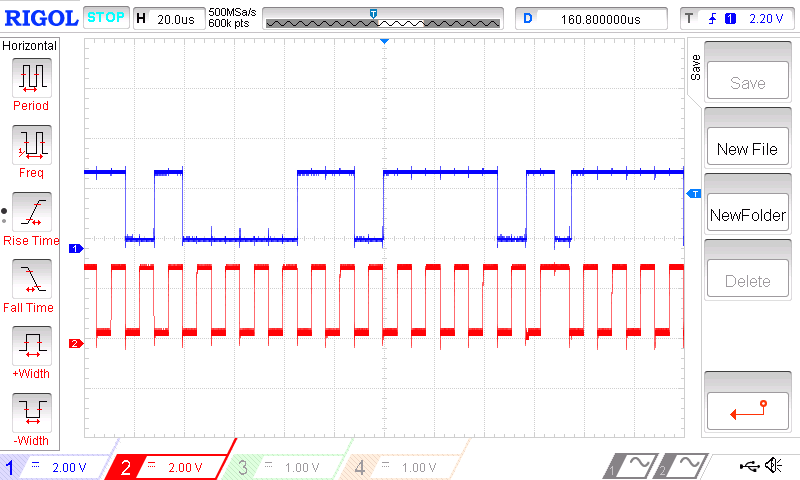
Receive 0x46 and 0x18 from slave device to master device in 10 bit address mode.



