VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

—o0o—

LABORATORY REPORT

Waveform generator

SUPERVISOR: Nguyễn Tuấn Hùng

SUBJECT: Digital signal processing on FPGA

GROUP: 08

List of Members

STT	MSSV	Họ Và Tên	Lớp
1	2213874	Nguyễn Thanh Tùng	L01
2	2210780	Nguyễn Đại Đồng	L01
3	2213496	Nguyễn Quốc Tín	L01

Ho Chi Minh, ../../20..

Contents

List of Figures

List of Tables

Chapter 1. Theoretical Background

1.1 Introduction of the DE10-Standard Board

1.1.1 Layout and Components

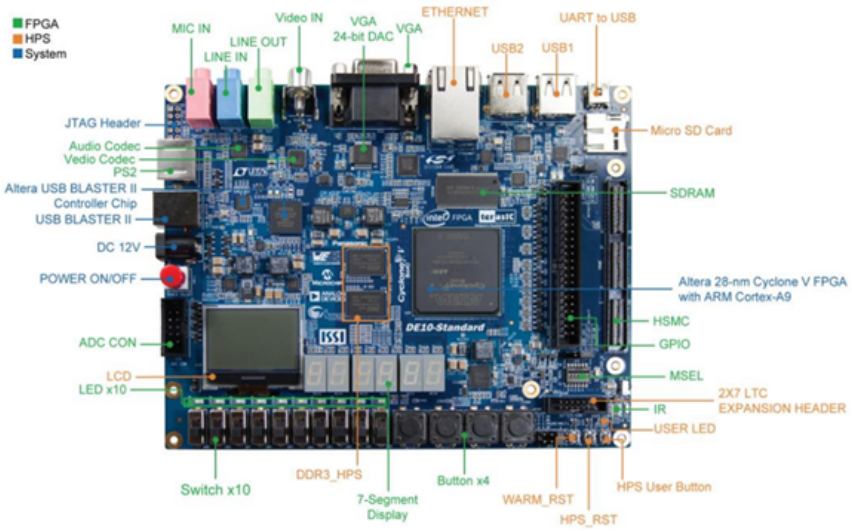

Figure 1.1: DE10-Standard development board.

The DE10-Standard board has many features that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the board:

- **FPGA**
  - + Altera Cyclone® V SE 5CSXFC6D6F31C6N device
  - + Altera serial configuration device – EPCS128
  - + USB-Blaster II onboard for programming; JTAG Mode
  - + 64MB SDRAM (16-bit data bus)
  - + 4 push-buttons
  - + 10 slide switches
  - + 10 red user LEDs

+ Six 7-segment displays
+ Four 50MHz clock sources from the clock generator
+ 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
+ VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
+ TV decoder (NTSC/PAL/SECAM) and TV-in connector
+ PS/2 mouse/keyboard connector
+ IR receiver and IR emitter
+ One HSMC with Configurable I/O standard 1.5/1.8/2.5/3.3
+ One 40-pin expansion header with diode protection
+ A/D converter, 4-pin SPI interface with FPGA

- **HPS (Hard Processor System)**

+ 800MHz Dual-core ARM Cortex-A9 MPCore processor
+ 1GB DDR3 SDRAM (32-bit data bus)
+ 1 Gigabit Ethernet PHY with RJ45 connector
+ 2-port USB Host, normal Type-A USB connector
+ Micro SD card socket
+ Accelerometer (I2C interface + interrupt)
+ UART to USB, USB Mini-B connector
+ Warm reset button and cold reset button
+ One user button and one user LED
+ LTC 2x7 expansion header
+ 128x64 dots LCD Module with Backlight

