TP de Synthèse – Autoradio

Note (1) Ce TP est évalué. Appelez le prof dès que quelque chose fonctionne.

Note (2) Votre projet sera dans un projet github à partager avec vos profs. Le compte-rendu sera dans le fichier `README.md`.

Note (3) En cas de doute, appelez votre prof **avant** de faire une connerie !

1 Démarrage

1. Créez un projet pour la carte `NUCLEO_L476RG`. Initialisez les périphériques avec leur mode par défaut, mais **n'activez pas la BSP**.
2. Testez la LED LD2.
3. Testez l'`USART2` connecté à la `STLink` interne.
4. Débrouillez-vous pour que la fonction `printf` fonctionne.
5. Activez FreeRTOS en mode `CMSIS V1`.
6. Faites fonctionner le shell :
   (a) Dans une tâche,
   (b) En mode interruption,
   (c) Avec un driver sous forme de structure.

Remarque Vous pouvez vous aider des codes disponibles sur ce projet github : `https://github.com/lfiack/rtos_td_shell`

2 Le GPIO Expander et le VU-Metre

2.1 Configuration

1. Quelle est la référence du GPIO Expander ? Vous aurez besoin de sa data-sheet, téléchargez-la.
2. Sur le STM32, quel SPI est utilisé ?
3. Quels sont les paramètres à configurer dans `STM32CubeIDE` ?
4. Configurez-les.

2.2 Tests

1. Faites clignoter une ou plusieurs LED.
2. Pour toutes les tester, vous pouvez faire un chenillard (par exemple).

2.3 Driver

1. Écrivez un driver pour piloter les LED. Utilisez une structure.
2. Écrivez une fonction shell permettant d'allumer ou d'éteindre n'importe quelle LED.

3 Le CODEC Audio SGTL5000

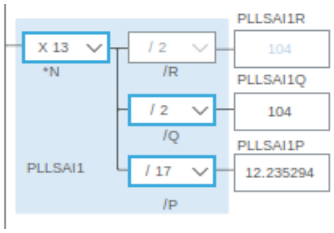
3.1 Configuration préalables

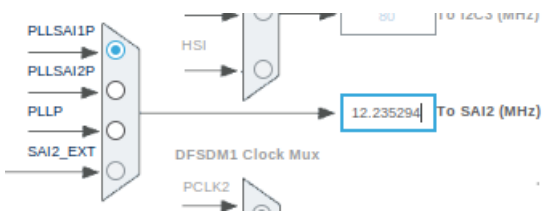
Le CODEC a besoin de deux protocoles de communication :
— L'I2C pour la configuration,
— L'I2S pour le transfert des échantillons audio.
Les configurations suivantes sont à faire sur le logiciel STM32CubeIDE dans la partie graphique CubeMX. Le protocole I2S est géré par le périphérique SAI (*Serial Audio Interface*).

1. Quelles pins sont utilisées pour l'I2C ? À quel I2C cela correspond dans le STM32 ?
2. Activez l'I2C correspondant, laissez la configuration par défaut.
3. Configurez le SAI2 :
   — SAI A : Master with Master Clock Out,
   — Cochez I2S/PCM protocol,
   — SAI B : Synchronous Slave,
   — Cochez I2S/PCM protocol.
4. Si nécessaire, déplacez les signaux sur les bonnes broches. Vous pouvez déplacer une broche avec un `[Ctrl+Clic Gauche]`. Les signaux du SAI doivent être connectés au broches suivantes :

   — **PB12** : SAI2_FS_A
   — **PB13** : SAI2_SCK_A
   — **PB14** : SAI2_MCLK_A
   — **PB15** : SAI2_SD_A
   — **PC12** : SAI2_SD_B

5. Dans l'onglet `Clock Configuration`, configurez PLLSAI1 pour obtenir la fréquence To SAI2 à 12.235294 MHz.





6. Configurez les blocs `SAI A` et `SAI B` de la manière suivante :

SAI A	
Synchronization Inputs	Asynchronous
Basic Parameters	
Audio Mode	Master Transmit
Output Mode	Stereo
Companding Mode	No companding mode
SAI SD Line Output Mode	Driven
Protocol Parameters	
Protocol	I2S Standard
Data Size	16 Bits
Number of Slots (only Ev...	2
Clock Parameters	
Master Clock Divider	Enabled
Audio Frequency	48 KHz
Real Audio Frequency	47.794 KHz
Error between Selected	-0.42 %
Advanced Parameters	
Fifo Threshold	Empty
Output Drive	Disabled
SAI B	
Synchronization Inputs	Synchronous with other block of same SAI
Basic Parameters	
Audio Mode	Slave Receive
Output Mode	Stereo
Companding Mode	No companding mode
SAI SD Line Output Mode	Driven
Protocol Parameters	
Protocol	I2S Standard
Data Size	16 Bits
Number of Slots (only Even Val...	2
Clock Parameters	
Real Audio Frequency	0
Error between Selected	0
Advanced Parameters	
Fifo Threshold	Empty
Output Drive	Disabled

7. Activez les interruptions.
8. Configurez le DMA pour le SAI A et le SAI B. Activez le mode circulaire.
9. Avant de passer à la suite, il est nécessaire d'activer l'horloge MCLK pour que le CODEC fonctionne. Pour cela, dans la fonction `main()`, **après les initialisations**, ajoutez la ligne suivante :

```
__HAL_SAI_ENABLE(&hsai_BlockA2);
```

Note Sans cette ligne, l'I2C ne fonctionne pas, parce que le CODEC ne reçoit pas d'horloge !