ACK

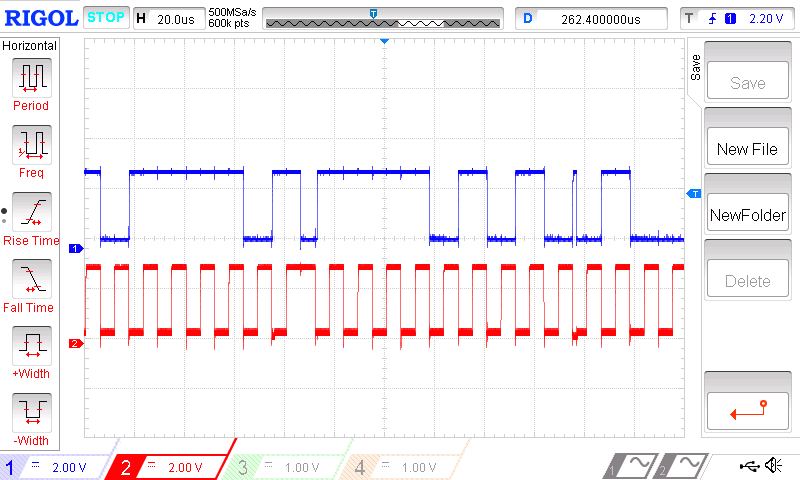
Start Condition

Header byte



ACK

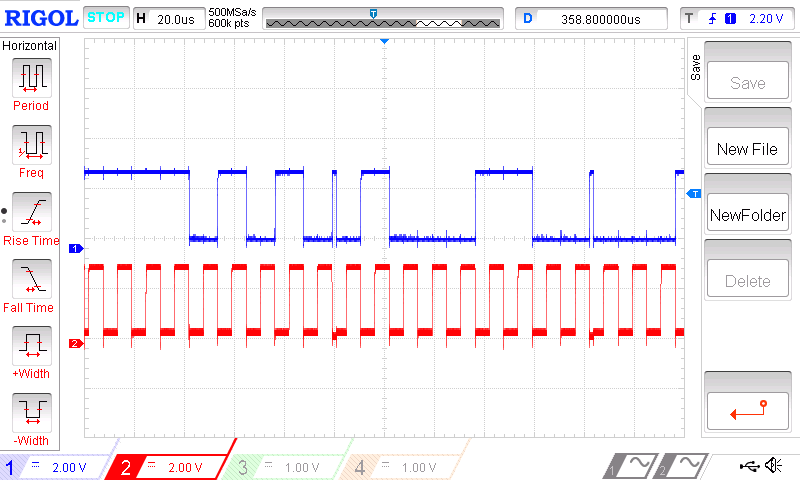
Slave address



ACK

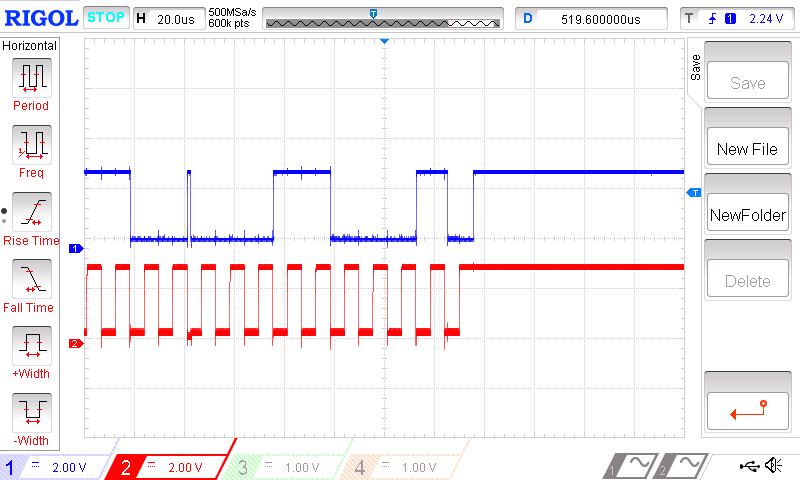
Restart Condition

Header byte with write bit = 1



ACK

0x46

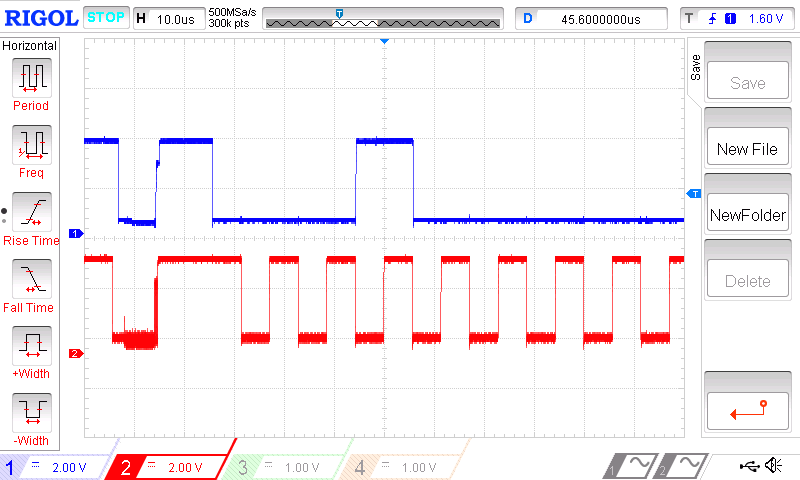


NACK

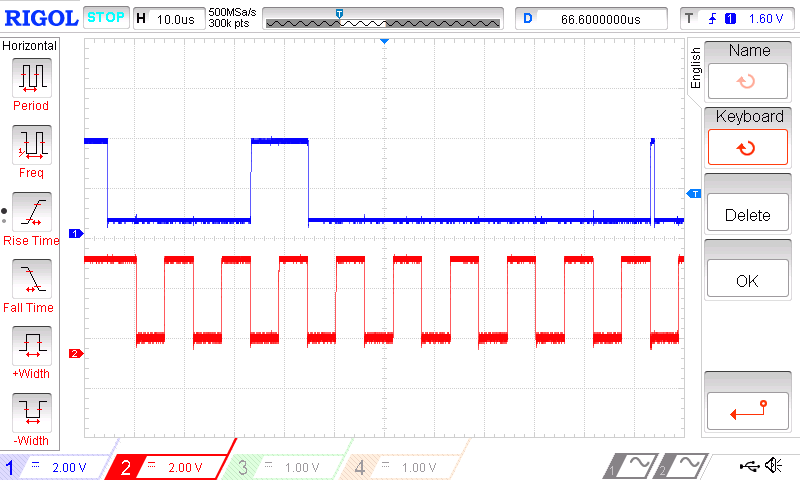
Stop Condition

0x18

Transmission using DMA

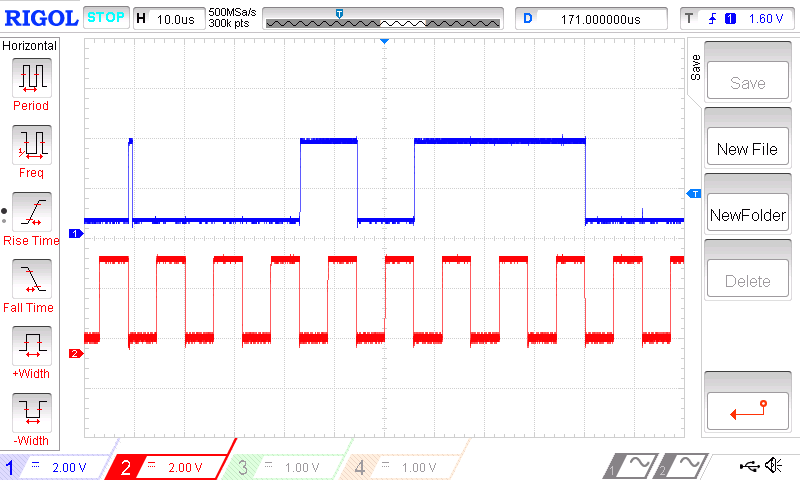


Start Conditon



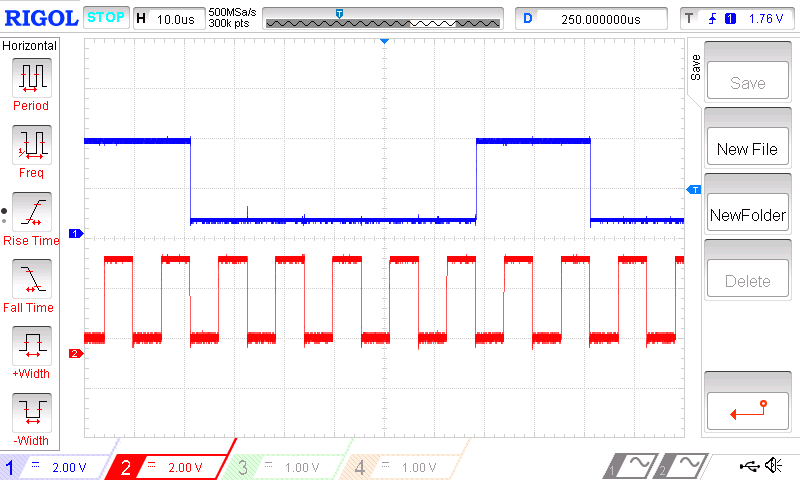
ACK

Slave address byte



ACK

0x17



0xC



ACK

Stop condition

**4.Discussion**

From the observation of the result, once the start condition was generated, the first byte data which was the address of the slave will be sent. Every data byte had an acknowledge bit at the end of the byte. The acknowledge bit was controlled by the receiver. When the receiver was successfully receive the data, it pull the acknowledge bit to low, else the bit will remain high. When the master device was receiving, the master device purposely