I2S ADC DAC EXAMPLE

```
#include <stdio.h>
#include <string.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <esp_spi_flash.h>
#include <esp_err.h>
#include <esp_log.h>
#include <esp_partition.h>
#include <driver/i2s.h>
#include <driver/adc.h>
#include <esp_adc_cal.h>
#include "audio_example_file.h"


static const char *TAG = "ad/da";
#define VREF        1100
#define ADC1_TEST_CHANNEL  (ADC1_CHANNEL_7)
#define PARTITION_NAME     "storage"

#define RECORD_IN_FLASH_EN      (1)
#define REPLAY_FROM_FLASH_EN    (1)

#define EXAMPLE_I2S_NUM           (0)
#define EXAMPLE_I2S_SAMPLE_RATE   (16000)
#define EXAMPLE_I2S_SAMPLE_BITS   (16)
#define EXAMPLE_I2S_BUF_DEBUG     (0)
#define EXAMPLE_I2S_READ_LEN      (16 * 1024)
#define EXAMPLE_I2S_FORMAT        (I2S_CHANNEL_FMT_LEFT_RIGHT)
#define
```

```
#define EXAMPLE_I2S_CHANNEL_NUM ((EXAMPLE_I2S_FORMAT
        < I2S_CHANNEL_FMT_ONLY_RIGHT)? (2) : (1))
#define I2S_ADC_UNIT     ADC_UNIT_1
#define I2S_ADC_CHANNEL  ADC1_CHANNEL_0

#define FLASH_RECORD_SIZE  (EXAMPLE_I2S_CHANNEL_NUM
        * EXAMPLE_I2S_SAMPLE_RATE * EXAMPLE_I2S_SAMPLE_BITS
        / 8 * 5)
#define FLASH_ERASE_SIZE  (FLASH_RECORD_SIZE %
        FLASH_SECTOR_SIZE == 0)? FLASH_RECORD_SIZE :
        FLASH_RECORD_SIZE + (FLASH_SECTOR_SIZE -
        FLASH_RECORD_SIZE % FLASH_SECTOR_SIZE)
#define FLASH_SECTOR_SIZE  (0x1000)
#define FLASH_ADDR         (0x200000)


void example_i2s_init() {
  int i2s_num = EXAMPLE_I2S_NUM;
  i2s_config_t i2s_config = {
    .mode = I2S_MODE_MASTER | I2S_MODE_RX | I2S_MODE_TX
        | I2S_MODE_DAC_BUILT_IN | I2S_MODE_ADC_BUILT_IN,
    .sample_rate = EXAMPLE_I2S_SAMPLE_RATE,
    .bits_per_sample = EXAMPLE_I2S_SAMPLE_BITS,
    .communication_format = I2S_COMM_FORMAT_I2S_MSB,
    .channel_format = EXAMPLE_I2S_FORMAT,
    .intr_alloc_flags = 0,
    .dma_buf_count = 2,
    .dma_buf_len = 1024,
    .use_apll = 1,
  };
```

```
    i2s_driver_install(i2s_num, &i2s_config, 0, NULL);
    i2s_set_dac_mode(I2S_DAC_CHANNEL_BOTH_EN);
    i2s_set_adc_mode(I2S_ADC_UNIT, I2S_ADC_CHANNEL);
}

void example_erase_flash() {
#if RECORD_IN_FLASH_EN
    printf("Erasing flash \n");
    const esp_partition_t *data_partition = NULL;
    data_partition = esp_partition_find_first(ESP_PARTITION_TYPE_D
        ESP_PARTITION_SUBTYPE_DATA_FAT, PARTITION_NAME);
    if(data_partition != NULL) {
        printf("partition addr: 0x%.08x, size: %.d, label: %.s
            data_partition->address, data_partition->size,
            data_partition->label);
    }
    printf("Erase size: %d Bytes\n", FLASH_ERASE_SIZE);
    ESP_ERROR_CHECK(esp_partition_erase_range(data_partition,
                    0, FLASH_ERASE_SIZE));
#else:
    printf("Skip flash erasing...\n");
#endif
}

void example_disp_buf(uint8_t *buf, int length) {
#if EXAMPLE_I2S_BUF_DEBUG
    printf("=====\n");
```

```
    for (int i=0; i< length; i++) {
      printf("%.02x", buf[i]);
      if((i+1) %8 == 0) {
        printf("\n");
      }
    }
    printf("=====\n");
#endif
}


void example_reset_play_mode() {
  i2s_set_clk(EXAMPLE_I2S_NUM, EXAMPLE_I2S_SAMPLE_RATE,
      EXAMPLE_I2S_SAMPLE_BITS, EXAMPLE_I2S_CHANNEL_NUM);
}


void example_set_file_play_mode() {
  i2s_set_clk(EXAMPLE_I2S_NUM, 16000, EXAMPLE_I2S_SAMPLE_BITS, 1);
}


int example_i2s_dac_data_scale(uint8_t *d_buff, uint8_t *s_buff,
                               uint32_t len) {
    uint32_t j = 0;
#if (EXAMPLE_I2S_SAMPLE_BITS == 16)
    for (int i=0; i<len; i++) {
      d_buff[j++] = 0;
      d_buff[j++] = s_buff[i];
    }
```

```
    return (len * 2);
#else
    for (int i=0; i<len; i++) {
      d_buff [j++] = 0;
      d_buff [j++] = 0;
      d_buff [j++] = 0;
      d_buff [j++] = s_buff [i];
    }
    return (len * 4);
#endif
}


void example_i2s_adc_data_scale (uint8_t *d_buff,
            uint8_t *s_buff, uint32_t len) {
    uint32_t j = 0;
    uint32_t dac_value = 0;
#if (EXAMPLE_I2S_SAMPLE_BITS == 16)
    for (int i=0; i<len; i+=2) {
      dac_value = ((((uint16_t) (s_buff [i+1] & 0xff)) << 8)
          | ((s_buff [i+0]))));
