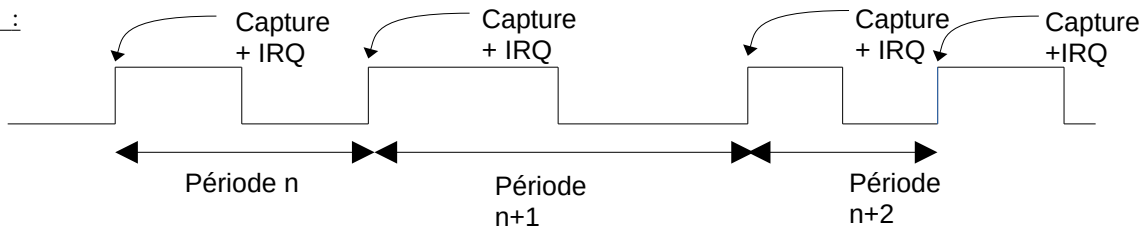
SY41 Final

Durée 1h30. Un feuille de notes autorisée. Pas de calculatrice ou autre appareil.

Il s'agit d'écrire une partie d'application consistant à mesurer des intervalles de temps (période) et d'envoyer la valeur mesurée sur une interface série. L'application principale attend qu'une nouvelle capture soir réalisée (cœur endormi). Dès qu'une capture a eu lieu l'application principale transmet la valeur mesurée en microseconde sur la liaison série contrôlée par l'USART2.

1 Mesure de périodes (8)

L'application utilise le timer TIM3 (timer sur 16 bits) pour effectuer des mesures de périodes. La période du signal est mesurée entre 2 fronts montant d'un signal d'entré.
Le timer est configuré de tel manière qu'une capture soit effectuée dans le registre CCR1 à chaque front montant du signal d'entré (input mode). Le NVIC est aussi déjà considéré configuré pour que les IRQ du TIM3 soient transmises au cœur. Le TIM3 est configuré poure qu'une IRQ soit générée à chaque capture dans CCR1 (drapeau CCR1IF) et à chaque overflow du compteur principal (drapeau UIF). Le compteur est paramétré pour compter à la fréquence de 1MHz avec ARR fixé à la valeur maximale (ARR = 65535). Le timer est actif (compte) et les interruptions sont actives (NVIC et bit N°0 et N°1 du registre DIER du TIM3 à 1).



La fonction utilisateur dont le prototype est donné ci-dessous est appelée pour récupérer une mesure. Cette fonction est une fonction synchrone (bloquante) qui retourne la dernière mesure effectuée (en nombre de ticks du timer). Si une nouvelle mesure est déjà prête, la fonction retourne tout de suite, dans le cas contraire, la fonction attendra l'acquisition d'une nouvelle mesure avec le cœur du SoC endormi. (utilisez l'instruction _WFI() pour endormir le cœur.

Donnez le code commenté de la fonction utilisateur et de la routine de traitement des interruptions associée, dont les prototypes sont donnés ci-dessous. Des variables globales partagées pourront être utilisées (et leur utilité commentée).

- ***uint16_t* mesure_periode(void) *:*** fonction renvoyant la valeur de la dernière période mesurée
- ***void TIM3_IRQHandler(void)*** **:** la routine de traitement des interruption du tim3

2 Envoi de la mesure via l'USART (8)

Pour l'envoi de la donnée mesurée le périphérique USART2 est utilisé. Il est considéré dans cet exercice que le périphérique USART2 est configuré (mode asynchrone, vitesse, pas de parité, 1 bit de stop). Les interruptions seront utilisées pour le transfert de données afin de maintenir la puce endormi au maximum et ainsi économiser l'énergie. Le NVIC est aussi déjà considéré configuré pour que les IRQ de l'USART2 soient transmises au cœur, les interruptions

de l'USART2 sont masquées au niveau de l'interface (bit TXEIE du registre CR1 à 0)

Vous devez ici écrire une fonction utilisateur générique asynchrone d'envoi de données. La fonction utilisateur appelée USART2_send_IT() lance le transfert et revient immédiatement. Les données sont transférées en arrière plan.

Donnez le code commenté de la fonction utilisateur et de la routine de traitement des interruptions associée, dont les prototypes sont donnés ci-dessous. Des variables globales partagées pourront être utilisées (et leur utilité commentée).

- **`void USART2_send_IT(uint8_t *data, uint32_t len)`** : fonction prenant en paramètre l'adresse des données à transmettre (data) et le nombre de donnée 8 bits (len). La fonction est asynchrone, elle revient immédiatement
- **`void USART2_IRQHandle(void)`** : la routine de traitement des interruption de l'USART2

La donnée sur 16 bits est envoyée directement sans mise en forme, poids fort en premier.