generate a non-acknowledge(NACK) in order to get back the control of the line to send the stop condition. The reason of a master device have to generate the stop condition to close the communication instead of just leave it there after finishing sent or received data is because so that the master device are able to switch back to the slave device and allow the other master device to communicate with it.

In the waveform some of the glitch like wave was observed as shown in the Figure 4.1.

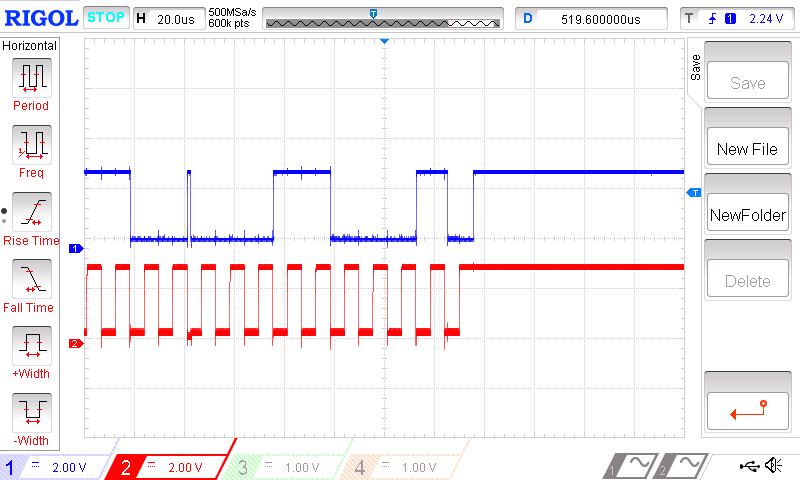


Figure 4.1

This glitch was caused by the clock stretching from the slave device. It only happen between transmission of two bytes data. For example, during data transmission from master device to slave device, after the slave device received the first byte of data, the slave device actually need time to read the data from the Data Register, it consumed time. So, the slave device will pull the SCL to low to hold the data from transmitting to prevent overrun error occur.

