I2S

Aperçu

I2S (Inter-IC Sound) est un protocole de communication synchrone série qui est généralement utilisé pour transmettre des données audio entre deux appareils audio numériques. Un bus I2S se compose des lignes suivantes:

- Ligne d'horloge de bits
- Ligne de sélection de canal
- Ligne de données série

Chaque contrôleur I2S possède les fonctionnalités suivantes qui peuvent être configurées à l'aide du pilote I2S:

- Fonctionnement en tant que maître ou esclave du système
- Capable d'agir comme émetteur ou récepteur
- Contrôleur DMA dédié qui permet de diffuser des échantillons de données sans exiger du CPU de copier chaque échantillon de données

Chaque contrôleur peut fonctionner en mode de communication semi-duplex. Ainsi, les deux contrôleurs peuvent être combinés pour établir une communication en duplex intégral. La sortie I2S0 peut être acheminée directement vers les canaux de sortie du convertisseur numérique-analogique (DAC) (GPIO 25 et GPIO 26) pour produire une sortie analogique directe sans impliquer de codecs I2S externes. I2S0 peut également être utilisé pour transmettre des signaux PDM (modulation de densité d'impulsions).

Les périphériques I2S prennent également en charge le mode LCD pour la communication de données sur un bus parallèle, tel qu'utilisé par certains écrans LCD et modules de caméra. Le mode LCD a les modes de fonctionnement suivants:

- Mode de transmission du maître LCD
- Mode de réception esclave de la caméra
- Mode ADC / DAC

Remarque

Pour les applications d'horloge de haute précision, utilisez la source d'horloge APLL_CLK, qui a la gamme de fréquences de 16 ~ 128 MHz. Vous pouvez activer la source d'horloge APLL_CLK en définissant: cppi2s_con fi g_t :: use_apll sur VRAI.

Si: cppi2s_con fi g_t :: use_apll = VRAI et: cppi2s_con fi g_t :: fi xed_mclk > 0, alors la fréquence de sortie de l'horloge maître pour I2S sera égale à la valeur de: cppi2s_conf g_t :: fi xed_mclk, ce qui signifie que la fréquence mclk est fournie par l'utilisateur, au lieu d'être calculée par le pilote.

La fréquence d'horloge de la ligne de sélection de mot, appelée ici fréquence d'horloge audio gauche-droite (LRCK), est toujours le diviseur de la fréquence de sortie de l'horloge maître et pour laquelle ce qui suit est toujours vrai: 0 <MCLK / LRCK / canaux / bits_par_échantillon <64.

Aperçu fonctionnel

Installation du pilote

Installez le pilote I2S en appelant la fonction: cpp: func'i2s_driver_install 'et en passant les arguments suivants:

- Numéro de port
- La structure: cppi2s_con fi g_t avec des paramètres de communication dé fi nis
- Exemple de configuration et taille de la file

d'attente d'événements:

```
statique const int i2s_num = 0 ; // numéro de port i2s

statique const i2s_config_t i2s_config = {
    . mode = I2S_MODE_MASTER | I2S_MODE_TX,
    . sample_rate = 44100 ,
    . bits_per_sample = 16 ,
    . channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    . communication_format = I2S_COMM_FORMAT_STAND_I2S,
    . intr_alloc_flags = 0 , // priorité d'interruption par défaut
    . dma_buf_count = 8 ,
    . dma_buf_len = 64 ,
    . use_apll = false};


i2s_driver_install (I2S_NUM, & i2s_config, 0 , NUL);
```

Définition des broches de communication

Une fois le pilote installé, con fi gurez les broches GPIO physiques vers lesquelles les signaux seront acheminés. Pour cela, appelez la fonction: cpp: func'i2s_set_pin 'et passez-lui les arguments suivants:

- Numéro de port
- La structure: cppi2s_pin_con fi g_t dé fi nissant les numéros de broches GPIO vers lesquels le pilote doit acheminer les signaux BCK, WS, DATA out et DATA.
  Si vous souhaitez conserver un code PIN actuellement attribué pour un signal spéci fi que, ou si ce signal n'est pas utilisé, alors passez la macro: cI2S_PIN_NO_CHANGE. Voir l'exemple ci-dessous.

```
statique const i2s_pin_config_t pin_config = {
    . bck_io_num = 26 ,
    . ws_io_num = 25 ,
    . data_out_num = 22 ,
    . data_in_num = I2S_PIN_NO_CHANGE};


i2s_set_pin (i2s_num, & pin_config);
```

