BEÁGYAZOTT RENDSZEREK (C)

2. forduló

BOSCH
Invented for life

A kategória támogatója: Robert Bosch Kft.

Ismertető a feladatlaphoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

> Helyezéseket a 4. forduló után mutatunk, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.
>
> A feltűnően rövid idő alatt megoldott feladatlapok kizárást vonnak maguk után, bármilyen más gyanús esetben fenntartjuk a jogot a forduló érvénytelenítésére!

Jó versenyzést kívánunk!

Miután az előző fordulóban sikeresen bekonfiguráltuk a TAS5727 audio IC-t az STM32F411 segítségével, ebben a fordulóban az audio streamre koncentrálunk.

Az audio stream-et az IC és a mikrovezérlő között az I2S2 periféria és DMA használatával szeretnénk megoldani.

Adatlapok:

https://www.ti.com/lit/ds/symlink/tas5727.pdf?ts=1690751063666

https://www.st.com/resource/en/datasheet/stm32f411ce.pdf

1. feladat 4 pont

Válaszd ki a helyes megoldásokat!

Válaszok

- [ ] A DMA a "Direct Memory Access" rövidítése.
- [ ] A DMA egy speciális hardver amivel a perifériák közvetlenül hozzáférnek a memóriához a processzor közbeavatkozása nélkül.
- [ ] A DMA egy módszer amivel a processzor közvetlenül hozzáfér a perifériákhoz speciális hardver nélkül.
- [ ] A DMA csak a periféria és a memória között tudnak adatot másolni.

2. feladat 6 pont

Mekk mester szeretné DMA-val megoldani az I2S TX transfert a STM32F411 és a TAS5727 között.

Segíts neki kiválasztani a helyes DMA inicializálást.

Válasz

- [ ]

```
#include "stm32f4xx_hal.h"

DMA_HandleTypeDef hdma_spi2_tx;
I2S_HandleTypeDef hi2s2;

void HAL_I2S_DMAInit() {
    hdma_spi2_tx.Instance = DMA1_Stream4;
    hdma_spi2_tx.Init.Channel = DMA_CHANNEL_0;
    hdma_spi2_tx.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_spi2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_spi2_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_spi2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.Mode = DMA_CIRCULAR;
    hdma_spi2_tx.Init.Priority = DMA_PRIORITY_LOW;
    hdma_spi2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_spi2_tx) != HAL_OK) {
```

```
        Error_Handler();
    }

    __HAL_LINKDMA(&hi2s2, hdmatx, hdma_spi2_tx);
}
```

```
#include "stm32f4xx_hal.h"

DMA_HandleTypeDef hdma_spi2_tx;
I2S_HandleTypeDef hi2s2;

void HAL_I2S_DMAInit() {
    hdma_spi2_tx.Instance = DMA1_Stream3;
    hdma_spi2_tx.Init.Channel = DMA_CHANNEL_0;
    hdma_spi2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_spi2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_spi2_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_spi2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.Mode = DMA_CIRCULAR;
    hdma_spi2_tx.Init.Priority = DMA_PRIORITY_LOW;
    hdma_spi2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_spi2_tx) != HAL_OK) {
        Error_Handler();
    }

    __HAL_LINKDMA(&hi2s2, hdmatx, hdma_spi2_tx);
}
```

```
#include "stm32f4xx_hal.h"

DMA_HandleTypeDef hdma_spi2_tx;
I2S_HandleTypeDef hi2s2;

void HAL_I2S_DMAInit() {
    hdma_spi2_tx.Instance = DMA1_Stream4;
    hdma_spi2_tx.Init.Channel = DMA_CHANNEL_0;
    hdma_spi2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_spi2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_spi2_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_spi2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.Mode = DMA_CIRCULAR;
```

```
    hdma_spi2_tx.Init.Priority = DMA_PRIORITY_LOW;
    hdma_spi2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_spi2_tx) != HAL_OK) {
        Error_Handler();
    }

    __HAL_LINKDMA(&hi2s2, hdmatx, hdma_spi2_tx);
}
```

```
#include "stm32f4xx_hal.h"

DMA_HandleTypeDef hdma_spi2_tx;
I2S_HandleTypeDef hi2s2;

void HAL_I2S_DMAInit() {
    hdma_spi2_tx.Instance = DMA1_Stream3;
    hdma_spi2_tx.Init.Channel = DMA_CHANNEL_0;
    hdma_spi2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_spi2_tx.Init.PeriphInc = DMA_PINC_ENABLE;
    hdma_spi2_tx.Init.MemInc = DMA_MINC_DISABLE;
    hdma_spi2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_HALFWORD;
    hdma_spi2_tx.Init.Mode = DMA_CIRCULAR;
    hdma_spi2_tx.Init.Priority = DMA_PRIORITY_LOW;
    hdma_spi2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_spi2_tx) != HAL_OK) {
        Error_Handler();
    }

    __HAL_LINKDMA(&hi2s2, hdmatx, hdma_spi2_tx);
}
```

