**What is I2S Protocol?**

The protocol which is used to transmit digital audio data from one device to another device is known as I2S or Inter-IC Sound protocol. This protocol transmits PCM (pulse-code modulated) audio data from one IC to another within an electronic device. I2S plays a key role in transmitting audio files which are pre-recorded from an MCU to a DAC or amplifier. This protocol can also be utilized to digitize audio using a microphone. There is no compression within I2S protocols, so you cannot play OGG or MP3 or other audio formats that condense the audio, however, you can play WAV files.

**Features**

The **I2S protocol features** include the following.

* It has 8 to 32 data bits for each sample.
* Tx & Rx FIFO interrupts.
* It supports DMA.
* 16-bit, 32-bit, 48-bit, or 64-bit word select period.
* Simultaneous bi-directional audio streaming.
* 8-bit, 16-bit, and 24-bit sample width.
* It has different sample rates.
* The data rate is up to 96 kHz through the 64-bit word select period.
* Interleaved stereo FIFOs or Independent right & left channel FIFOs
* Independent enable of Tx & Rx.

**I2S Communication Protocol Working**

The I2S [communication protocol](https://www.elprocus.com/communication-protocols/)  is a 3 Wire protocol that simply handles audio data through a 3-line serial bus which includes SCK (Continuous Serial Clock), WS (Word Select) & SD (Serial Data).

**3-Wire Connection of I2S:**

**SCK**

The SCK or Serial Clock is the first line of the I2S protocol which is also known as BCLK or bit clock line which is used to obtain the data on a similar cycle. The serial clock frequency is simply defined by using the formula like Frequency = Sample Rate x Bits for each channel x no. of channels.

**WS**

In the I2S communication protocol, the WS or word select is the line which is also known as FS (Frame Select) wire that separates the right or left channel.

If WS = 0 then left channel or channel-1 is used.

If WS = 1 then the right channel or channel-2 is used.

**SD**

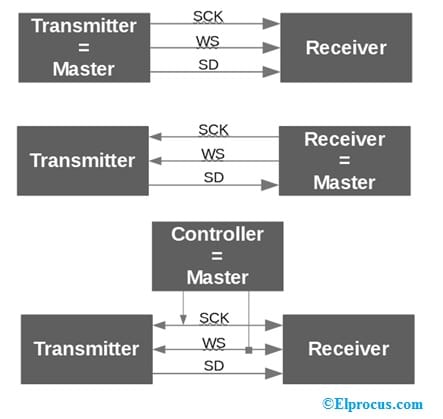
The Serial Data or SD is the last wire where the payload is transmitted within 2 complements. So, it is very significant that the MSB is first transferred, because both the transmitter & receiver may include different word lengths. Thus, the transmitter or the receiver has to recognize how many bits are transmitted.

* If the word length of the receiver is greater than the transmitter, then the word is shortened (LSB bits are set to zero).
* If the word length of the receiver is less than the word length of the transmitter, then the LSB bits are ignored.

The **transmitter** can sent the data either on the **leading edge or trailing edge of the clock pulse**. This can be configured in the corresponding **control registers**. But the **receiver latches the serial data and WS only on the leading edge of the clock pulse**. The transmitter transmits data only after one clock pulse after change in WS.  The receiver uses the WS signal for synchronisation of the serial data.

**I2S Network Components**

When multiple I2S components are connected to each other then this is called an I2S network. The component of this network includes different names & also different functions. So, the following diagram shows 3 different networks. Here an ESP NodeMCU board is used as the transmitter and an I2S audio breakout board is used as the receiver. The three wires used to connect the transmitter and receiver are SCK, WS & SD.

I2S Network Components

In the first diagram, the transmitter (Tx) is the master so it controls the SCK (serial clock) & WS (word select) lines.

In the second diagram, the receiver is the master. So both the SCK & WS lines start from the receiver & the transmitter ends.

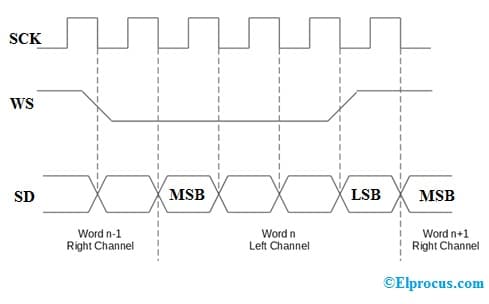
In the third diagram, an exterior controller is connected to the nodes within the network which works like the master device. So this device generates the SCK & WS.

In the above-all I2S networks, there is only a single master device available and many other components that transmit or receive sound data.

In I2S any device can be the master by providing the clock signal.

**I2S Timing Diagram**

For a better understanding of the I2S & its functionality, we have the I2S communication protocol timing diagram shown below. The timing diagram of the I2S protocol is shown below which includes three wires SCK, WS & SD.

