Laboratoire de microprocesseurs - n°7 - l'ADC

1. Convertisseur en mode single/polling 1
2. Convertisseur en mode single/irq 1
3. Conversion d'une séquence de canaux en mode continu/irq 2
4. Conversion d'une séquence déclenchée par un timer 2

Le but est de développer une interface permettant d'utiliser le convertisseur analogique numérique (ADC) inclu dans le microcontrôleur et étudié précédemment en TD. Le programme principal utilise une interface pour afficher les valeurs converties via la console série (/dev/ttyACM0, 115200 bauds, 8 bits, pas de parité, 1 bit de stop).

On rappelle les différentes configurations valides vis à vis de l'interface proposée.

	poll single	irq single	irq cont	irq triggered	commentaires
CR2[0]	1	1	1	1	ADC on
CR1[8]	1	1	1	1	convert input selected by SQR[ ] registers
CR2[1]	0	0	1	0	continuous conversion mode
CR1[11]	0	0	0	1	triggered conversion mode
callback	n	y	y	y	should there be a callback?
CR2[10]	1	1	1	1	EOC bit is set after each conversion
CR1[5]	0	1	1	1	generate an IRQ after each conversion

1. Convertisseur en mode single/polling

La fonction `main` permet de tester la fonction `adc_channel_sample`. Cette fonction permet de configurer l'ADC pour convertir 1 canal d'entrée en mode monocoup (*single/polling*). Les interruptions ne sont pas utilisées.

a. Analyser le code de la fonction `main` (défini par le symbole MAIN1). Compléter le fichier `lib/adc.c` avec le code développé en TD.

b. Tester le code avec les commandes '0' et '1'. Vérifier en particulier le contenu des registres `CCR`, `CR1`, `CR2` et `SQR[]`.

c. Les commandes 'b', 'r' et 't' permettent d'obtenir respectivement une image de la tension d'alimentation, de la référence interne et de la température du microcontrôleur. Retrouver, à partir de la valeur fournie par le convertisseur, les indications de la documentation (voir la partie 6.3.23 p. 116 de la datasheet pour la référence de tension).

   Modifier la valeur fournie par l'interrogation du capteur de température pour afficher directement la température (voir le paragraphe 11.9 p. 225 du manuel d'utilisation du microcontrôleur STM32, ainsi que la partie 6.3.21 p. 116 de la datasheet du microcontrôleur).

2. Convertisseur en mode single/irq

On souhaite rajouter la possibilité de faire fonctionner l'ADC sur interruption lorsqu'on appelle la fonction `adc_channel_sample`. Cette fonction permet maintenant d'utiliser l'ADC soit par scrutation, soit sur interruption.

La différence est faite au moment de l'initialisation. Si une callback est définie, le fonctionnement choisi est implicitement sur interruption. Dans ce cas, la fonction `adc_channel_sample` ne fait que la configuration du canal, lance la conversion et sort en renvoyant la valeur 0. Sinon, la fonction attend que la conversion soit terminée avant de renvoyer la valeur convertie.

a. Modifier les fonctions `adc_init` (code d'initialisation) et `adc_channel_sample` du ficher `lib/adc.c` pour tenir compte du paramètre de callback. Compléter la routine d'interruption qui doit récupérer le numéro de canal à partir des registres SQR[ ], appeler la callback en lui passant le numéro de canal et le résultat de la conversion, et désarmer la demande d'interruption.

b. Analyser le code `main` (symbole MAIN2). Noter en particulier la différence de structure du programme par rapport au fonctionnement par scrutation.

c. Tester le fonctionnement. Vérifier en particulier le passage par la routine d'interruption et l'appel de la callback.

3. Conversion d'une séquence de canaux en mode continu/irq

Le convertisseur permet la conversion de séquences de canaux dans un ordre quelconque. La fonction `adc_channel_enable` permet d'ajouter un nouveau canal à la liste des canaux qui seront traités.

On profite de cette partie pour introduire la possibilité d'avoir une conversion en continu. La période d'échantillonnage est fixée par le paramètre `in_sample_time`. Par exemple, si le convertisseur est configuré pour $n = 12$ bits et $in\_sample\_time = 3$, la période d'échantillonnage est

$$T_e = \frac{n + in\_sample\_time}{ADC\_CLK} = 714 \text{ ns}$$

avec $ADC\_CLK = 21$ MHz.

On ne considère qu'un fonctionnement sur interruption.

a. Modifier la fonction `adc_init`, `adc_channel_enable` et la routine d'interruption.

b. Analyser le code du programme principal (symbole MAIN3). Tester le programme.

4. Conversion d'une séquence déclenchée par un timer

L'inconvénient du mode continu, c'est qu'on ne maîtrise pas finement la fréquence d'échantillonnage. Une manière plus souple est d'utiliser un Timer pour, soit périodiquement générer des interruptions qui permettront de lancer la conversion d'une séquence d'entrée, soit de générer des évènements qui seront directement transmis à l'ADC. C'est cette deuxième solution qu'on souhaite mettre en œuvre.

La configuration du mode "triggered" passe par les configurations suivantes au niveau de l'ADC.

bits registre	champ	commentaire
CR2[27:24]	EXTSEL	Sélection de la source d'évènement
CR2[29:28]	EXTEN	Autorisation/polarité de l'évènement
CR1[15:13]	DISCNUM	Nombre de canaux à convertir (copie de SQR[0][23:20])

Au niveau du Timer utilisé pour générer l'évènement "trigger", il faut aussi configurer l'élément interne du timer connecté au signal *trigger output* (TRGO). Ici, on connecte le signal interne '*update*' du timer (déclenché par le dépassement du compteur CNT par rapport au registre d'autoreload ARR).

bits registre Timer	champ	commentaire
CR2[6:4]	MMS	valeur : `0b010` The update event is selected as trigger output (TRGO).

a. Modifier la fonction `adc_init` pour prendre en compte le mode “triggered” (CR1[11]).

b. Modifier la fonction `adc_channel_enable` pour mettre à jour lel champ DISCNUM de CR1 à chaque fois qu’un nouveau canal est rajouté.

c. Compléter la fonction `adc_set_trigger_evt` pour configurer les champs EXTSEL et EXTEN. A noter que les définitions contenues dans le fichier `lib/adc.h` fournissent les valeurs pour ces deux champs. On ne gère que les cas ADC_TRIG_ON_TIM2_TRGO et ADC_TRIG_ON_TIM3_TRGO. La configuration n’est faite que si on est en mode *“triggered”* (bits CR1[11] à 1).

   La fonction réalise aussi la configuration des timers concernés.

d. Analyser le programme principal (symbole MAIN4) et tester les modifications faites. Par défaut, une callback est associée au timer `_TIM3` ce qui configure les interruptions pour le timer à chaque évènement ‘update’ et qui permet de faire changer d’état une led à chaque fois que la routine d’interruption est appelée, ce qui illustre de manière visible la cadence des conversions (pour des cadences faibles). Modifier la période du timer `_TIM3` et visualiser l’effet sur les conversions de l’ADC.

e. Si on met le paramètre de callback du timer à NULL, les interruptions ne sont plus utilisées pour le timer. La période d’échantillonnage est néanmoins fixée par le timer : c’est le signal TRGO du timer qui lance les séquences de conversion. Vérifier que c’est bien le cas.