1.1.2 Block Diagram of the DE10-Standard Board

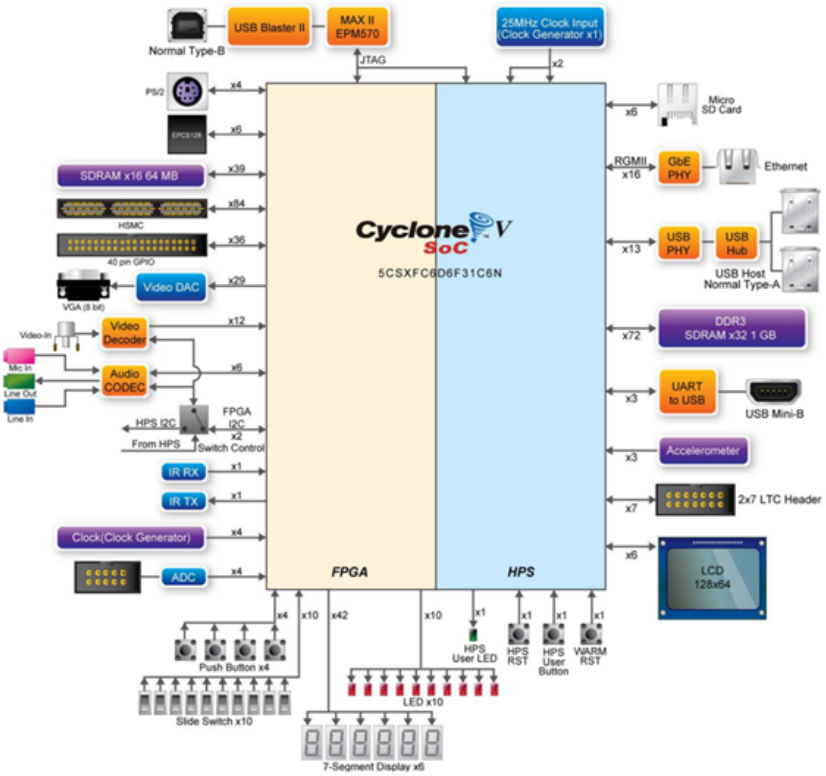

Figure 1.2: Block diagram of DE10-Standard.

All the connections are established through the Cyclone V SoC FPGA device to provide maximum flexibility for users. Detailed information about 1.2:

- **FPGA Device**
  - + Cyclone V SoC 5CSXFC6D6F31C6N Device
  - + Dual-core ARM Cortex-A9 (HPS)
  - + 110K programmable logic elements
  - + 5,140 Kbits embedded memory
  - + 6 fractional PLLs
  - + 2 hard memory controllers
  - + 3.125G transceivers
- **Configuration and Debug**

+ Quad serial configuration device – EPCS128 on FPGA
+ Onboard USB-Blaster II (normal type B USB connector)

- **Memory Device**

+ 64MB (32Mx16) SDRAM on FPGA
+ 1GB (2x256Mx16) DDR3 SDRAM on HPS
+ Micro SD card socket on HPS

- **Communication**

+ Two port USB 2.0 Host (ULPI interface with USB type A connector)
+ UART to USB (USB Mini-B connector)
+ 10/100/1000 Ethernet
+ PS/2 mouse/keyboard
+ IR emitter/receiver
+ I2C multiplexer

- **Connectors**

+ One HSMC (8-channel Transceivers, Configurable I/O standards 1.5/1.8/2.5/3.3V)
+ One 40-pin expansion headers
+ One 10-pin ADC input header
+ One LTC connector (SPI Master, I2C, and GPIO interface)

- **Display**

+ 24-bit VGA DAC
+ 128x64 dots LCD Module with Backlight

- **Audio**

+ 24-bit CODEC, Line-in, Line-out, and microphone-in jacks

- **Video Input**

+ TV decoder (NTSC/PAL/SECAM) and TV-in connector

- **ADC**

+ Interface: SPI
+ Fast throughput rate: 500 KSPS
+ Channel number: 8
+ Resolution: 12-bit
+ Analog input range: $0 \sim 4.096$ V

- **Switches, Buttons, and Indicators**

+ 5 user Keys (FPGA x4, HPS x1)
+ 10 user switches (FPGA x10)
+ 11 user LEDs (FPGA x10, HPS x1)
+ 2 HPS reset buttons (HPS_RESET_n and HPS_WARM_RST_n)
+ Six 7-segment displays

- **Sensors**

+ G-Sensor on HPS

- **Power**

+ 12V DC input

1.2 Direct Digital Synthesis (DDS)

DDS is a method of generating different Analog waveforms using digital techniques. It operates by saving the points of a waveform in its digital form. Then it reconstructs the waveform by recalling these digital data. It is a technique which uses digital data and analog signal processing blocks to generate signal waveforms that are repetitive in nature. Many years ago, these digital synthesizers were limited by the clock frequency, which would hinder the speed of operation of the digital logic. Now, with increasing frequency, the limits of the DDS are also increasing.

The technique uses digital data processing to generate a frequency- and phase-tunable output related to a fixed frequency reference, or clock source. In a DDS architecture, the reference or system clock frequency is divided down by the scaling factor, set by a programmable binary tuning word. The tuning word is typically 24-48 bits long which enables a DDS implementation to provide superior output frequency tuning resolution. Today's cost-competitive,

high-performance, functionally-integrated, and small package-sized DDS products are fast becoming an alternative to traditional frequency-agile analog synthesizer solutions. The integration of a high-speed, high-performance, D/A converter and DDS architecture onto a single chip (forming what is commonly known as a Complete-DDS solution) enabled this technology to target a wider range of applications and provide, in many cases, an attractive alternative to analog-based PLL synthesizers. For many applications, the DDS solution holds some distinct advantages over the equivalent agile analog frequency synthesizer employing PLL circuitry.

Simply stated, a direct digital frequency synthesizer translates a train of clock pulses into an analog waveform, typically a sine, triangular, or square wave. As Figure 1 shows, its essential parts are: a *phase accumulator*, which produces a number corresponding to a phase angle of the output waveform, a *phase-to-digital converter(LUT)*, which generates the instantaneous digital fraction of the output amplitude occurring at a particular phase angle, and a *digital-to-analog converter (DAC)*, which converts that digital value to a sampled analog data point.

