BEÁGYAZOTT RENDSZEREK (C)

1. 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:

MINDEN kérdésre van helyes válasz.

Olyan kérdés NINCS, amire az összes válasz helyes, ha mégis az összes választ bejelölöd, arra a feladatra automatikusan 0 pont jár.

Több válaszlehetőség esetén a helytelen válasz megjelölése mínusz pontot ér.

A radio button-os kérdésekre egy helyes válasz van.

Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN beküldi azt az addig megjelölt válaszokkal.

Azokat a feladatlapokat, amelyekhez csatolmány tartozik, javasoljuk NEM mobilon elindítani, erre az érintett feladatlapok előtt külön felhívjuk a figyelmet.

Az adatbekérős feladatokra NEM jár részpontszám, csak a feleletválasztósakra.

Helyezéseket a 4. forduló után mutatunk, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.

Ha egyszerre több böngészőből, több ablakban vagy több eszközről megnyitod ugyanazt a feladatlapot, nem tudjuk vállalni az adatmentéssel kapcsolatban esetlegesen felmerülő anomáliákért a felelősséget!

A ChatGPT használata nem tiltott, de az arra való hivatkozással észrevételt NEM fogadunk el!

A feltűnően rövid idő alatt megoldott feladatlapok kizárást vonnak maguk után, bármilyen más gyanús esetben fenntartjuk a jogot a forduló érvénytelenítésére!

Jó versenyzést kívánunk!

A kategória feladatait javasoljuk NEM mobilon / tableten megoldani!

Mekk Mester beágyazott fejlesztőként dolgozik, és sokféle kihívással kell szembenéznie a projektek kapcsán. A szűk határidők miatt könnyű hibát véteni, segíts hát megtalálni őket, de vigyázz, mert az idő szorít!

Az első projekt egy audió lejátszó lesz, amelynek megvalósításához egy STM32F411 mikrovezérlőt és a TAS5727 audio IC-t fogjuk használni.

Az első fordulóban az audio IC regisztereinek a beállításait I2C-n keresztül a mikrovezérlő I2C1-es perifériájának a használatával szeretnénk elvégezni.

Adatlapok

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

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

1. feladat 6 pont

Jelöld meg a helyes válaszokat!

Válaszok

- [ ] Az I2C alkalmas egymástól nagyobb távolságokra lévő eszközök vezetékes kommunikációjára.
- [ ] Az I2C széles körben elterjedt a processzor és audio IC-k közötti digitális audió átvitelre.
- [ ] Az I2C egy kétirányú szinkron adatátviteli protokoll.
- [ ] Az I2C nem alkalmas full-duplex üzemmódra.
- [ ] Az I2C eszközök IO portjai open-drain vagy open-collector módban üzemelnek.
- [ ] Az I2C eszközök sebessége nem térhet el a szabványban megadott értékektől(100kHz/400kHz)

2. feladat 12 pont

Mekk Mester szeretne az audio IC regisztereinek beállításaihoz segítő függvényeket írni.

Segíts neki kiválasztani a helyes megoldást (feltételezve hogy az I2C beállítások a cubeMX-el helyesen el lettek végezve).

