Exam DSE TP 11/01/2021

(1) Ces lignes permettant de désactiver la chien de garde (watchdog) - c'est à dire que le watchdog ne sera pas en mesure de surveiller la bonne exécution du code. Si on les omet on aura donc des ~~interruptions~~ reset périodiques déclenchés par le ~~watchdog~~ watchdog (overflow du Tim watchdog)

(1) Ces lignes permettant de configurer le crossbar. - Elles permettent d'affecter des broches (Ports P0 à P3) aux signaux gérés par les périphériques

XBR0 = 0x04
XBR1 = 0x00
XBR2 = 0x 5C

P0.0 → TX0) UART0
P0.1 → RX0)
P0.2 → TX1) UART1
P0.3 → RX1)
P0.4 T0
P0.5 T0EX
P0.6 SYSCLOCK

Entrée CLK) Tim 4
Entrée Capture
Sortie SYSCLOCK

(2) SYSCLOCK est bien routé sur une broche du uC via le CROSSBAR → il est routé sur P0.6 -

Pour observer correctement le signal, il faut impérativement configurer P0.6 en Push Pull.

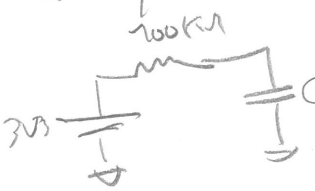
Configurer P0.6 en PUSH PULL

P0MDOUT |= 0x40; (|= (1<<6)

(4) Il y a 3 fonctions d'interruption → donc 3 Interruptions

ISR_Horodatage : interrupt 14 → Timer 3
ISR_CP_Evnt ——— 5 → Timer 2
ISR_Mesure_TH ——— 16 → Tim 4.

(5) Le Signal généré a un tps de montée très élevé ($\approx 10\mu s$)
ceci est dû au fait que la broche est configurée en Drain ouvert c'est
à dire que l'impédance de sortie au niveau haut est $\approx 100k\Omega$ (Résistance de Pull up)



$\Rightarrow t_m \approx 10\mu s = 4,2\ RC$.

En configurant la broche en Push Pull, l'impédance de sortie au niveau haut devient
égale à $\approx 50\Omega$ (~~impédance~~ impédance des Transistors mos passant)
d'où un tps de montée beaucoup plus faible $\approx 50ns$.

Config en Push Pull : → Agir sur les Registres PxMODOUT (P0 à P3)
PH7OUT (P4 à P7)

ex broche : PE3 ~~PP~~ ⇒ PEMODOUT |= 0x08;
PEMODOUT.3 à 1

(6) Interruption Périodique Tim 3.

Analyse Fonction Timer2_Init

Valeur de Rechargt : 0x B800 ⇒ 47104 ⇒ comptage de 65536 - 47104.
→ 18432 cps d'horloge

CLKTIMER = SYSCLK/12. = 1,8432 MHz

or SYSCLOCK = 22,1184 MHz (fct oscillator_Init)

période du Tim := période CLKTIMER * Nbre de cps d'horloge

$= 1/1,8432\times10^6 \times 18432.$

$= 0,01\ s = \boxed{10ms}$

(7) Interruption Timer 3 (Interrupt 14)

→ Gestion de ~~l'heure~~ l'horodatage - Récurrence 10 ms

gestion des variables Info_centième
— secondes
— minutes
— heures

(8) Source d'horloge UART

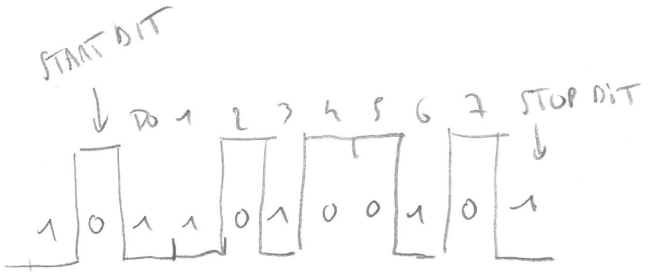
c'est le Timer 1 qui est la source d'horloge pour l'UART 0

dans CFG_UART

TICON 0x30 → mise à 0 de NCLK0 et TCLK0
↳ sélection Timer 1

(9) Analyse UART

Signaux observés sur la liaison RS232 → Logique inversée
Tension Négative : 1
— positive : 0