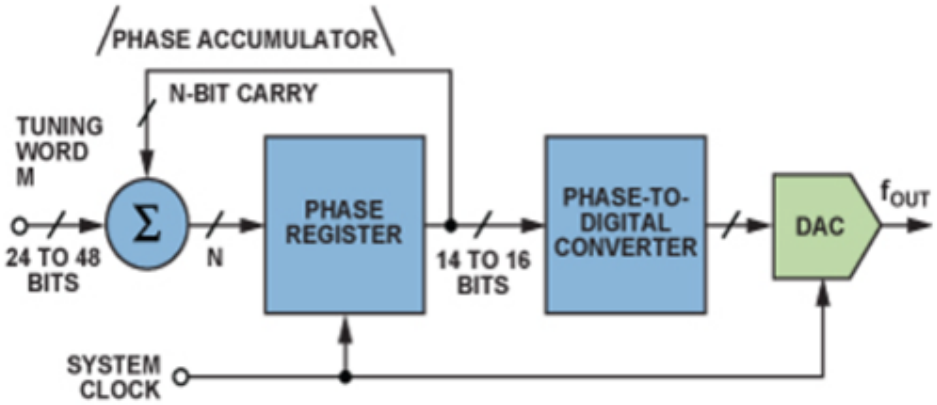

Figure 1.3: Functional block diatram for DDS system.

On each clock cycle, the phase accumulator adds a fixed value known as the *frequency tuning word (fcw)*. The result is a digital ramp representing phase over time, which is used to address the LUT. The LUT stores samples of the desired waveform (typically a sine wave), and the output is then converted to analog by the DAC and smoothed by the LPF.

For sine-wave outputs, the phase-to-digital converter is usually a sine lookup table in figure

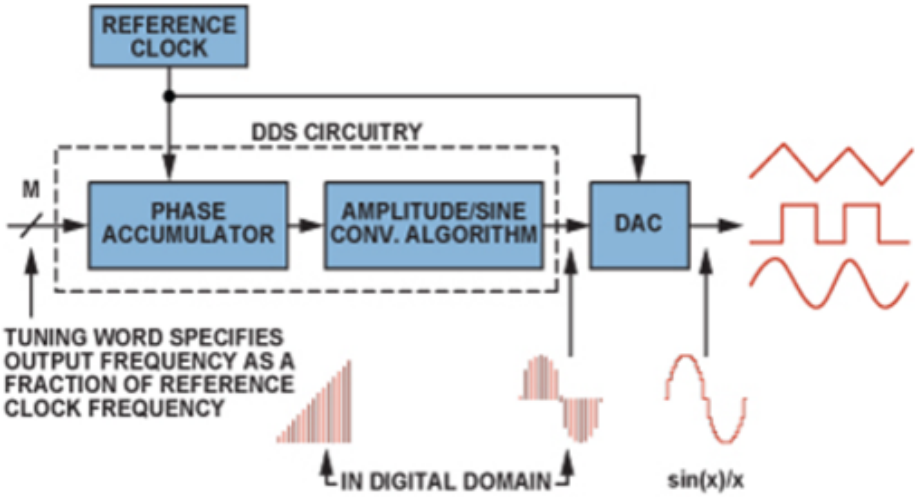

Figure 1.4: Typical DDS architecture and signal path with DAC.

1.3 Phase accumulator and how to generate waveform

The Phase Accumulator is an n-bit register that operates with modulo 2^n arithmetic. At every clock cycle, it adds a fixed value M, called the **Frequency Control Word (FCW)** or tuning word, to its current phase value.

- The **FCW** determines the phase step size.
- A larger FCW increases the phase more quickly ⇒ higher output frequency.
- A smaller FCW increases the phase more slowly ⇒ lower output frequency.

Operating principle: On each rising edge of the system clock, the accumulator updates its value by adding the FCW. This process creates a steadily increasing phase ramp that cycles from 0 to $2^n - 1$. When the maximum value is reached, the accumulator **overflows** and wraps around to 0, similar to a clock hand completing a full revolution. This mechanism is often called a *digital phase wheel* as show in Figure 1.5.

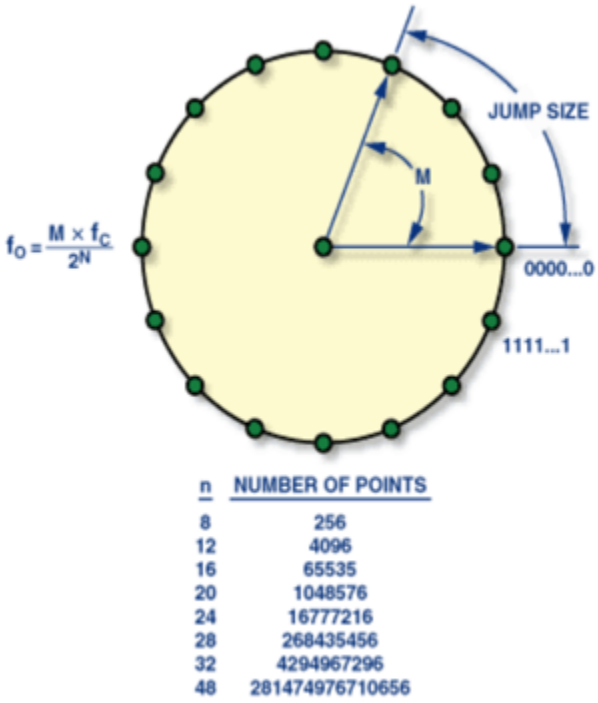

Figure 1.5: Digital phase wheel.