Válasz

```
#include <stdint.h>

#include "Log.h"
#include "stm32f4xx_hal.h"

#define TAS5727_DEVICE_ADDRESS 0x54
#define LOG_Tag "TAS5727"

extern I2C_HandleTypeDef hi2c1;

static void TAS5727WriteRegister(uint8_t registerAddress, uint8_t registerValu
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727WriteRegisterWord(uint8_t registerAddress, uint16_t registe
    HAL_StatusTypeDef status;
    uint8_t data[2];

    data[0] = (uint8_t)(registerValue >> 8);
    data[1] = (uint8_t)(registerValue & 0x00FF);

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727WriteRegisterDWord(uint8_t registerAddress, uint32_t regist
    HAL_StatusTypeDef status;
    uint8_t data[4];

    data[0] = (uint8_t)(registerValue >> 24);
```

```
    data[1] = (uint8_t)(registerValue >> 16);
    data[2] = (uint8_t)(registerValue >> 8);
    data[3] = (uint8_t)(registerValue & 0x00FF);

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727ReadRegister(uint8_t registerAddress, uint8_t* registerValu
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Read(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress,
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Read register failed, status: %d", status);
    }
}

static void TAS5727ReadRegisterWord(uint8_t registerAddress, uint16_t* registe
    HAL_StatusTypeDef status;
    uint8_t data[2];

    status = HAL_I2C_Mem_Read(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress,
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Read register failed, status: %d", status);
    }

    *registerValue = ((((uint16_t)data[0]) << 8) | data[1]);
}
```

```
#include <stdint.h>

#include "Log.h"
#include "stm32f4xx_hal.h"

#define TAS5727_DEVICE_ADDRESS 0x54
#define LOG_Tag "TAS5727"

extern I2C_HandleTypeDef hi2c1;

static void TAS5727WriteRegister(uint8_t registerAddress, uint8_t registerValu
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
```

```
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727WriteRegisterWord(uint8_t registerAddress, uint16_t registe
    HAL_StatusTypeDef status;
    uint8_t data[2];

    data[0] = (uint8_t)(registerValue & 0x00FF);
    data[1] = (uint8_t)(registerValue >> 8);

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727WriteRegisterDWord(uint8_t registerAddress, uint32_t regist
    HAL_StatusTypeDef status;
    uint8_t data[4];

    data[0] = (uint8_t)(registerValue & 0x00FF);
    data[1] = (uint8_t)(registerValue >> 8);
    data[2] = (uint8_t)(registerValue >> 16);
    data[3] = (uint8_t)(registerValue >> 24);

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727ReadRegister(uint8_t registerAddress, uint8_t* registerValu
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Read(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress,
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Read register failed, status: %d", status);
    }
}

static void TAS5727ReadRegisterWord(uint8_t registerAddress, uint16_t* registe
    HAL_StatusTypeDef status;
    uint8_t data[2];

    status = HAL_I2C_Mem_Read(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress,
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Read register failed, status: %d", status);
```

```
    }

    *registerValue = ((((uint16_t)data[1]) << 8) | data[0]);
}
```



```
#include <stdint.h>

#include "Log.h"
#include "stm32f4xx_hal.h"

#define TAS5727_DEVICE_ADDRESS 0x54
#define LOG_Tag "TAS5727"

extern I2C_HandleTypeDef hi2c1;

static void TAS5727WriteRegister(uint8_t registerAddress, uint8_t registerValu
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727WriteRegisterWord(uint8_t registerAddress, uint16_t registe
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727WriteRegisterDWord(uint8_t registerAddress, uint32_t regist
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Write(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Write register failed, status: %d", status);
    }
}

static void TAS5727ReadRegister(uint8_t registerAddress, uint8_t* registerValu
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Read(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress,
```

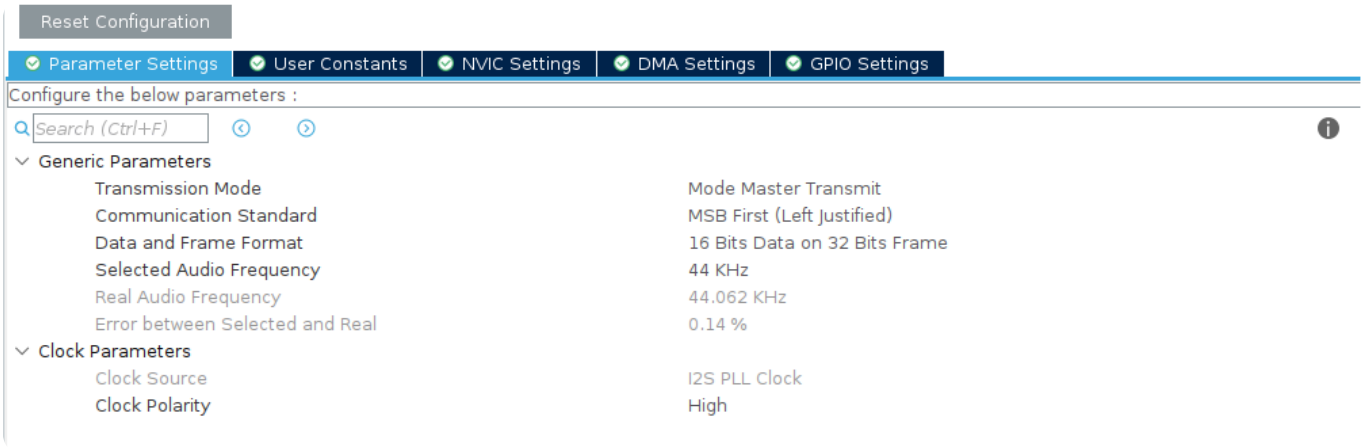
```
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Read register failed, status: %d", status);
    }
}

static void TAS5727ReadRegisterWord(uint8_t registerAddress, uint16_t* registe
    HAL_StatusTypeDef status;

    status = HAL_I2C_Mem_Read(&hi2c1, TAS5727_DEVICE_ADDRESS, registerAddress,
    if (status != HAL_OK) {
        LOG_E(LOG_Tag, "Read register failed, status: %d", status);
    }
}
```

3. feladat 6 pont

Az alábbi ábrán láthatod a mikrovezérlő I2S2 mód beállításait. Válaszd ki hozzá a megfelelő TAS5727 inicializálási rutint (feltételezve hogy a GPIO inicializálások megfelelően el lettek végezve a cubeMX segítségével).



