I2S EXAMPLE

```
#include <stdio.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <driver/i2s.h>
#include <esp_system.h>
#include <math.h>


#define SAMPLE_RATE     (36000)
#define I2S_NUM         (0)
#define WAVE_FREQ_HZ    (100)
#define PI              (3.14159265)
#define I2S_BCK_IO      (GPIO_NUM_26)
#define I2S_WS_IO       (GPIO_NUM_25)
#define I2S_DO_IO       (GPIO_NUM_22)
#define I2S_DI_IO       (-1)
#define SAMPLE_PER_CYCLE (SAMPLE_RATE/WAVE_FREQ_HZ)


static void setup_triangle_sine_waves(int bits) {
  int *samples_data = malloc((((bits+8)/16)*SAMPLES_PER_CYCLE*4);
  unsigned int i, sample_val;
  double sin_float, triangle_float, triangle_step
      = (double) pow(2, bits) / SAMPLES_PER_CYCLE;
  size_t i2s_bytes_write = 0;

  printf("\r\nTest bits = %d free mem = %d, written_data = %d\n",
      bits, esp_get_free_heap_size(), (((bits+8)/16)*
                                       SAMPLES_PER_CYCLE*4));
```

```
triangle_float = -(pow(2, bits)/2 -1);
for(i=0; i<SAMPLE_PER_CYCLE; i++) {
  sin_float = sin(i * PI/180.0);
  if (sin_float >= 0) {
    triangle_float += triangle_step;
  } else {
    triangle_float -= triangle_step;
  }
  sin_float *= (pow(2, bits)/2 -1);
  if(bits == 16) {
    sample_val = 0;
    sample_val += (short) triangle_float;
    sample_val = sample_val << 16;
    sample_val += (short) sin_float;
    samples_data[i] = sample_val;
  } else if (bits == 24) {
    samples_data[i*2] = ((int) triangle_float) << 8;
    samples_data[i*2+1] = ((int) sin_float) << 8;
  } else {
    samples_data[i*2] = (int) triangle_float;
    samples_data[i*2+1] = (int) sin_float;
  }
  i2s_set_clk(I2S_NUM, SAMPLE_RATE, bits, 2);
  i2s_write(I2S_NUM, samples_data, ((bits+8)/16)
    *SAMPLES_PER_CYCLE*4, &i2s_bytes_write, 100);
  free(samples_data);
}
```

```
void app_main() {
  i2s_config_t i2s_config = {
    .mode = I2S_MODE_MASTER | I2S_MODE_TX,
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = 16,
    .channel_format = I2S_CHANNEL_FMT_LEFT_RIGHT,
    .communication_format = I2S_COMM_FORMAT_I2S |
                            I2S_COMM_FORMAT_I2S_MSB,
    .dma_buf_count = 6,
    .dma_buf_len = 60,
    .use_apll = false,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
  };
  i2s_pin_config_t pin_config = {
    .bck_io_num = I2S_BCK_IO,
    .ws_io_num = I2S_WS_IO,
    .data_out_num = I2S_DO_IO,
    .data_in_num = I2S_DI_IO,
  };
  i2s_driver_install(I2S_NUM, &i2s_config, 0, NULL);
  i2s_set_pin(I2S_NUM, &pin_config);
  int test_bits = 16;
  while (1) {
    setup_triangle_sine_waves(test_bits);
    test_bits += 8;
    if (test_bits > 32)
      test_bits = 16;
  }
}
```