$$f_{out} = \frac{M}{2^n} \cdot f_{clk}$$

where:

+ f_{out} : output frequency
+ f_{clk} : system clock frequency
+ M : Frequency Control Word (FCW)
+ n : number of bits in the phase accumulator

1.4 Concept of Waveform Generation

1.4.1 Sine Wave

Implementation idea: The total has 1024 samples, which means storing the entire $0° \rightarrow 360°$ range for one full sine wave cycle. However, this would require four times the ROM size

$\rightarrow$ instead, it is sufficient to store only $\frac{1}{4}$ of the cycle, that is $0° \rightarrow 90°$.

First, the MSB bit is used to distinguish between the positive sine region ($0° \rightarrow 180°$, the first two quadrants) and the negative sine region ($180° \rightarrow 360°$, the last two quadrants). Next, within each region, the bit [MSB-1] is used as a symmetry marker to indicate when the address should be mirrored, thereby producing a complete half sine wave.

For the first positive half cycle, the phase step is selected depending on the desired output frequency. The phase accumulator will accumulate and gradually increase the address in the first quadrant according to the stored $\frac{1}{4}$ samples when bit [MSB-1] = 0. If bit [MSB-1] = 1, then a mirror operation is performed by taking the Maximum – address of the first quadrant or simply applying the NOT operation to the address. This produces the second quadrant and completes the first positive half cycle of the sine wave.

For the negative half cycle, a two's complement operation is required to switch from positive to negative values when MSB = 1. Then, mirroring is applied to obtain the third quadrant. Finally, the last quadrant is mirrored again from the third quadrant.

In summary, the MSB determines the positive or negative region, while bit [MSB-1] determines whether mirroring is applied or not.

1.4.2 ECG Waveform

Since the ECG waveform is complex, the group uses a HEX file as the LUT with 1024 samples. Similarly, the phase accumulator accumulates phase and increases the address, with each address containing one amplitude value of the ECG signal. As a result, the ROM sequentially outputs each ECG sample. By combining these samples over time, a complete ECG waveform is formed.

1.4.3 Square Wave

For the square wave, the implementation is simpler and does not require a LUT. Specifically:

The phase accumulator acts as a "time ruler" for one waveform cycle. A square wave has two levels: HIGH (24'h0FFF00) and LOW (24'h0). The waveform is determined by comparing the accumulated phase with predefined thresholds.

The duty cycle values are selected as desired. From this, the threshold values are de-

termined to switch between HIGH and LOW. For example, with a duty cycle of 90% of 1024 samples, it requires counting $10'd102$ samples to create the LOW level first, and this count is chosen as the threshold. Once the threshold is exceeded, the signal switches to HIGH. The same approach is used for other duty cycle values.

For some special duty cycles such as 50%, the symmetry of the MSB can be exploited. After counting half a cycle with MSB = 0, the state changes at MSB = 1. For 75% or 25% duty cycles, an OR operation between the two MSB bits can be used.

1.4.4 Sawtooth Wave

The sawtooth waveform is generated by accumulating phase, similar to other waveforms. By exploiting the wrap-around phenomenon, the phase increases linearly as a ramp and resets immediately to 0 when overflow occurs.

1.4.5 Triangle Wave

The triangle waveform is derived from the sawtooth waveform with two linear ramps: one increasing in the first half and one decreasing in the second half. This is implemented by accumulating phase to form a linear ramp (sawtooth), then exploiting symmetry with the MSB to invert the ramp in the second half, or by taking Maximum – accumulated value from the first ramp.