      d_buff [j++] = 0;
      d_buff [j++] = dac_value * 256 / 4096;
#else
    for (int i=0; i<len; i+=4) {
      dac_value = ((((uint16_t) (s_buff [i+3] & 0xf )) << 8) |
          ((s_buff [i+2]))));
      d_buff [j++] = 0;
      d_buff [j++] = 0;
```

```
        d_buff[j++] = 0;
        d_buff[j++] = dac_value * 256 / 4096;
    }
#endif
}


void example_i2s_adc_dac(void *arg) {
    const esp_partition_t *data_partition = NULL;
    data_partition = esp_partition_find_first(ESP_PARTITION_TYPE_DATA,
            ESP_PARTITION_SUBTYPE_DATA_FAT, PARTITION_NAME);
    if (data_partition != NULL) {
        printf("partition addr: 0x%08x, size: %d, label: %s\n",
        data_partition->address, data_partition->size,
        data_partition->label);
    } else {
        ESP_LOGE(TAG, "Partition error: can't find partition name
                : %s\n", PARTITION_NAME);
        vTaskDelete(NULL);
    }
    example_erase_flash();
    int i2s_read_len = EXAMPLE_I2S_READ_LEN;
    int flash_wr_size = 0;
    size_t bytes_read, bytes_written;

#if RECORD_IN_FLASH_EN
    char* i2s_read_buff = (char*) calloc(i2s_read_len, sizeof(char));
    uint8_t *flash_write_buff = (uint8_t*)
                calloc(i2s_read_len, sizeof(char));
```

```
i2s_adc_enable(EXAMPLE_I2S_NUM);
while (flash_wr_size < FLASH_RECORD_SIZE) {
    i2s_read(EXAMPLE_I2S_NUM, (void*) i2s_read_buff,
        i2s_read_len, &bytes_read, portMAX_DELAY);
    example_disp_buf((uint8_t**) i2s_read_buff, 64);
    esp_partition_write(data_partition, flash_wr_size,
        i2s_read_buff, i2s_read_len);
    flash_wr_size += i2s_read_len;
    ets_printf("Sound recording %u%%\n",
        flash_wr_size * 100 / FLASH_RECORD_SIZE)
}
i2s_adc_disable(EXAMPLE_I2S_NUM);
free(i2s_read_buff);
i2s_read_buff = NULL;
free(flash_write_buff);
flash_write_buff = NULL;
#endif

uint8_t *flash_read_buff = (uint8_t*)
        calloc(i2s_read_len, sizeof(char));
uint8_t *i2s_write_buff = (uint8_t*)
        calloc(i2s_read_len, sizeof(char));
while (1) {
#if REPLAY_FROM_FLASH_EN
for (int rd_offset=0; rd_offset < flash_wr_size; rd_offset += FLASH_SECTOR_SIZE) {
    esp_partition_read(data_partition, rd_offset,
        flash_read_buff, FLASH_SECTOR_SIZE);
    example_i2s_adc_data_scale(i2s_write_buff, EXAMPLE_I2S_NUM,
        flash_read_buff, FLASH_SECTOR_SIZE);
```

```
        i2s_write(EXAMPLE_I2S_NUM, i2s_write_buff,
          FLASH_SECTOR_SIZE, &bytes_written, portMAX_DELAY);
        printf("Playing: %d %%\n", rd_offset * 100 / flash_wr_size);
    }
#endif

    printf("Playing file example: \n");
    int offset = 0;
    int tot_size = sizeof(audio_table);
    example_set_file_play_mode();
    while (offset < tot_size) {
        int play_len = ((tot_size - offset) > (4*1024)) ?
                    (4*1024) : (tot_size - offset);
        int i2s_wr_len = example_i2s_dac_data_scale(
            i2s_write_buff, (uint8_t*)(audio_table + offset), play_len);
        i2s_write(EXAMPLE_I2S_NUM, i2s_write_buff, i2s_wr_len,
                            &bytes_written, portMAX_DELAY);
        offset += play_len;
        example_disp_buf((uint8_t*) i2s_write_buff, 32);
    }

    vTaskDelay(100 / portTICK_PERIOD_MS);
    example_reset_play_mode();
}

free(flash_read_buff);
free(i2s_write_buff);
vTaskDelete(NULL);
}
```

```
void adc_read_task(void *arg){
    adc1_config_width(ADC_WIDTH_12Bit);
    adc1_config_channel_atten(ADC1_TEST_CHANNEL, ADC_ATTEN_11db);
    esp_adc_cal_characteristics_t characteristics;
    while(1){
        uint32_t voltage;
        esp_adc_cal_get_voltage(ADC1_TEST_CHANNEL,
                &characteristics, &voltage);
        ESP_LOGI(TAG, "%d mV", voltage);
        vTaskDelay(200 / portTICK_RATE_MS);
    }
}

esp_err_t app_main(){
    example_i2s_init();
    esp_log_level_set("I2S", ESP_LOG_INFO);
    xTaskCreate(example_i2s_adc_dac, "example_i2s_adc_dac",
            1024*2, NULL, 5, NULL);
    xTaskCreate(adc_read_task, "ADC read task", 2048,
            NULL, 5, NULL);
    return ESP_OK;
}
```