Exercices pour le cours de GIF-3002, Systèmes Microprocesseurs et Interfaces, Partie 1 de 2

Ce document contient des exercices accompagnant les notes de cours de GIF-3002 afin de consolider et approfondir certaines notions vues en classe.

Les exercices sont divisés par cours ou par sujet présenté par l'enseignant.

De nombreuses personnes ont contribué à la réalisation de ces exercices, souvent au détriment de leur sommeil. Plusieurs exercices sont le produit d'un effort additionnel, à une heure indue. Veuillez nous pardonner les fautes, les omissions ou la mise en page minimaliste…

Bon apprentissage,

Etienne Tremblay, Chargé de cours
Marc-André Béland, Assistant de cours
Patrick Clément-Bonhomme, Assistant de cours
Marc-André Gardner, Assistant de cours
Bernard Lebel, Assistant de cours

Cours 1 : Introduction

Q1.1 Donnez l'ordre d'apparition des technologies suivantes à l'échelle mondiale :

1. ***Les transistors à effet de champs (CMOS)***
2. ***Les circuits intégrés***
3. ***Les transistors bipolaires (BJT)***
4. ***Les ordinateurs***
5. ***Les microprocesseurs***

4-3-2-1-5

Q1.2 Le processeur Intel Core i7, produit en 2010, contient environ 1.17 milliards de transistors. Combien de transistors les processeurs de 2015 devraient-ils contenir si les dimensions du processeur ne changent pas, si on utilise toujours la technologie la plus récente et si la loi de Moore est respectée?

$1.17*10^9 * 2^{5*12mois/24mois} = 1.17*10^9 * 5.66 = 6.82*10^9 transistors$

Q1.3 Dans le domaine des microprocesseurs pour PC, nommez les deux compagnies ayant le plus grand volume de ventes mondialement. Dans le domaine des microcontrôleurs, nommez trois des cinq compagnies ayant le plus grand volume de ventes mondialement. Dans le domaine des architectures de microcontrôleurs, nommez trois des cinq architectures les plus communes mondialement.

Microprocesseurs pour PC : Intel (~80%) – AMD (~19.5%) – VIA Technologie (~0.2%)
Microcontrôleurs : Renesas + NEC, Freescale/Motorola, Microchip
Architectures de microcontrôleur: Intel (8051), Renesas (740, H8/S, M32R), Freescale (68XX), PIC, ARM.

Q1.4 Dans un système microprocesseur, qui impose les données sur le bus de données, qui impose les adresses sur le bus d'adresses et qui gère les lignes de contrôle?

Le microprocesseur impose les adresses, le microprocesseur gère certaines lignes de contrôle, les périphériques gèrent le reste des lignes de contrôle (interruption, arbitration de bus) et tout le monde (microprocesseur, mémoire, périphériques) peut imposer des données sur le bus de données.

Q1.5 Afin de minimiser le nombre de broches du microprocesseur, de minimiser le nombre de fils dans un système microprocesseur et de minimiser le nombre de contrôleurs de bus nécessaires dans le microcontrôleur, il y avait traditionnellement un seul bus, comprenant des lignes d'adresses, de données et de contrôle, partant du microprocesseur. Pourquoi les microprocesseurs modernes ont-ils souvent plusieurs bus?

Beaucoup de microprocesseurs modernes ont une architecture Harvard afin d'accéder simultanément aux instructions et aux données. Il peut y avoir d'autres raisons comme par exemple pour gérer des caches.

Q1.6 Vous écrivez un programme en C ayant la ligne suivante : <a = b + c;>. Sachant que :

- ***a, b, et c sont des variables de type "short" (16 bits) aux adresses 0x1000, 0x1002 et 0x1004***
- ***Le microprocesseur exécutant le programme a 4 registres 16 bits généraux, R0, R1, R2 et R3.***

- ***Le microprocesseur exécutant le programme a une architecture LOAD/STORE (LOAD Reg, [adresse]) et supporte l'instruction ADD destination, source 1, source2.***

Écrivez la séquence d'instructions à être exécutée par le programme qui pourrait correspondre à la ligne de code <a = b + c;>.