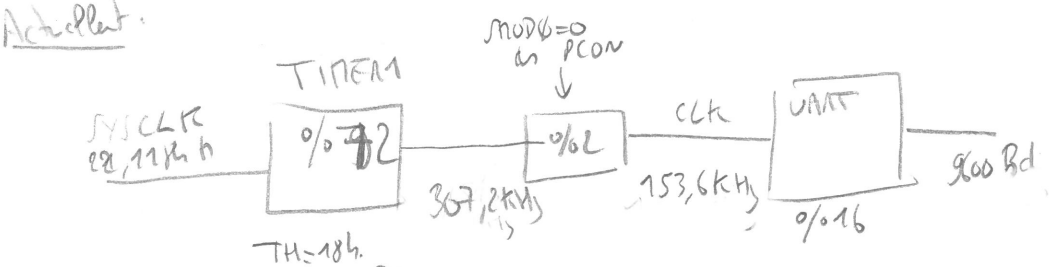
Valeur Transmise 01001011

4B

106 µs → 9433 ⇒ Vitesse de Transmission 9600 Bd

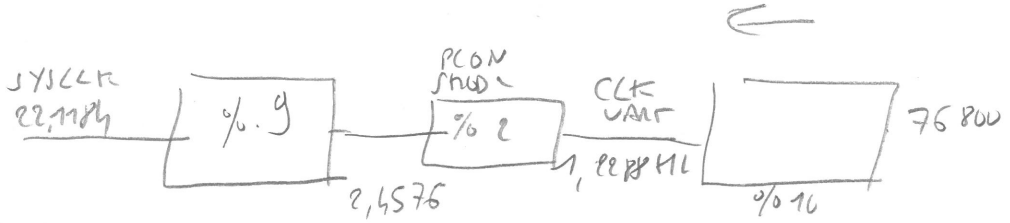
(10) Vitesse de Transmission UART

Actuellement :



TH=184
256-184=72

On veut un Baud Rate de 76800



Suite 10

Solution modifier la Config Timer 1 →
Configurer le Timer 1 pour diviser la Fréquence par 9 (au lieu de 12)
256 - 9 = 247 = 0x F7.

⇒ dans Timer 1_Init
Remplacer TH1 = 18h (0x D8) par TH1 = 242 (0xF7)

(11) Explication de ISR_CP_EVENT
c'est une fonction d'interruption déclenchée par 1 évènement Timer 2.

2 Evènements peuvent provoquer 1 interruption dans le Timer 2.
- l'overflow (TF2 = 1)
- Un évènement "capture" (EXF2) = 1

Dans le code on traite ces 2 possibilités d'interruption.

- Si TF2 = 1 ⟶ RAZ TF2
et la variable globale CP_Event (entre Pong) est incrémentée de 65536
- Si EXF2 = 1 ⟶ RAZ EXF2 (c'est tout)
↳ c'est 1 sécurité de traiter cette éventuelle interruption ~~non~~ souhaitée par EXF2

(12) · Explication ISR_Reset_TH
c'est une fonction d'interruption déclenchée par 1 évènement Timer 4
gestion de 2 variables globales : - CP_overflow
- old_Time_Capture

le Timer 4 peut faire une demande d'interruption sur 2 évènements
TF4 → overflow et EXF4 → capture.

- Si overflow : (TF4 = 1) alors RAZ TF4 et incrémentation de la variable CP_overflow.

Suite 12

- Si capture → IRQ EXTI

→ calcul d'une durée

0,0452µs correspond à la période de l'horloge Timer 4 exprimée en micro-seconde

⇒ le Tps calculé est donc en micro-seconde

Le calcul est fait en comptant le nbre d'overflows survenus depuis la dernière capture auquel on ajoute la valeur actuelle de capture et on soustrayant la valeur précédente de capture.

⇒ la valeur actuelle de capture est ensuite stockée pour être conservée pour la prochaine capture.

Pour que ce code fonctionne parfaitement il manque une remise à zéro de la variable CP_overflow.

A chaque capture → mesure du Temps et remise à zéro de la variable CP_overflow