Laboratoire de microprocesseurs - ZG6
DMA

I Présentation du sujet

L'objectif de ce laboratoire est d'aborder l'utilisation d'un contrôleur DMA (*Direct Memory Access*) permettant des transferts de données, entre un périphérique et de la mémoire ou de mémoire à mémoire, sans solliciter le processeur.

Dans ce laboratoire, le périphérique qui utilise un canal DMA est une UART. Les caractères que l'UART reçoit depuis un terminal série seront placés dans un tampon mémoire via un canal DMA ; de la même façon, les caractères à émettre seront lus dans un tampon mémoire pour être transmis à l'UART via un canal DMA. Le projet de départ configure l'USART2 pour une communication série au format 8N1, sans interruptions, à 115200 bauds. Le contrôleur DMA1 est utilisé et est en partie configuré dans le projet de départ.

Dans un premier temps (MAIN1), le contrôleur DMA est directement configuré dans le programme principal afin de tester le fonctionnement de l'UART associée au DMA. Dans un deuxième temps (MAIN2), l'API du DMA est utilisée. L'API est fournie complète, seule son utilisation est testée.

II MAIN1 : configuration directe du DMA

II.1 Configuration du DMA

Le contrôleur DMA est configuré une première fois pour transmettre une chaîne caractères au terminal série qui l'affiche. Lorsque la chaîne a été émise, le programme est organisé autour d'une boucle infinie où le contrôleur DMA est, à chaque passage dans la boucle, configuré d'abord pour la réception de 10 caractères, puis ensuite pour l'émission de 10 caractères. Les trois configurations correspondantes sont données, sauf pour ce qui concerne le registre CR.

Compléter le code fourni pour que les 3 configurations soient complètes.

II.2 Test du transfert des caractères

a) Compiler puis télécharger le code. Placer ensuite un point d'arrêt sur la ligne qui démarre le premier transfert DMA (juste avant la boucle d'attente sur CR[0]). Exécuter le code puis observer la valeur des registres qui gèrent la voie 6 du DMA (rappel : **`p /x *_DMA1_Stream6`**). Analyser la valeur des registres, notamment :
   - justifier la valeur de NDTR et PAR.
   - vérifier le contenu de la mémoire à l'adresse donnée par M0AR. La commande GDB suivante peut être utilisée : **`x /64bc _DMA1_Stream6->M0AR`**.

b) Relancer l'exécution du programme et vérifier que le message d'invite s'affiche bien.

c) Saisir des caractères dans le terminal série. Vérifier qu'après le 10ème saisi, les caractères sont affichés.

d) Arrêter l'exécution du programme. Le programme doit s'arrêter sur la ligne d'attente de la réception des 10 caractères (`while(_DMA1_Stream5->CR & 1) ;`).
   - observer la valeur des registres qui gèrent la voie 5 du DMA.
   - **sans relancer l'exécution du programme**, saisir 5 caractères dans le terminal série.
   - observer à nouveau la valeur des registres qui gèrent la voie 5 du DMA.

II.3 Flot de données contrôlé par le périphérique

Dans la configuration proposée, le contrôle du flot de données est dévolu au DMA (*DMA flow controller*) qui connaît le nombre de données à transférer. Le canal DMA peut aussi être contrôlé par le périphérique (*Peripheral flow controller*). Dans ce cas, le nombre de données à transférer n'est pas connu à l'avance.

a) Modifier la configuration du registre CR qui gère la voie 5 du DMA (réception des caractères) pour que le contrôle du flot de données soit dévolu au périphérique.
b) Dans le mode *Peripheral flow controller*, le registre NDTR permet de calculer le nombre de caractères reçus. Modifier la condition d'attente de fin de transfert, en vous aidant de la documentation (***§9.3.15***), pour qu'après 10 caractères reçus, le code remette le bit EN de CR à 0 et passe à la partie émettant les caractères vers le terminal série.
c) Placer un point d'arrêt sur la condition d'attente de fin de transfert (point b) ci-dessus) et observer la valeur de NDTR après la saisie de quelques caractères (moins de 10) dans le terminal série, sans relancer le programme.
d) Ôter le point d'arrêt et relancer l'exécution du programme.

III MAIN2 : utilisation de l'API du DMA

Afin de pouvoir utiliser les DMA plus facilement dans différents projets, une API a été définie. Le programme MAIN2 utilise cette API pour implémenter deux fonctions : ***uart_gets()*** et ***uart_puts()***. Le code est fourni complet à l'exception de l'initialisation des DMA par la fonction ***dma_stream_init()***. Cette fonction doit être appelée avec les paramètres suivants :

- DMA_t* *dma* : identifie le DMA utilisé,
- uint32_t *stream* : numéro de la voie,
- DMAEndPoint_t **src* : description de la source des données,
- DMAEndPoint_t **dest* : description de la destination des données,
- uint32_t *mode* : mode de fonctionnement de la voie (*cf* étiquettes symboliques dans ***dma.h***),
- OnTC *cb* : fonction de rappel (*callback*).

La description des source et destination des données (points de connexion/*EndPoints*) est basée sur un type de donnée comprenant les champs suivants (*cf* ***dma.h***) :

- Peripheral *type* : périphérique utilisé. Voir les valeurs possibles dans ***dma.h***,
- void* *addr0* : adresse où les données peuvent être lues ou écrites (mémoire ou registre),
- void* *addr1* : adresse de la mémoire utilisée dans le mode double tampon (*Double buffer*),
- int *channel* : numéro du canal DMA,
- uint32_t *cfg* : mode de fonctionnement du point de connexion (*cf* étiquettes symboliques dans ***dma.h***).

III.1 Fonction uart_puts() avec DMA

La fonction ***uart_puts()*** transfère des données d'un tampon mémoire, où est stockée une chaîne de caractères, vers l'UART.

a) Compléter l'initialisation du descripteur **ep_buf** décrivant la source des données. Prévoir que le pointeur sur les données à transmettre s'auto-incrémente.
b) Compléter l'initialisation du descripteur **dma1s6_uart2_tx** décrivant la destination des données.
c) Compléter l'appel de la fonction **dma_stream_init()**.

III.2 Fonction uart_gets() avec DMA

La fonction ***uart_gets()*** transfère des données d'un tampon mémoire où est stockée une chaîne de caractères vers l'UART.

a) Compléter l'initialisation du descripteur **ep_buf** décrivant la destination des données. Prévoir que le pointeur sur les données à écrire s'auto-incrémente.
b) Compléter l'initialisation du descripteur **dma1s5_uart2_rx** décrivant la source des données.
c) Compléter l'appel de la fonction **dma_stream_init()**.

III.3 Test

Après exécution de la fonction **dma_stream_init()**, pour chacune des fonctions ***uart_puts()*** et ***uart_gets()***, observer la valeur des registres du DMA.

Vérifier que le programme MAIN2 fonctionne correctement.