LOAD R0, [0x1000]
LOAD R1, [0x1002]
ADD R2, R1, R0
STORE R2, [0x1004]

Q1.7 Dites quelles instructions se retrouvent dans le microprocesseur et décrivez le(s) programme(s) auxquels ces instructions appartiennent. Dites également quelles composantes servent à contenir les données dans un microprocesseur.

Les seules instructions qui se retrouvent dans le microprocesseur sont les instructions qui sont à être exécutée : elles appartiennent au(x) programme(s) en cours d'exécution. Seuls les registres contiennent des données dans le microprocesseur.

Q1.8 Pour les applications suivantes, dites s'il faudrait utiliser un microprocesseur ou un microcontrôleur et dites pourquoi :

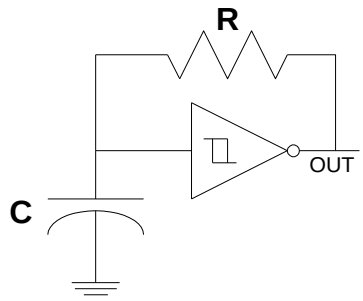
1. ***Contrôleur de grille-pain***
2. ***Contrôleur d'automobile***
3. ***Téléphone cellulaire***
4. ***Ordinateur portable***
5. ***Supercalculateur***

1. Un microcontrôleur : simple, peu couteux, pas besoin d'une grande puissance de calcul, une seule application
2. Un microcontrôleur : simple, peu couteux, pas besoin d'une grande puissance de calcul, une seule application. Également plus robuste car tout intégré.
3. Un microcontrôleur : prend moins d'espace, consomme moins d'énergie, application spécifique.
4. Un microprocesseur : beaucoup de puissance de calcul requise, applications diverses exécutées, versatilité requise, possibilité d'amélioration des composantes externes.
5. Un microprocesseur : beaucoup de puissance de calcul requise, applications diverses exécutées, versatilité requise, possibilité d'amélioration des composantes externes.

Q1.9 Décrivez deux méthodes utilisées pour programmer la FLASH d'un microcontrôleur, sachant que l'adresse initiale, au démarrage, du compteur de programme des microcontrôleurs (PC) désigne souvent la FLASH ou une autre mémoire non-volatile.

1) Une mémoire ROM ou la FLASH elle-même contient un programme s'exécutant après la mise sous tension qui permet de reprogrammer la FLASH.
2) Du matériel permet de prendre le contrôle du bus relié à la FLASH à la place du microprocesseur et reprogramme la FLASH.

Q1.10 Quel signal est généré sur la broche OUT du circuit suivant?



Le circuit génère un signal d'horloge dont la fréquence dépend de R, C et des tensions de seuil de l'inverseur Schmitt Trigger.

Q1.11 Décrivez les tâches d'un microprocesseur.

Voir section 2.1.1, point 2 du fichier Introduction.pdf

Q1.12 Pour quelle raison la logique trois états existe-t-elle?

La logique trois états existe afin d'enlever l'influence d'un dispositif sur un circuit, dans le cas où plusieurs dispositifs sont connectés sur la même ligne. Cela empêche les courts-circuits et le cas où un dispositif tente de mettre un 1 logique sur la ligne alors qu'un autre tente de mettre un 0 logique.

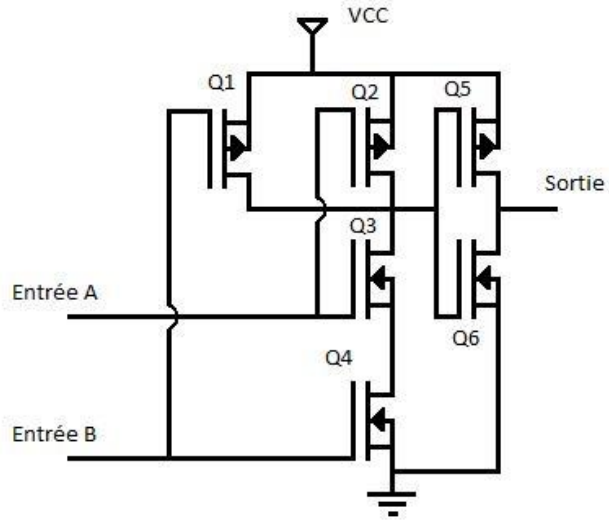
Q1.13 Quelle est la différence entre un microprocesseur et un microcontrôleur? Donnez les avantages et inconvénients de chacun.

