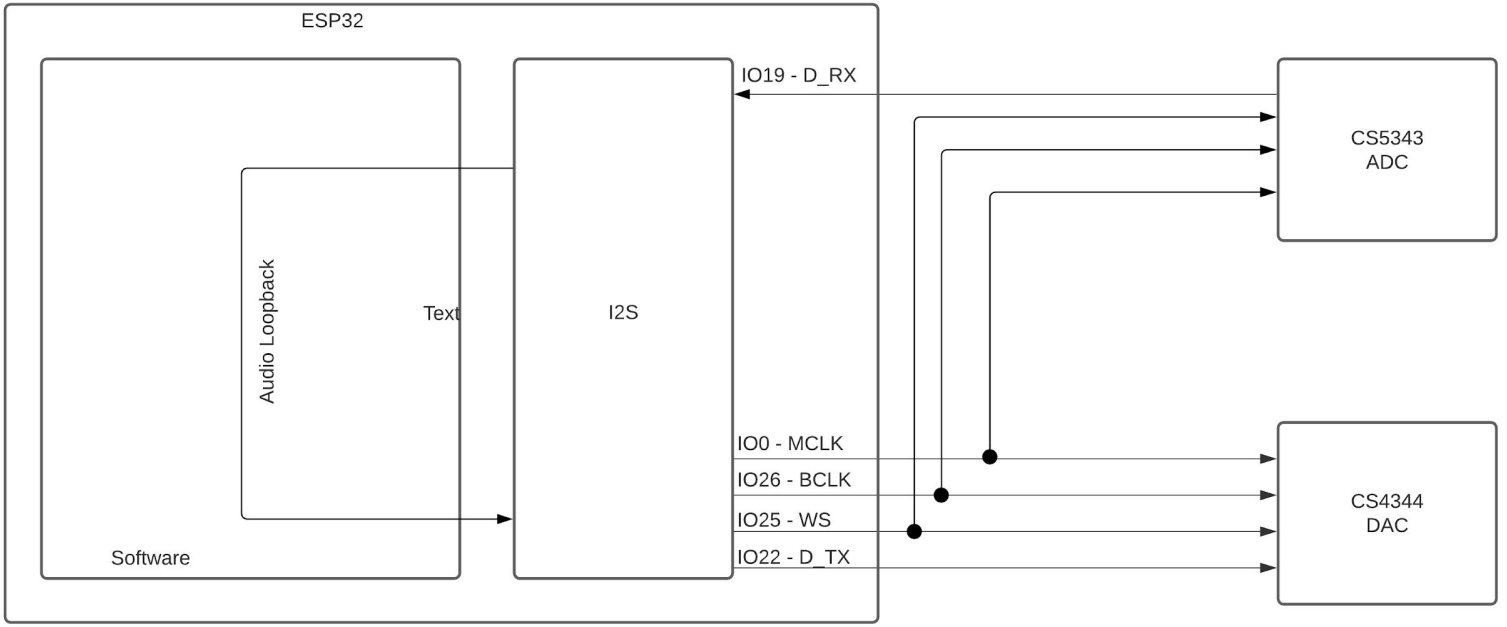
Audio DSP on ESP32

I²S Setup

System diagram



ESP32 driver overview

Functional Overview

Installing the Driver

Install the I2S driver by calling the function :cpp:func`i2s_driver_install` and passing the following arguments:

- Port number
- The structure `i2s_config_t` with defined communication parameters
- Event queue size and handle

Configuration example:

```
static const int i2s_num = 0; // i2s port number

static const i2s_config_t i2s_config = {
    .mode = I2S_MODE_MASTER | I2S_MODE_TX,
    .sample_rate = 44100,
    .bits_per_sample = 16,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = I2S_COMM_FORMAT_STAND_I2S,
    .intr_alloc_flags = 0, // default interrupt priority
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = false
};

i2s_driver_install(I2S_NUM, &i2s_config, 0, NULL);
```

- ESP32 has 2 bidirectional I²S modules (up to 4 channels in + out simultaneously)
- Driver has no support for MCLK

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/i2s.html

How to setup MCLK?