From the GPIO configuration, no pull up and pull down was selected as external pull up voltage was used.

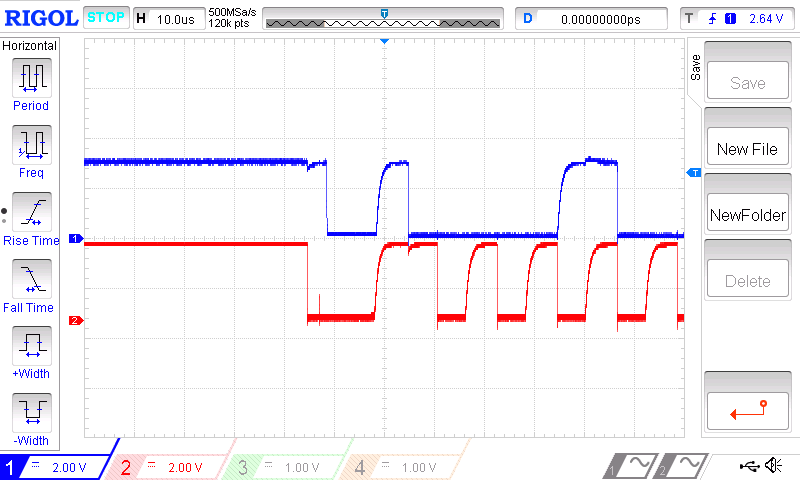
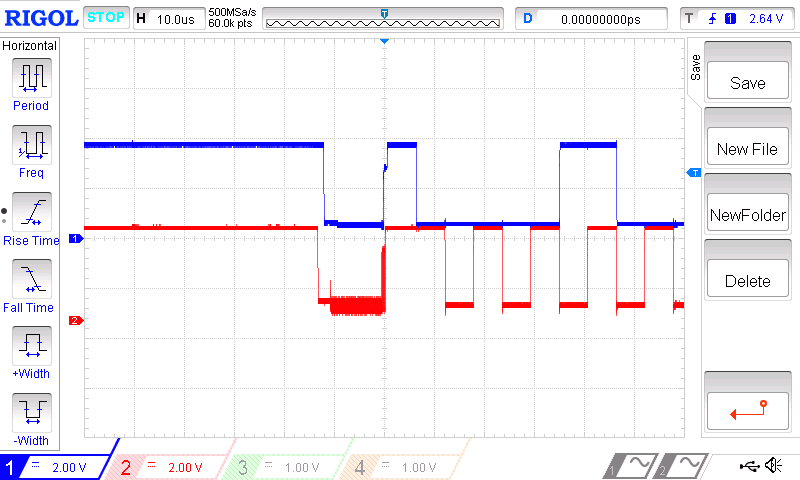


Figure 4.2 Figure 4.3

Figure 4.2 shown the signals with the external pull up voltage source while Figure 4.3 shown the signals with internal pull up source. The reason may cause the shark like wave shown in Figure 4.3 was the internal pull up source not fast enough to charge up the voltage. As the maximum allowed injected current for the I/O pin was only 25mA which given in the datasheet ( <http://www.st.com/web/en/resource/technical/document/datasheet/DM00071990.pdf> ), while the external pull up voltage was 3V, by using Ohm’s law calculation R = V/I, 120 Ohms was the minimum resistor used for the experiment. However 200 Ohm resistor was used to reduce the power source.

In the transmission using DMA, the data need to be send to the slave device were store in a transmit buffer memory. In this experiment, the data was transmitted were 0x17 and 0xC. After generated the start condition, the slave address were send manually by setting the Data Register of the master device. So that the TxE flag was set after sent the address of the slave device. When TxE flag was set, the data in the buffer memory will automatically sent the according data 0x17 to the Data Register of the master device and transmit to the slave device. When the TxE flag was set, the second data 0xC was transmitted.