1.5 Configuration for the WM8731 Audio Codec

1.5.1 Audio Codec WM8731 Overview

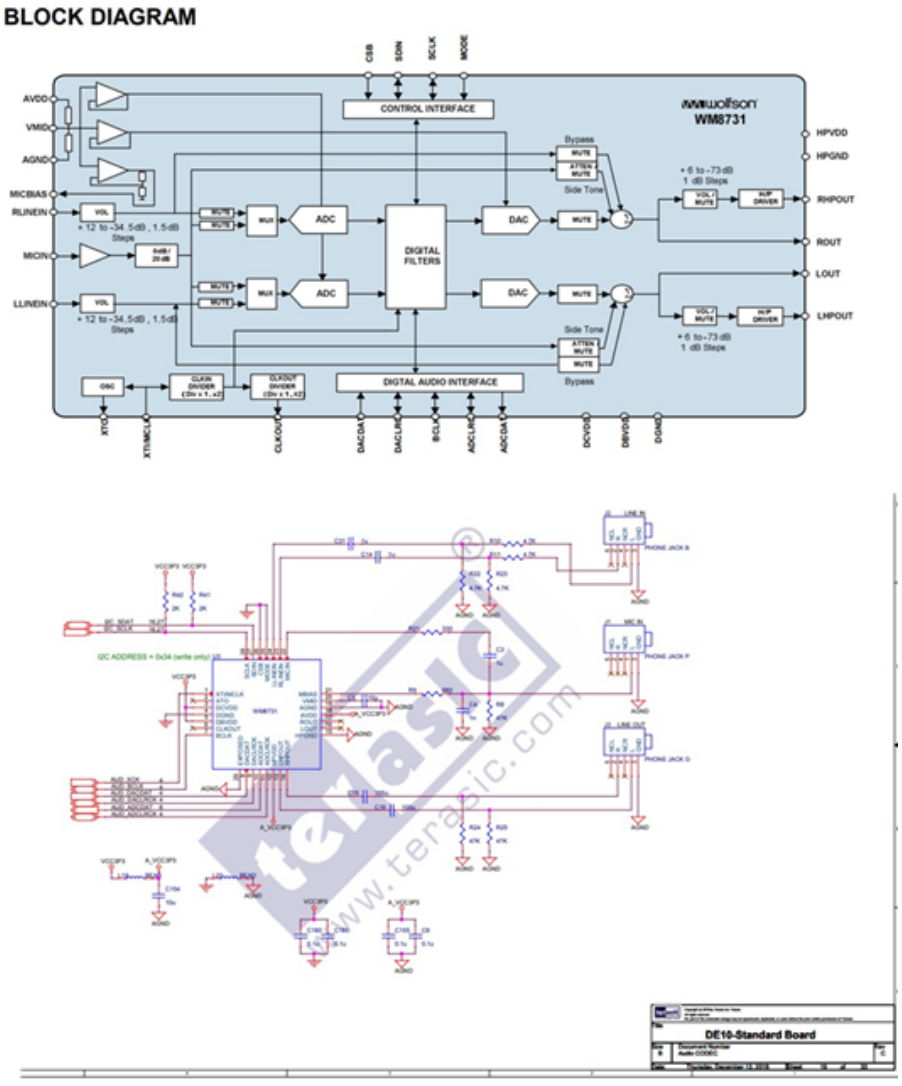

Figure 1.6: Block Diagram of the Integrated ADC/DAC Audio Codec in the WM8731.

a) **Role:** The WM8731 is an integrated audio ADC/DAC that supports stereo audio signal processing.

b) **Application:** It is used for audio capture/playback via the I^2S interface and is configured through I^2C/SPI.

c) **Supported Modes:** The WM8731 provides multiple operating features:

- 24-bit ADC and DAC using sigma-delta architecture with digital decimation/interpolation filters, providing high SNR and noise reduction.
- Control interface via two-wire or three-wire communication (commonly I^2C) for full access to configuration registers (volume, mute, de-emphasis, and other modes).
- Digital audio data interface supporting multiple formats (I^2S, Left Justified, Right Justified, DSP mode), with data word lengths from 16 to 24 bits, and sampling rates from 8 kHz to 96 kHz.
- Utility functions: integrated headphone driver with efficient output stage, mute function, and flexible power management system with multiple power-down modes for individual blocks.

d) **Important Pins:**

Table 1.1: Key WM8731 Connections with DE10

WM8731 Pin	**Connected to DE10**	**Function**
SCLK	I2C_SCLK (I^2C clock)	I^2C clock
SDIN	I2C_SDAT (I^2C data)	I^2C data
MCLK (XTI)	AUD_XCK [3:0]	Main codec clock (typically 12 MHz or 18.432 MHz)
BCLK	AUD_BCLK [3:0]	I^2S bit clock (e.g., 3.072 MHz = 48 kHz × 64)
DACLRC	AUD_DACLRCK [3:0]	Left/Right clock for DAC
DACDAT	AUD_DACDAT [3:0]	I^2S data output, audio data from FPGA → WM8731

Pins `SCLK` and `SDIN` require pull-up resistors (2.2kΩ–10kΩ) to ensure reliable I^2C operation. The master clock (`MCLK`) must be generated by the DE10 PLL with precise frequency (e.g., 12.288 MHz for a 48 kHz sampling rate).

e) **I^2C Addressing:** In the I^2C protocol, each device is assigned a 7-bit (or sometimes 10-bit) address. For the DE10 system, the WM8731 codec is configured as a slave device

with the I^{2}C address `0x34` (write-only mode).

Since WM8731 only supports configuration via write operations, the effective 7-bit slave address is:

$$0x34 \text{ (hex)} = 0110100 \text{ (binary)}$$

The 8-bit I^{2}C frame address is then:

$$0x68 \text{ (hex)} = 01101000 \text{ (binary)}$$

where the 7-bit device address is followed by a single R/W bit (set to 0 for write).