3 Questions annexes (4) :

1. la fonction USART2_send_IT étant asynchrone, que se passe-t-il si un transfert est en cours au moment de l'appel par l'application ? Comment géreriez-vous ce cas ?
2. A quoi l'application utilisant la fonction USART2_send_IT doit-elle veillez pour qu'il n'y ait pas de corruption des données lors du transfert ? Comment cela peut-il être géré au niveau de la fonction USART2_send_IT() ?
3. La puce envoi 2 octets, sachant que le baud rate est à 115200 baud (on approximera le baud rate à 100 000 baud pour les calculs), quelle serait la période du signal d'entrée la plus petite mesurable (sans over-capture) en négligeant le temps d'exécution du processeur.

4 Annexes

4.1 Définitions

```
typedef struct
{
  __IO uint32_t CR1;         /*!< TIM control register 1,              Address offset: 0x00 */
  __IO uint32_t CR2;         /*!< TIM control register 2,              Address offset: 0x04 */
  __IO uint32_t SMCR;        /*!< TIM slave mode control register,     Address offset: 0x08 */
  __IO uint32_t DIER;        /*!< TIM DMA/interrupt enable register,   Address offset: 0x0C */
  __IO uint32_t SR;          /*!< TIM status register,                 Address offset: 0x10 */
  __IO uint32_t EGR;         /*!< TIM event generation register,       Address offset: 0x14 */
  __IO uint32_t CCMR1;       /*!< TIM capture/compare mode register 1, Address offset: 0x18 */
  __IO uint32_t CCMR2;       /*!< TIM capture/compare mode register 2, Address offset: 0x1C */
  __IO uint32_t CCER;        /*!< TIM capture/compare enable register, Address offset: 0x20 */
  __IO uint32_t CNT;         /*!< TIM counter register,                Address offset: 0x24 */
  __IO uint32_t PSC;         /*!< TIM prescaler,                       Address offset: 0x28 */
  __IO uint32_t ARR;         /*!< TIM auto-reload register,            Address offset: 0x2C */
  __IO uint32_t RCR;         /*!< TIM repetition counter register,     Address offset: 0x30 */
  __IO uint32_t CCR1;        /*!< TIM capture/compare register 1,      Address offset: 0x34 */
  __IO uint32_t CCR2;        /*!< TIM capture/compare register 2,      Address offset: 0x38 */
  __IO uint32_t CCR3;        /*!< TIM capture/compare register 3,      Address offset: 0x3C */
  __IO uint32_t CCR4;        /*!< TIM capture/compare register 4,      Address offset: 0x40 */
  __IO uint32_t BDTR;        /*!< TIM break and dead-time register,    Address offset: 0x44 */
  __IO uint32_t DCR;         /*!< TIM DMA control register,            Address offset: 0x48 */
  __IO uint32_t DMAR;        /*!< TIM DMA address for full transfer,   Address offset: 0x4C */
  __IO uint32_t OR;          /*!< TIM option register,                 Address offset: 0x50 */
} TIM_TypeDef;

typedef struct
{
  __IO uint32_t SR;         /*!< USART Status register,                   Address offset: 0x00 */
  __IO uint32_t DR;         /*!< USART Data register,                     Address offset: 0x04 */
```

```
  __IO uint32_t BRR;        /*!< USART Baud rate register,                  Address offset: 0x08 */
  __IO uint32_t CR1;        /*!< USART Control register 1,                  Address offset: 0x0C */
  __IO uint32_t CR2;        /*!< USART Control register 2,                  Address offset: 0x10 */
  __IO uint32_t CR3;        /*!< USART Control register 3,                  Address offset: 0x14 */
  __IO uint32_t GTPR;       /*!< USART Guard time and prescaler register, Address offset: 0x18 */
} USART_TypeDef;

#define PERIPH_BASE           0x40000000UL /*!< Peripheral base address in the alias region          */
/*!< Peripheral memory map */
#define APB1PERIPH_BASE       PERIPH_BASE
#define APB2PERIPH_BASE       (PERIPH_BASE + 0x00010000UL)
#define AHB1PERIPH_BASE       (PERIPH_BASE + 0x00020000UL)
#define AHB2PERIPH_BASE       (PERIPH_BASE + 0x10000000UL)
/*!< APB1 peripherals */
#define TIM3_BASE             (APB1PERIPH_BASE + 0x0400UL)
#define USART2_BASE           (APB1PERIPH_BASE + 0x4400UL)
#define TIM3                 ((TIM_TypeDef *) TIM3_BASE)
#define USART2               ((USART_TypeDef *) USART2_BASE)
```

4.2 Registres TIM (registres et bits utiles pour l'exercice uniquement)

13.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

13.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

- This bit is set by hardware on an update event. It is cleared by software.
  0: No update occurred.
  1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow or underflow (for TIM2 to TIM5) and if UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

4.3 Registre USART (registres et bits utiles pour l'exercice uniquement)

19.6.1 Status register (USART_SR)

Address offset: 0x00

Reset value: 0x00C0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register.

0: Data is not transferred to the shift register

1: Data is transferred to the shift register)

Note: This bit is used during single buffer transmission.

19.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **TXEIE**: TXE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TXE=1 in the USART_SR register