**I2S Protocol Timing Diagram**

In the above diagram, first, the serial clock  has a Frequency = Sample Rate \* Bits for each channel \* no. of channels). The word select line is the second line that changes in between ‘1’ for the right channel & ‘0’ for the left channel.

The third line is the serial data line where the data is transmitted on each clock cycle on the falling edge denoted with dots from HIGH to LOW.

In addition, we can notice that the WS line varies one CLK cycle before the MSB is transmitted which gives the receiver time for storing the earlier word & clear the input register for the next word. The MSB is sent when SCK changes after WS changes.

Whenever a data is transmitted between the transmitter and receiver there would a propagation delay  which would be

propagation delay = (time difference between the external clock and the internal clock of the receiver )+( time difference between the internal clock to when data is received).

To minimise the propagation delay and for synchronisation of the data transmission between the transmitter and the receiver  it is required that the transmitter has a clock period  of

**T >tr  –**To assume that T is the clock period of the transmitter and tr is the minimum clock period of the transmitter.

Under above condition if we consider for example a **transmitter  with the data transmission rate 2.5MHz then:**

tr = 360ns

clock High tHC (minimum) >0.35 T.

clock Low tLC (minimum> > 0.35T.

**Receiver as a slave with the data transmission rate 2.5MHz then:**

clock High tHC (minimum) < 0.35 T

clock Low tLC (minimum) < 0.35T.

setup time tst(minimum) < 0.20T.

**I2S Protocol Arduino**

The main objective of this project is to make an I2S  theremin interface using the Arduino I2S library. The required components to make this project are; Arduino MKR Zero, [Breadboard](https://www.elprocus.com/a-brief-on-breadboard-basics-and-connections/), Jumper wires, Adafruit MAX98357A, 3W, 4 ohms speaker, and RobotGeek Slider.

The Arduino I2S library simply allows you to transmit & receive digital audio data over the I2S bus. So this example targets to explain how to utilize this library to drive an I2S DAC for reproducing sound computed in the Arduino design.

This circuit can be connected as; The I2S DAC utilized in this example requires simply three wires as well as a power supply for the I2S bus. The connections for the I2S on the Arduino MKRZero follow as;

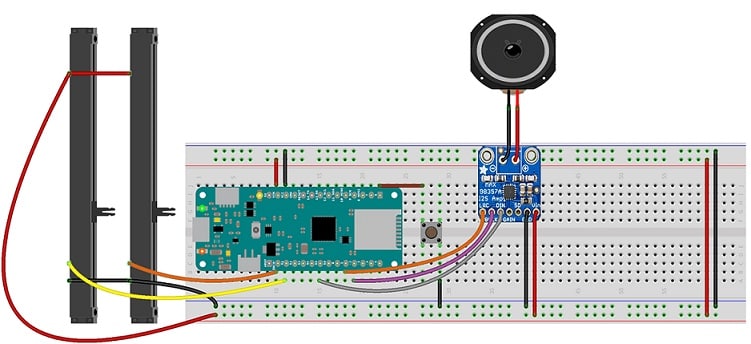
Serial Data (SD) on pin A6;

Serial Clock (SCK) on pin2;

The frame or Word Select (FS) on pin3;

**Working**

Basically, the theremin has two controls pitch and volume. So, these two parameters are modified by moving two slide potentiometers, however, you can also adjust them to read them. The two potentiometers are connected in a voltage divider form, so moving these potentiometers you will get values from 0 to 1023. After that, these values are mapped in between the maximum & minimum frequency and the least and highest volume.

**I2S Thermin Diagram**

The sound transmitted on the I2S bus is a simple sine wave whose amplitude & frequency is modified based on the reading of the potentiometers.

**Code**

The code to interface a Theremin with an Arduino MKRZero, 2-slider potentiometers & an I2S DAC is given below.

**#include <I2S.h>**

**const int maxFrequency = 5000; //maximum generated frequency**  
**const int minFrequency = 220; //minimum generated frequency**  
**const int maxVolume = 100; //max volume of the generated frequency**  
**const int minVolume = 0; //min volume of the generated frequency**  
**const int sampleRate = 44100; //samplerate of the generated frequency**  
**const int wavSize = 256; //buffer size**  
**short sine[wavSize]; //buffer in which the sine values are stored**  
**const int frequencyPin = A0; //pin connected to the pot which determines the frequency of the signal**  
**const int amplitudePin = A1; //pin connected to the pot which determines the amplitude of the signal**  
**const int button = 6; //pin connected to the button control  to display the frequency**

void setup()  
{

**Serial.begin(9600); //configue the serial port**  
**// Initialize the I2S transmitter.**  
**if (!I2S.begin(I2S\_PHILIPS\_MODE, sampleRate, 16)) {**  
**Serial.println(“Failed to initialize I2S!”);**