3. feladat 5 pont

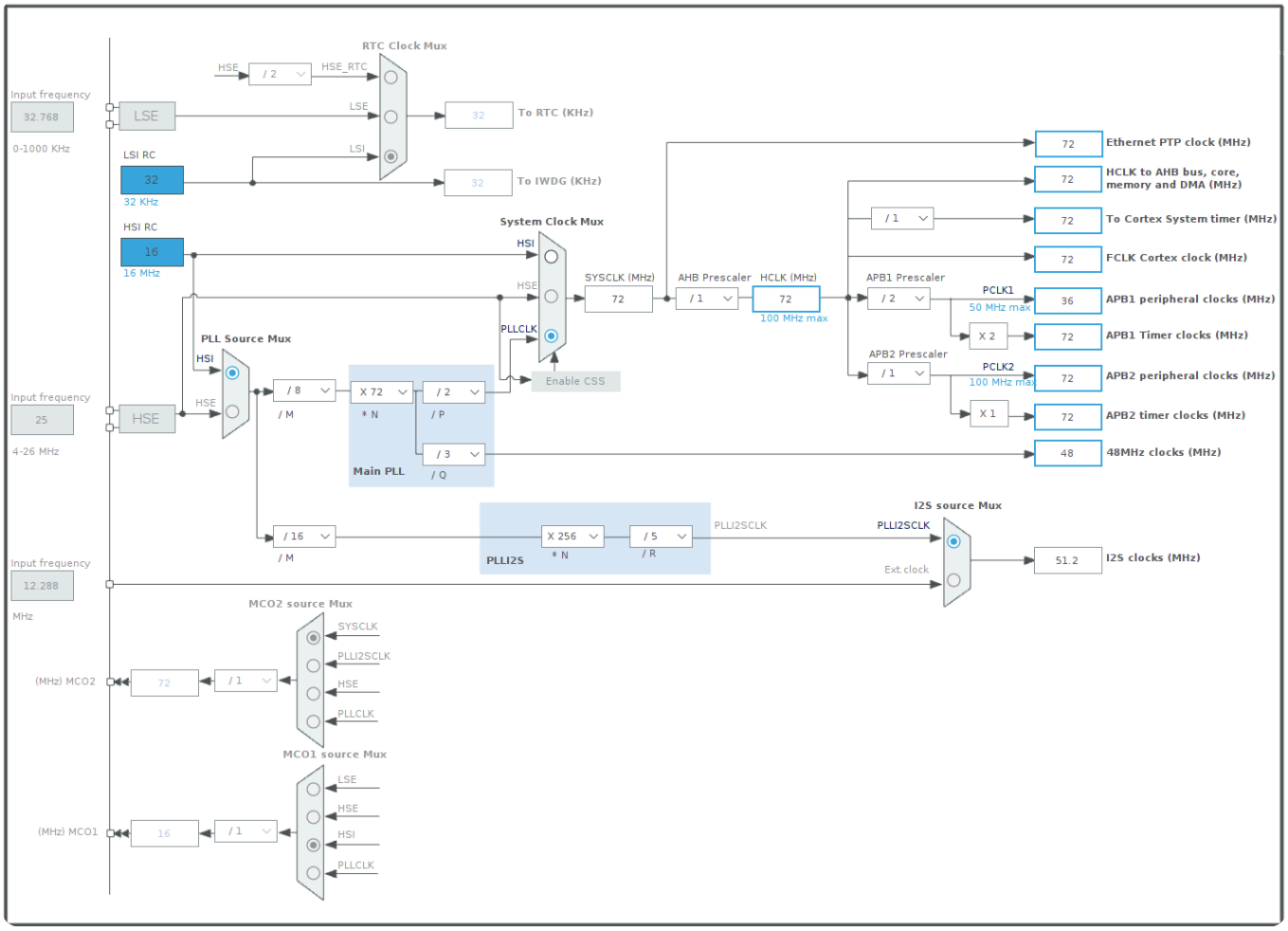
Valami hiba csúszott a hang lejátszásába. Melyik sorban van a hiba?

```c
#include <stdbool.h>

#include "cmsis_os.h"
#include "stm32f4xx_hal.h"

extern I2S_HandleTypeDef hi2s2;

// 20ms audio buffer 48kHz audio esetén
static uint8_t audioBuffer[3840];

#define TAS5727_REGISTER_SYSTEM_CONTROL_2 0x05

static void enableOutput(bool enable) {
    if (enable) {
        TAS5727WriteRegister(TAS5727_REGISTER_SYSTEM_CONTROL_2, 0x00);
    } else {
        TAS5727WriteRegister(TAS5727_REGISTER_SYSTEM_CONTROL_2, 0x40);
    }
    osDelay(3);
}

void TAS5727Play() {
    memset(audioBuffer, 0, sizeof(audioBuffer));
    HAL_I2S_Transmit_DMA(&hi2s2, (uint16_t*)audioBuffer, sizeof(audioBuffer));

    enableOutput(true);
}

void TAS5727Pause(bool pause) {
    if (pause) {
        HAL_I2S_DMAPause(&hi2s2);
    } else {
        HAL_I2S_DMAResume(&hi2s2);
    }
}

void TAS5727Stop() {
    enableOutput(false);

    HAL_I2S_DMAStop(&hi2s2);
}
```