3.2 Configuration du CODEC par l'I2C

1. À l'aide d'un oscilloscope, vérifiez la présence d'une horloge sur le signal `MCLK`.
2. À l'aide de la fonction `HAL_I2C_Mem_Read()`, récupérez la valeur du registre `CHIP_ID` (addresse `0x0000`). L'adresse I2C du CODEC est `0x14`.
3. Observez les trames I2C à l'oscilloscope.
4. Montrez à l'enseignant.
5. Cherchez dans la documentation du SGTL5000 la valeur à assigner aux registres suivants :

```
— CHIP_ANA_POWER
— CHIP_LINREG_CTRL
— CHIP_REF_CTRL
— CHIP_LINE_OUT_CTRL
— CHIP_SHORT_CTRL
— CHIP_ANA_CTRL
— CHIP_ANA_POWER
— CHIP_DIG_POWER
— CHIP_LINE_OUT_VOL
— CHIP_CLK_CTRL
— CHIP_I2S_CTRL
— CHIP_ADCDAC_CTRL
— CHIP_DAC_VOL
```

6. Créez une paire de fichier `sgtl5000.c` / `sgtl5000.h`
7. Dans le fichier `sgtl5000.c`, créez une fonction d'initialisation.
8. Dans cette fonction, écrivez le code permettant de configurer ces registres.

3.3 Signaux I2S

1. Démarrez la réception et la transmission sur l'I2S avec le SAI :

```
HAL_SAI_Receive_DMA()
HAL_SAI_Transmit_DMA()
```

2. Observez à l'oscilloscope les différents signaux d'horloge.
3. Montrez à l'enseignant.

3.4 Génération de signal audio

1. Générez un signal triangulaire.
2. Vérifier à l'oscilloscope, montrez à l'enseignant.

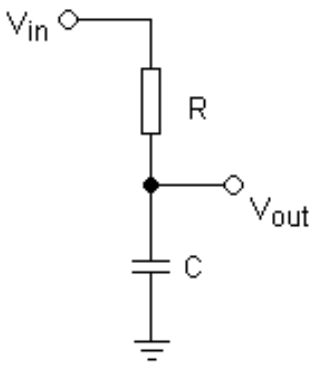
3.5 Bypass numérique

1. Écrivez le code permettant de lire les échantillons de l'ADC, et de les écrire sur le DAC.
2. Vérifier à l'oscilloscope, montrez à l'enseignant.

4 Visualisation

1. Écrivez le code permettant de visualiser le volume sonore sur les LED. `https://en.wikipedia.org/wiki/VU_meter`
2. Montrez à l'enseignant.

5 Filtre RC



1. Écrivez l'équation différentielle régissant le filtre RC ci-contre. Mettre l'équation sous la forme suivante :

$$V_{in} = X \cdot \frac{dV_{out}(t)}{dt} + Y \cdot V_{out}$$

2. Écrivez l'équation de récurrence correspondante. On peut remplacer $\frac{dV(t)}{dt}$ par $\frac{V[n]-V[n-1]}{T}$. Pour faciliter le codage, on utilisera plutôt l'expression suivante :

$$f_s \cdot (V[n] - V[n-1])$$

Avec f_s la fréquence d'échantillonnage.

Mettre sous la forme suivante :

$$V_{out}[n] = \frac{A \cdot V_{in}[n] + B \cdot V_{out}[n-1]}{D}$$

3. Donnez les expressions de A, B et D. Remplacez RC par $\frac{1}{2 \cdot \pi \cdot f_c}$.
4. Pour une fréquence d'échantillonnage de 48kHz, combien de cycles processeurs disposons-nous pour traiter chaque échantillon ?
5. Créez une paire de fichiers `RCFilter.c` / `RCFilter.h`
6. Créez la structure suivante dans `RCFilter.h` :

```c
typedef struct {
    uint32_t coeff_A;
    uint32_t coeff_B;
    uint32_t coeff_D;
    uint16_t out_prev;
} h_RC_filter_t;
```

7. Écrivez les fonctions suivantes :

```c
// Calcule les coefficients A, B et D
// Et les stocke dans la structure
void RC_filter_init(h_RC_filter_t * h_RC_filter,
        uint16_t cutoff_frequency, uint16_t sampling_frequency);
// Implémente l'équation de récurrence
// Faites attention au type des différentes variables
uint16_t RC_filter_update(h_RC_filter_t * h_RC_filter,
        uint16_t input);
```

8. Ajoutez une fonction au Shell pour modifier la fréquence de coupure.
9. Faites valider par votre enseignant.

6 Programmation d'un effet audio

1. Programmez un effet audio de votre choix :
   (a) Saturation/distortion
   (b) Tremolo
   (c) Filtre analogique
   (d) Delay (court...)
   (e) Chorus/Phaser/Flanger
   (f) Compresseur
   (g) Reverb