**while (1);**  
}

generateSine(); // fill buffer with sine values  
pinMode(button, INPUT\_PULLUP); //put the button pin in input pullup

}  
**void loop() {**

**if (digitalRead(button) == LOW)**

{

**float frequency = map(analogRead(frequencyPin), 0, 1023, minFrequency, maxFrequency); //map frequency**  
**int amplitude = map(analogRead(amplitudePin), 0, 1023, minVolume, maxVolume); //map amplitude**  
**playWave(frequency, 0.1, amplitude); //play sound**  
//print values on serial  
**Serial.print(“Frequency = “);**  
**Serial.println(frequency);**  
**Serial.print(“Amplitude = “);**  
**Serial.println(amplitude);**

}

}  
**void generateSine() {**  
**for (int i = 0; i < wavSize; ++i) {**  
**sine[i] = ushort(float(100) \* sin(2.0 \* PI \* (1.0 / wavSize) \* i)); //100 is used to not have small numbers**  
**}**  
}  
void playWave(float frequency, float seconds, int amplitude) {  
// Play back the provided waveform buffer for the specified  
// amount of seconds.  
// First calculate how many samples need to play back to run  
// for the desired amount of seconds.

unsigned int iterations = seconds \* sampleRate;

// Then calculate the ‘speed’ at which we move through the wave  
// buffer based on the frequency of the tone being played.

**float delta = (frequency \* wavSize) / float(sampleRate);**

// Now loop through all the samples and play them, calculating the  
// position within the wave buffer for each moment in time.

**for (unsigned int i = 0; i < iterations; ++i) {**  
**short pos = (unsigned int)(i \* delta) % wavSize;**  
**short sample = amplitude \* sine[pos];**

// Duplicate the sample so it’s sent to both the left and right channels.  
// It appears the order is the right channel, left channel if you want to write  
// stereo sound.

**while (I2S.availableForWrite() < 2);**  
**I2S.write(sample);**  
**I2S.write(sample);**

}  
}

**Difference between I2C and I2S Protocol**

The difference between I2C and I2S Protocol includes the following.

|  |  |
| --- | --- |
| **I2C** | **I2S** |
| The [I2C protocol](https://www.elprocus.com/i2c-bus-protocol-tutorial-interface-applications/) stands for inter-IC bus protocol | The I2S stands for Inter-IC Sound protocol**.** |
| It is mainly used to run signals among integrated circuits placed on a similar PCB. | It is utilized for connecting digital audio devices. |
| It uses two lines between several masters & slaves like SDA & SCL**.** | It uses three lines WS, SCK  & SD. |
| It supports multi-master & multi-slave. | It supports a single master. |
| This protocol supports CLK stretching. | This protocol does not have CLK stretching. |
| I2C includes additional overhead start & stop bits. | I2S does not include any start & stop bits. |

**Advantages**

The **advantages of the I2S bus** include the following.

* I2S utilizes separate CLK & serial data lines. So it has very simple receiver designs as compared to asynchronous systems.
* It is a single master device so there is no issue with data synchronization.
* The microphone based on I2S o/p does not need an analog front end but is utilized within a wireless microphone by using a digital transmitter. By using this, you can have a totally digital connection between the transmitter & transducer.

**Disadvantages**

The **disadvantages of the I2S bus** include the following.

* I2S is not proposed for transferring data through cables.
* I2S is not supported within high-level applications.
* This protocol has a synchronization problem between three signal lines which is noticed at high bit rate & sampling frequency. So this problem mainly occurs because of the variation of propagation delays among clock lines & data lines.
* I2S does not include an error detection mechanism, so it can cause errors within data decoding.
* It is mainly utilized for inter-IC communication on a similar PCB.
* There are no typical connectors & interconnecting cables for I2S, so different designers use different connectors.

**Applications**

The **applications of the I2S protocol** include the following.

* I2S is used for connecting digital audio devices.
* This protocol is extensively utilized in transferring audio data from a DSP or microcontroller to an audio codec to play audio.
* Initially, the I2S interface is utilized within CD player designs. Now, it can be found where digital audio data is being sent between ICs.
* I2S is used in DSPs, audio ADCs, DACs, microcontrollers, sample rate converters, etc.
* I2S is particularly designed to use in between integrated circuits for communicating digital audio data.
* This protocol plays a key role in connecting the microcontroller &  its peripheral devices when the I2S focuses on the audio data transmission between digital audio devices.

Thus, this is all about an overview of the **I2S protocol specification** which includes working, differences, and its applications. [I²S is a 3 wire synchronous serial protocol](https://en.wikipedia.org/wiki/I%C2%B2S) used to transfer digital stereo audio between two integrated circuits. The **I2S Protocol Analyzer** is a signal decoder that includes all DigiView Logic Analyzers. This DigiView software simply provides wide search, navigation, export, measurement, plot & printing capabilities to all types of signals. Here is a question for you, what is the I3C protocol?