1.5.2 Register Configuration for WM8731

The WM8731 codec is configured to operate according to the requirements of the Lab. These configuration registers are transmitted via the I^2C protocol.

Table 1.2: Register configuration for WM8731

Index	**Address**	**Data**	**Register**	**Function**
0	0x0F	0x00	Reset Register	Reset the entire codec to its default state
1	0x06	0x10	Power Down Control	Disable MIC input, enable LINEOUT, LINEIN, ADC, DAC
2	0x02	0x79	Left Headphone Out	Left headphone volume: 0 dB, Zero-Cross detect enabled
3	0x03	0x79	Right Headphone Out	Right headphone volume: 0 dB, Zero-Cross detect enabled
4	0x00	0x17	Left Line In	Left line-in volume: 0 dB, not muted
5	0x01	0x17	Right Line In	Right line-in volume: 0 dB, not muted
6	0x04	0x10	Analog Audio Path Control	Select LINEIN → ADC, disable MIC, disable bypass, enable DAC

Next page...

Table 1.2 Register configuration for WM8731

Index	Address	Data	Register	Function
7	0x05	0x00	Digital Audio Path Control	Disable high-pass filter, disable de-emphasis, disable DAC soft-mute
8	0x07	0x0A	Digital Audio Interface Format	Select Left-justified mode, 24-bit, MSB first
9	0x08	0x01	Sampling Control	USB mode, MCLK = 12 MHz, Fs = 48 kHz
10	0x09	0x01	Active Control	Activate codec (ACTIVE = 1)
11	0x06	0x02	Power Down Control	Keep ADC, DAC, LINEIN, LINEOUT active; only power down Microphone

1.5.3 I^2C Protocol

a) **Overview:**

I^2C stands for **Inter-Integrated Circuit**. It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it has become a widely used protocol for short-distance communication. It is also known as the ***Two-Wired Interface (TWI)***.

I^2C uses only two bi-directional open-drain lines for data communication, called **SDA** and **SCL**. Both these lines are pulled high.

- *Serial Data (SDA):* Transfer of data takes place through this pin.
- *Serial Clock (SCL):* It carries the clock signal.

I^2C operates in two modes:

- Master mode
- Slave mode

According to the I^2C protocol rules, the data line cannot change when the clock line is high; it can change only when the clock line is low. Since the two lines are *open-drain*,

pull-up resistors are required to keep the lines at a high level, as the devices on the I^2C bus are active low.

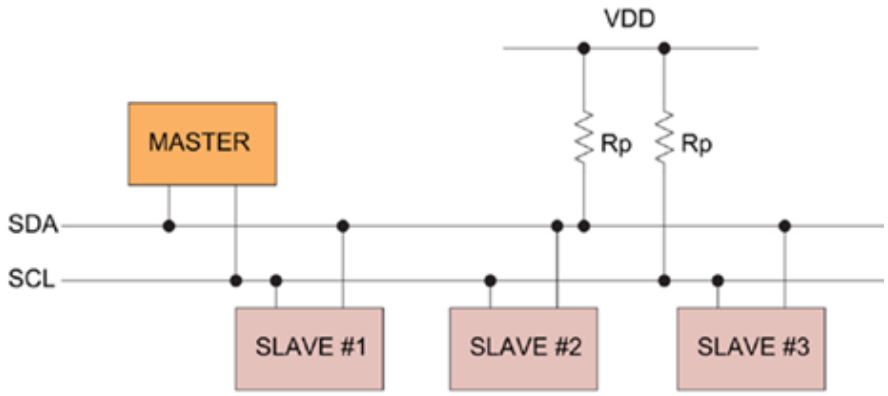

Figure 1.7: Generalized I^2C Connection Diagram.

b) **How I^2C works:**

With I^2C, data is transferred to **messages**. Messages are broken up into **frames** of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame:

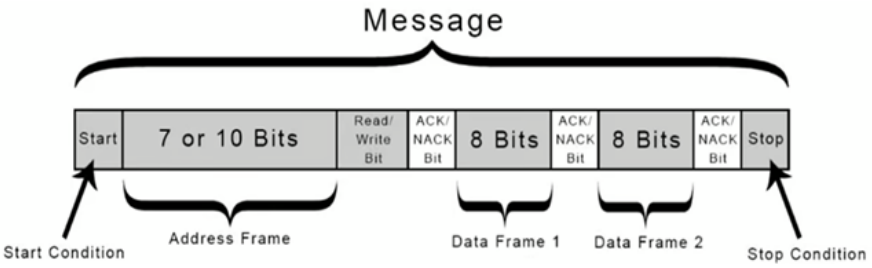

Figure 1.8: Messages are broken up into frames of data.

- *Start Condition:* The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.
- *Stop Condition:* The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.