Válasz

```
#include "Log.h"
#include "cmsis_os.h"
#include "TAS5727.h"
#include "stm32f4xx_hal.h"

extern I2C_HandleTypeDef hi2c1;

#define LOG_Tag "TAS5727"

#define AMP_RST_GPIO_Port               GPIOB
#define AMP_RST_Pin G                   PIO_PIN_8
```

```
#define AMP_SHUTDWN_Pin                 GPIO_PIN_9
#define AMP_SHUTDWN_GPIO_Port   GPIOB

#define TAS5727_REGISTER_CLOCK_CONTROL                  0x00
#define TAS5727_REGISTER_DEVICE_ID                              0x01
#define TAS5727_REGISTER_ERROR_STATUS                   0x02
#define TAS5727_REGISTER_SYSTEM_CONTROL_1               0x03
#define TAS5727_REGISTER_SERIAL_DATA_INTERFACE  0x04
#define TAS5727_REGISTER_SYSTEM_CONTROL_2               0x05
#define TAS5727_REGISTER_SOFT_MUTE                              0x06
#define TAS5727_REGISTER_MASTER_VOLUME                  0x07
#define TAS5727_REGISTER_CHANNEL1_VOL                   0x08
#define TAS5727_REGISTER_CHANNEL2_VOL                   0x09
#define TAS5727_REGISTER_CHANNEL3_VOL                   0x0A
#define TAS5727_REGISTER_VOLUME_CONFIGURATION   0x0E
#define TAS5727_REGISTER_PWM_SHUTDOWN_GROUP     0x19
#define TAS5727_REGISTER_OSCILLATOR_TRIM                0x1B
#define TAS5727_REGISTER_PWM_OUTPUT_MUX                 0x25

void TAS5727Init() {
    uint8_t readed;

    // Init sequence
    HAL_GPIO_WritePin(AMP_RST_GPIO_Port, AMP_RST_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(AMP_SHUTDWN_GPIO_Port, AMP_SHUTDWN_Pin, GPIO_PIN_RESET);
    osDelay(100);
    HAL_GPIO_WritePin(AMP_SHUTDWN_GPIO_Port, AMP_SHUTDWN_Pin, GPIO_PIN_SET);
    osDelay(1);
    HAL_GPIO_WritePin(AMP_RST_GPIO_Port, AMP_RST_Pin, GPIO_PIN_SET);
    osDelay(15);

    TAS5727ReadRegister(TAS5727_REGISTER_DEVICE_ID, &readed);
    LOG_I(LOG_Tag, "Device ID: %d", readed);

    TAS5727WriteRegister(TAS5727_REGISTER_OSCILLATOR_TRIM, 0x00);
    osDelay(55);

    TAS5727WriteRegister(TAS5727_REGISTER_SERIAL_DATA_INTERFACE, 0x00);

    TAS5727ReadRegister(TAS5727_REGISTER_ERROR_STATUS, &readed);
    LOG_I(LOG_Tag, "Error status: %d", readed);

    osDelay(500);
}
```

```c
#include "Log.h"
#include "cmsis_os.h"
#include "TAS5727.h"
#include "stm32f4xx_hal.h"

extern I2C_HandleTypeDef hi2c1;

#define LOG_Tag "TAS5727"

#define AMP_RST_GPIO_Port               GPIOB
#define AMP_RST_Pin G                   PIO_PIN_8
#define AMP_SHUTDWN_Pin                 GPIO_PIN_9
#define AMP_SHUTDWN_GPIO_Port   GPIOB

#define TAS5727_REGISTER_CLOCK_CONTROL                  0x00
#define TAS5727_REGISTER_DEVICE_ID                              0x01
#define TAS5727_REGISTER_ERROR_STATUS                   0x02
#define TAS5727_REGISTER_SYSTEM_CONTROL_1               0x03
#define TAS5727_REGISTER_SERIAL_DATA_INTERFACE  0x04
#define TAS5727_REGISTER_SYSTEM_CONTROL_2               0x05
#define TAS5727_REGISTER_SOFT_MUTE                              0x06
#define TAS5727_REGISTER_MASTER_VOLUME                  0x07
#define TAS5727_REGISTER_CHANNEL1_VOL                   0x08
#define TAS5727_REGISTER_CHANNEL2_VOL                   0x09