Válaszok

4. feladat 12 pont

Mekk mester szeretné átkonfigurálni futás közben az I2S PLL-t hogy támogatni tudja a 48kHz és 44kHz formátumú audiót is. A processzor órajel beállításait az alábbi ábrán láthatod.



Ügyelj arra, hogy betartsd a PLL megengedett tartományait:

- PLLI2SN >= 100Mhz és PLLI2SN <= 512Mhz

- PLLI2SR <= 216Mhz

- I2S <= 192Mhz

Válaszd ki a helyes megoldásokat! Maximum 1%-os órajel eltérés megengedett!

Válaszok

```
static void SetSampleRate44k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_44K;
    clkSettings.PLLI2S.PLLI2SN = 282;
    clkSettings.PLLI2S.PLLI2SM = 16;
    clkSettings.PLLI2S.PLLI2SR = 5;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}

static void SetSampleRate48k() {
    RCC_PeriphCLKInitTypeDef clkSettings;
```

```
    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_48K;
    clkSettings.PLLI2S.PLLI2SN = 258;
    clkSettings.PLLI2S.PLLI2SM = 16;
    clkSettings.PLLI2S.PLLI2SR = 3;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}
```

```
static void SetSampleRate44k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_44K;
    clkSettings.PLLI2S.PLLI2SN = 282;
    clkSettings.PLLI2S.PLLI2SM = 16;
    clkSettings.PLLI2S.PLLI2SR = 3;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}

static void SetSampleRate48k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_48K;
    clkSettings.PLLI2S.PLLI2SN = 282;
    clkSettings.PLLI2S.PLLI2SM = 16;
    clkSettings.PLLI2S.PLLI2SR = 3;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}
```

```
static void SetSampleRate44k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_48K;
    clkSettings.PLLI2S.PLLI2SN = 282;
    clkSettings.PLLI2S.PLLI2SM = 8;
    clkSettings.PLLI2S.PLLI2SR = 5;
```

```
    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}

static void SetSampleRate48k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_48K;
    clkSettings.PLLI2S.PLLI2SN = 258;
    clkSettings.PLLI2S.PLLI2SM = 8;
    clkSettings.PLLI2S.PLLI2SR = 3;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}
```



```
static void SetSampleRate44k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_44K;
    clkSettings.PLLI2S.PLLI2SN = 141;
    clkSettings.PLLI2S.PLLI2SM = 8;
    clkSettings.PLLI2S.PLLI2SR = 5;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}

static void SetSampleRate48k() {
    RCC_PeriphCLKInitTypeDef clkSettings;

    hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_48K;
    clkSettings.PLLI2S.PLLI2SN = 148;
    clkSettings.PLLI2S.PLLI2SM = 8;
    clkSettings.PLLI2S.PLLI2SR = 3;

    clkSettings.PeriphClockSelection = RCC_PERIPHCLK_I2S;
    HAL_RCCEx_PeriphCLKConfig(&clkSettings);
    HAL_I2S_Init(&hi2s2);
}
```

Megoldások beküldése