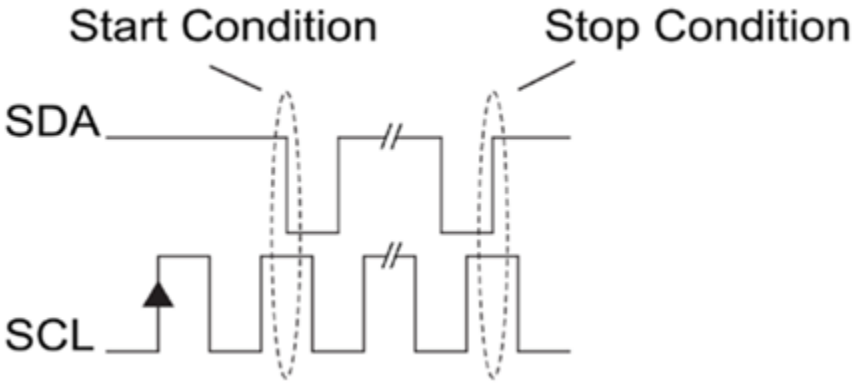

Figure 1.9: Start Condition And Stop Condition Transitions.

- *Address:* Frame: A 7- or 10-bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
- *Read/Write Bit:* A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
- *ACK/NACK Bit:* Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

c) **Addressing**

I^2C doesn't have slave select lines like SPI, so it needs another way to let the slave know that data is being sent to it, and not another slave. It does this by addressing. The address frame is always the first frame after the start bit in a new message.

The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage *ACK* bit back to the master. If the address doesn't match, the slave does nothing, and the *SDA* line remains high.

A slave address is sent in 8-bit byte format, MSB first, but the last bit signifies whether the transaction will be read or write to the slave. In effect, the upper 7 bits constitute the slave address, while the 8th bit serves as a *READ/WRITE* command bit. Thus, there is an address space of 128 unique addresses for addressing up to 128 slaves.

d) **Acknowledge and Not Acknowledge Bits (ACK/NACK)**

As a form of feedback, after every byte transmission the receiving device sends an Acknowledge or Not Acknowledge bit. An Acknowledge bit is generated by the receiver

by holding the SDA line low during a HIGH *SCL* period, while a Not Acknowledge bit is generated when the receiver leaves the *SDA* line passively pulled HIGH and does not respond in any way. This fact implies that in response to an address byte, all unmatched SLAVEs send a Not Acknowledge bit by not responding.

An ACK is used to denote that a byte (address or data) was transmitted and received successfully and that the transmission can continue to the next byte transfer, a stop condition or a repeated start. A NACK is generally used by the receiver to indicate whether an error occurred somewhere in the data transmission. This is used to signal to the transmitting device to terminate the transmission immediately or to make another attempt by sending a repeated start.

e) **WM8731 I^{2}C Data Transmission Steps**

Step 1. The master sends the start condition to every connected slave by switching the *SDA* line from a high voltage level to a low voltage level before switching the *SCL* line from high to low.

Step 2. The master sends each slave the 7-bit address (for the WM8731, the 7-bit I^{2}C slave address is typically `0x1A`) of the slave it wants to communicate with, along with the read/write bit.

The WM8731 supports a 2-wire MPU serial interface. The device operates as a slave device only. The WM8731 has one of two slave addresses that are selected by setting the state of pin 10, (CSB).

CBS State (Default = LOW)	Address
0	0011010
1	0011011

Table 1.3: 2-Wire MPU Interface Address Selection (Select Slave address to communicate).

Step 3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the *SDA* line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the *SDA* line high.

Step 4. The master sends or receives the data frame.

Step 5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.

Step 6. After successfully receiving the ACK (confirming that the SLAVE is ready to communicate), SDA will continue sending the configuration register DATA, which consists of the 7-bit register address (DATA $B_{15} - B_9$) and the 9-bit configuration value for that register (DATA $B_8 - B_0$), for a total of 16 bits divided into 2 frames.

In the first transmission, the frame containing the register address is sent after the ACK has been received. Since *SDA* sends data in 8-bit frames, the register address frame must take the MSB from the 9-bit register configuration value as the LSB of the address frame and send it first (DATA $B_{15} - B_8$).

Step 7. When the data transmission is completed, the process continues by waiting to check whether the ACK bit is returned. If successful, proceed to send the remaining frame, which contains the 8-bit configuration data (DATA $B_7 - B_0$).

Step 8. After transmission, wait for the ACK bit response to verify whether the DATA has been successfully sent.

Step 9. After confirming that the ACK has been successfully received, proceed to the stop process. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching *SDA* high. When the STOP condition is successfully met, it means the configuration data for one register has been fully transmitted. To perform the same operation for other registers, simply repeat all the steps from the beginning up to the STOP condition.

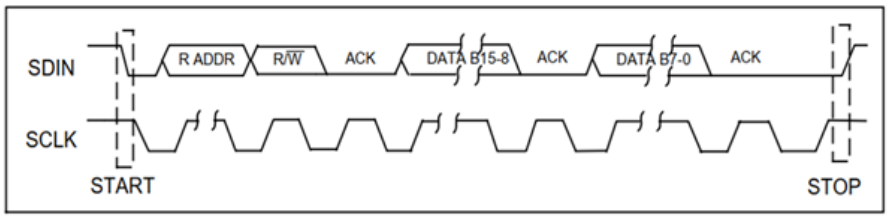

Figure 1.10: 2-Wire Serial Interface of WM8731.