#define TAS5727_REGISTER_CHANNEL3_VOL                   0x0A
#define TAS5727_REGISTER_VOLUME_CONFIGURATION   0x0E
#define TAS5727_REGISTER_PWM_SHUTDOWN_GROUP     0x19
#define TAS5727_REGISTER_OSCILLATOR_TRIM                0x1B
#define TAS5727_REGISTER_PWM_OUTPUT_MUX                 0x25

void TAS5727Init() {
    uint8_t readed;

    // Init sequence
    HAL_GPIO_WritePin(AMP_RST_GPIO_Port, AMP_RST_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(AMP_SHUTDWN_GPIO_Port, AMP_SHUTDWN_Pin, GPIO_PIN_RESET);
    osDelay(100);
    HAL_GPIO_WritePin(AMP_SHUTDWN_GPIO_Port, AMP_SHUTDWN_Pin, GPIO_PIN_SET);
    osDelay(1);
    HAL_GPIO_WritePin(AMP_RST_GPIO_Port, AMP_RST_Pin, GPIO_PIN_SET);
    osDelay(15);

    TAS5727ReadRegister(TAS5727_REGISTER_DEVICE_ID, &readed);
    LOG_I(LOG_Tag, "Device ID: %d", readed);

    TAS5727WriteRegister(TAS5727_REGISTER_OSCILLATOR_TRIM, 0x00);
    osDelay(55);
```

```
    TAS5727WriteRegister(TAS5727_REGISTER_SERIAL_DATA_INTERFACE, 0x06);

    TAS5727ReadRegister(TAS5727_REGISTER_ERROR_STATUS, &readed);
    LOG_I(LOG_Tag, "Error status: %d", readed);

    osDelay(500);
}
```



```
#include "Log.h"
#include "cmsis_os.h"
#include "TAS5727.h"
#include "stm32f4xx_hal.h"

extern I2C_HandleTypeDef hi2c1;

#define LOG_Tag "TAS5727"

#define AMP_RST_GPIO_Port               GPIOB
#define AMP_RST_Pin G                   PIO_PIN_8
#define AMP_SHUTDWN_Pin                 GPIO_PIN_9
#define AMP_SHUTDWN_GPIO_Port   GPIOB

#define TAS5727_REGISTER_CLOCK_CONTROL                  0x00
#define TAS5727_REGISTER_DEVICE_ID                              0x01
#define TAS5727_REGISTER_ERROR_STATUS                   0x02
#define TAS5727_REGISTER_SYSTEM_CONTROL_1               0x03
#define TAS5727_REGISTER_SERIAL_DATA_INTERFACE  0x04
#define TAS5727_REGISTER_SYSTEM_CONTROL_2               0x05
#define TAS5727_REGISTER_SOFT_MUTE                              0x06
#define TAS5727_REGISTER_MASTER_VOLUME                  0x07
#define TAS5727_REGISTER_CHANNEL1_VOL                   0x08
#define TAS5727_REGISTER_CHANNEL2_VOL                   0x09
#define TAS5727_REGISTER_CHANNEL3_VOL                   0x0A
#define TAS5727_REGISTER_VOLUME_CONFIGURATION   0x0E
#define TAS5727_REGISTER_PWM_SHUTDOWN_GROUP     0x19
#define TAS5727_REGISTER_OSCILLATOR_TRIM                0x1B
#define TAS5727_REGISTER_PWM_OUTPUT_MUX                 0x25

void TAS5727Init() {
    uint8_t readed;

    HAL_GPIO_WritePin(AMP_RST_GPIO_Port, AMP_RST_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(AMP_SHUTDWN_GPIO_Port, AMP_SHUTDWN_Pin, GPIO_PIN_RESET);
    osDelay(100);
    HAL_GPIO_WritePin(AMP_SHUTDWN_GPIO_Port, AMP_SHUTDWN_Pin, GPIO_PIN_SET);
```

```
    osDelay(1);
    HAL_GPIO_WritePin(AMP_RST_GPIO_Port, AMP_RST_Pin, GPIO_PIN_SET);
    osDelay(15);

    TAS5727WriteRegister(TAS5727_REGISTER_OSCILLATOR_TRIM, 0x00);
    osDelay(55);

    TAS5727WriteRegister(TAS5727_REGISTER_SERIAL_DATA_INTERFACE, 0x03);

    TAS5727ReadRegister(TAS5727_REGISTER_ERROR_STATUS, &readed);
    LOG_I(LOG_Tag, "Error status: %d", readed);

    osDelay(500);
}
```

Megoldások beküldése