ESP32 supports MCLK by outputting the internal I2Sn_CLK (from which also I2Sn_BCLK is derived) on GPIO 0, 1 and 3

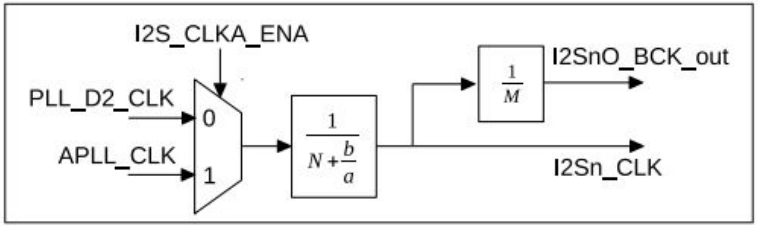


Figure 62: I²S Clock

5.10 IO_MUX Pad List

Table 20 shows the IO_MUX functions for each I/O pad:

Table 20: IO_MUX Pad Summary

GPIO	Pad Name	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Reset	Notes
0	GPIO0	GPIO0	CLK_OUT1	GPIO0	-	-	EMAC_TX_CLK	3	R
1	U0TXD	U0TXD	CLK_OUT3	GPIO1	-	-	EMAC_RXD2	3	-
2	GPIO2	GPIO2	HSPIWP	GPIO2	HS2_DATA0	SD_DATA0	-	2	R
3	U0RXD	U0RXD	CLK_OUT2	GPIO3	-	-	-	3	-
4	GPIO4	GPIO4	HSPIHD	GPIO4	HS2_DATA1	SD_DATA1	EMAC_TX_ER	2	R
5	GPIO5	GPIO5	VSPICS0	GPIO5	HS1_DATA6	-	EMAC_RX_CLK	3	-

https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

How to setup MCLK?

Two registers must be modified “by hand” (decide before which CLK_OUT to use

1. IO_MUX_GPIO0_REG -> must be set to CLK1 internal signal
2. Route I2S0 internal CLK to CLK_OUT via IO_MUX_PIN_CTRL

Register 5.33: IO_MUX_PIN_CTRL (0x3FF49000)

(reserved)	PIN_CTRL_CLK3	PIN_CTRL_CLK2	PIN_CTRL_CLK1	
31 … 12	11 … 8	7 … 4	3 … 0	
0x0	0x0	0x0	0x0	Reset

If you want to output clock for I2S0 to:

CLK_OUT1, then set PIN_CTRL[3:0] = 0x0;

CLK_OUT2, then set PIN_CTRL[3:0] = 0x0 and PIN_CTRL[7:4] = 0x0;

CLK_OUT3, then set PIN_CTRL[3:0] = 0x0 and PIN_CTRL[11:8] = 0x0.

Register 5.34: IO_MUX_*x*_REG (*x*: GPIO0-GPIO39) (0x10+4x*)**

(reserved)	MCU_SEL	FUN_DRV	FUN_IE	FUN_WPU	FUN_WPD	MCU_DRV	MCU_IE	MCU_WPU	MCU_WPD	SLP_SEL	MCU_OE	
31 … 15	14 … 12	11 10	9	8	7	6 5	4	3	2	1	0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x0	0x2	0	0	0	0x0	0	0	0	0	0	Reset

MCU_SEL Select the IO_MUX function for this signal. 0 selects Function 1, 1 selects Function 2, etc. (R/W)

Thanks to jgustavoam from https://www.esp32.com/viewtopic.php?t=14185

I2S MCLK speed?

ESP32 I2S driver will force either MCLK = 256*fs or 384*fs depending on selected bit-width

Our config (44.1 kHz, 32 bit) will end up in 256*fs as 256%32 = 0

```
esp_err_t i2s_set_clk(i2s_port_t i2s_num, uint32_t rate, i2s_bits_per_sample_t bits, i2s_channel_t ch)
{
    int factor = (256%bits)? 384 : 256; // According to hardware codec requirement(supported 256fs or 384fs)
    int clkmInteger, clkmDecimals, bck = 0;
    double denom = (double)1 / 64;
    int channel = 2;
    i2s_dma_t *save_tx = NULL, *save_rx = NULL;
```

See driver: esp-idf/components/driver/i2s.c

ESP32 I2S driver-init with MCLK

```
i2s_config_t i2s_config = {

    .mode = I2S_MODE_MASTER | I2S_MODE_TX  | I2S_MODE_RX,
    .sample_rate = 44100,
    .bits_per_sample = 32,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = I2S_COMM_FORMAT_STAND_MSB,
    .dma_buf_count = 6,
    .dma_buf_len = 512,
    .intr_alloc_flags = 0,
    .tx_desc_auto_clear = true
};


i2s_driver_install(0, &i2s_config, 0, NULL);

i2s_pin_config_t pin_config = {
    .bck_io_num = 26,
    .ws_io_num = 25,
    .data_out_num = 22,
    .data_in_num = 19
};

i2s_set_pin(0, &pin_config);

//enable MCLK on GPIO0
REG_WRITE(PIN_CTRL, 0xFF0);
PIN_FUNC_SELECT(PERIPHS_IO_MUX_GPIO0_U, FUNC_GPIO0_CLK_OUT1);
```