The I^2C communication protocol is a simple and effective way for devices to communicate with each other. It allows multiple devices to connect using just two wires, making it easy to add new components to a system. I^2C is popular in various applications because it supports multiple devices, is relatively easy to implement, and requires less wiring compared to other

protocols. Overall, I2C is a reliable choice for connecting sensors, displays, and other peripherals in electronic projects.

1.5.4 I^2S Communication Protocol

The protocol which is used to transmit digital audio data from one device to another is known as I^2S or Inter-IC Sound protocol. This protocol transmits PCM (pulse-code modulated) audio data from one IC to another within an electronic device. I^2S 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 I^2S protocols, so you cannot play OGG or MP3 or other audio formats that condense the audio, however, you can play WAV files.

In the WM8731 datasheet, it is specified that I^2S mode operates as follows:

I^2S mode is where the MSB (most-significant bit) is available on the 2nd rising edge of BCLK (Bit Clock) following an LRCLK/DACLRC (Left/Right Clock) transition - meaning the MSB is delayed by one BCLK compared to the left-justified format. This is the standard method to synchronize left/right channel data with the LRCLK signal, as illustrated in figure 1.11 below.

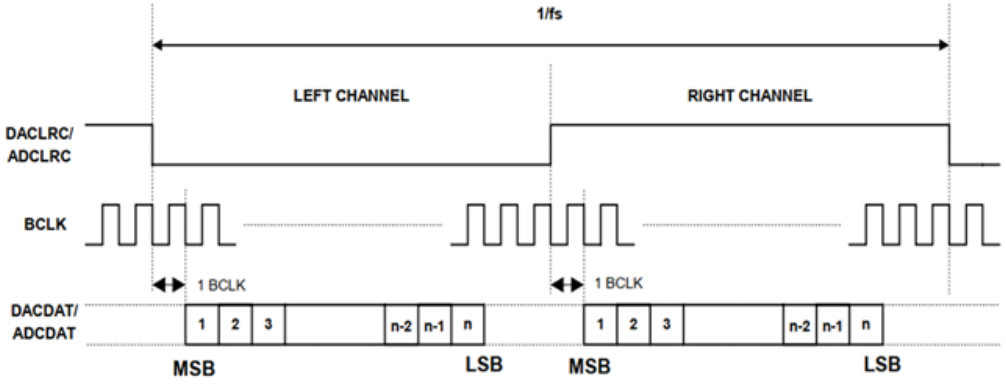

Figure 1.11: I^2S mode.

LRCLK must always transition on the falling edge of *BCLK*, as stated in the datasheet. This requirement must be met when interfacing devices in both master and slave modes. Each data bit is changed/shifted on the high-to-low transition of *BCLK* and sampled on the low-to-high transition of *BCLK*.

According to the provided I^2C register configuration, the WM8731 is set to:

- **Slave Mode:** When WM8731 operates in Slave mode, both BCLK and DACLRC are inputs, as shown in the figure 1.12.

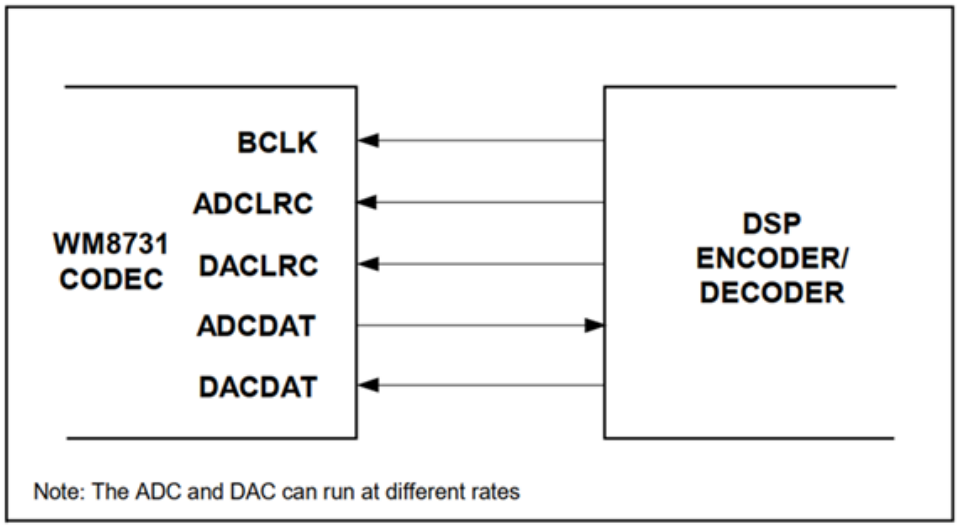

Figure 1.12: I^2S slave mode.