Exécution de la communication I2S

Pour effectuer une transmission:

- Préparez les données pour l'envoi
- Appelez la fonction: cppi2s_write et transmettez-lui l'adresse du tampon de données et la longueur des données

La fonction écrira les données dans le tampon I2S DMA Tx, puis les données seront transmises automatiquement.

```
i2s_write (I2S_NUM, samples_data, ((bits +8 ) / 16 ) * SAMPLE_PER_CYCLE * 4 , & i2s_bytes_write, 100 );
```

Pour récupérer les données reçues, utilisez la fonction: cppi2s_read. Il récupérera les données du tampon I2S DMA Rx, une fois les données reçues par le contrôleur I2S. Vous pouvez arrêter temporairement le pilote I2S en appelant la fonction: cppi2s_stop, qui désactivera les unités I2S Tx / Rx jusqu'à ce que la fonction: cppi2s_start soit appelée. Si la fonction: cpp: func'i2s_driver_install 'est utilisée, le pilote démarre automatiquement, éliminant ainsi le besoin d'appeler: cppi2s_start.

Suppression du pilote

Si la communication établie n'est plus requise, le pilote peut être supprimé pour libérer les ressources allouées en appelant: cppi2s_driver_uninstall.

Exemple d'application

Un exemple de code pour le pilote I2S peut être trouvé dans le répertoire périphériques / i2s. De plus, il existe deux courts exemples de configuration pour le pilote I2S.

Configuration I2S

```
#inclure "driver / i2s.h"
#inclure "freertos / queue.h"

statique const int i2s_num = 0 ; // numéro de port i2s

statique const i2s_config_t i2s_config = {
        . mode = I2S_MODE_MASTER | I2S_MODE_TX,
        . sample_rate = 44100 ,
        . bits_per_sample = 16 ,
        . channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
        . communication_format = I2S_COMM_FORMAT_STAND_I2S,
        . intr_alloc_flags = 0 , // priorité d'interruption par défaut
        . dma_buf_count = 8 ,
        . dma_buf_len = 64 ,
        . use_apll = false};


statique const i2s_pin_config_t pin_config = {
        . bck_io_num = 26 ,
        . ws_io_num = 25 ,
        . data_out_num = 22 ,
        . data_in_num = I2S_PIN_NO_CHANGE};


. . .

        i2s_driver_install (i2s_num, & i2s_config, 0 , NULL); // installer et démarrer le pilote i2s

        i2s_set_pin (i2s_num, & pin_config);

        i2s_set_sample_rates (i2s_num, 22050 ); // définir des taux d'échantillonnage
```

```
    i2s_driver_uninstall (i2s_num); // arrêter et détruire le pilote i2s
```

Configuration de I2S pour utiliser un DAC interne pour la sortie analogique

```
#inclure "driver / i2s.h"
#inclure "freertos / queue.h"

statique const int i2s_num = 0 ; // numéro de port i2s

statique const i2s_config_t i2s_config = {
    . mode = I2S_MODE_MASTER | I2S_MODE_TX | I2S_MODE_DAC_BUILT_IN,
    . sample_rate = 44100 ,
    . bits_per_sample = 16 , / * le module DAC ne prendra que les 8 bits de MSB * /
    . channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    . intr_alloc_flags = 0 , // priorité d'interruption par défaut
    . dma_buf_count = 8 ,
    . dma_buf_len = 64 ,
    . use_apll = false};


. . .

    i2s_driver_install (i2s_num, & i2s_config, 0 , NUL); // installer et démarrer le pilote i2s

    i2s_set_pin (i2s_num, NULL); // pour DAC interne, cela activera les deux canaux internes

    // Vous pouvez appeler i2s_set_dac_mode pour définir le mode de sortie DAC intégré. //
    i2s_set_dac_mode (I2S_DAC_CHANNEL_BOTH_EN);

    i2s_set_sample_rates (i2s_num, 22050 ); // définir des taux d'échantillonnage

    i2s_driver_uninstall (i2s_num); // arrêter et détruire le pilote i2s
```

Référence API