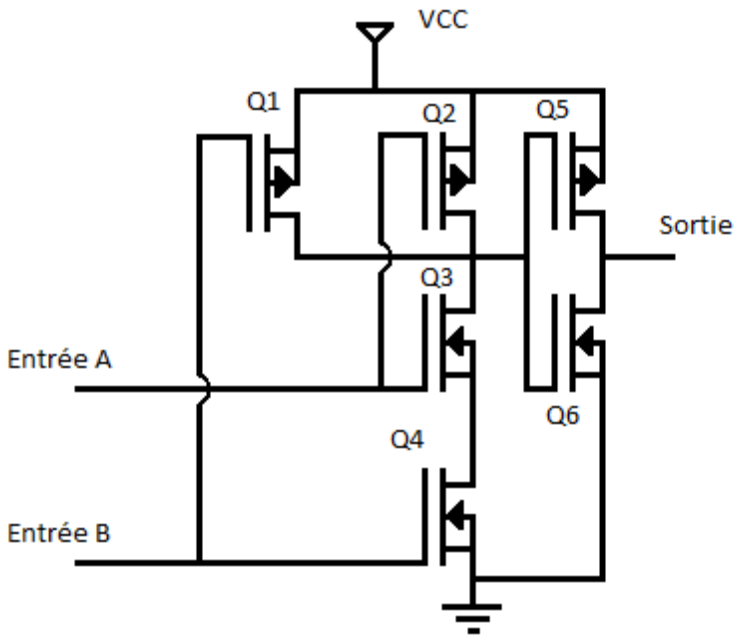
Le microcontrôleur est un circuit intégré qui comporte, sur le même die, un microprocesseur, de la mémoire et des périphériques.
Avantages et inconvénients : voir section 2.5.1 du fichier Introduction.pdf

Q1.14 Il faut brancher le bus de données d'un microcontrôleur à deux mémoires pour un interfaçage en lecture et écriture. Chaque ligne du microcontrôleur peut fournir jusqu'à 10 mA. Or, chaque entrée sur les mémoires nécessite 6 mA. Comment est-il possible de régler ce problème?

Il est possible d'utiliser des buffers bidirectionnels (exemple : http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/54FCT245.pdf)

Q1.15 Dans le circuit suivant, les entrées peuvent être égales soit au VCC, soit à la masse. Identifier le type de chaque MOSFET du circuit. Écrivez la table de vérité de ce circuit. Quelle est la fonction logique de ce circuit?



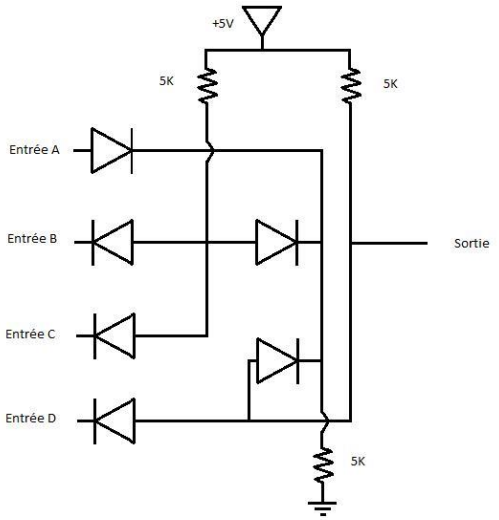


Q1, Q2, Q5 : P; Q3, Q4, Q6 : N

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Fonction logique : $S = A \cdot B$

Q1.16 Quelle est la fonction logique du circuit suivant? (On assume que la perte de tension dans chaque diode est de 0,7V et que les quatre entrées peuvent être seulement seulement 5V ou 0V, c'est-à-dire un '1' ou un '0' logique)



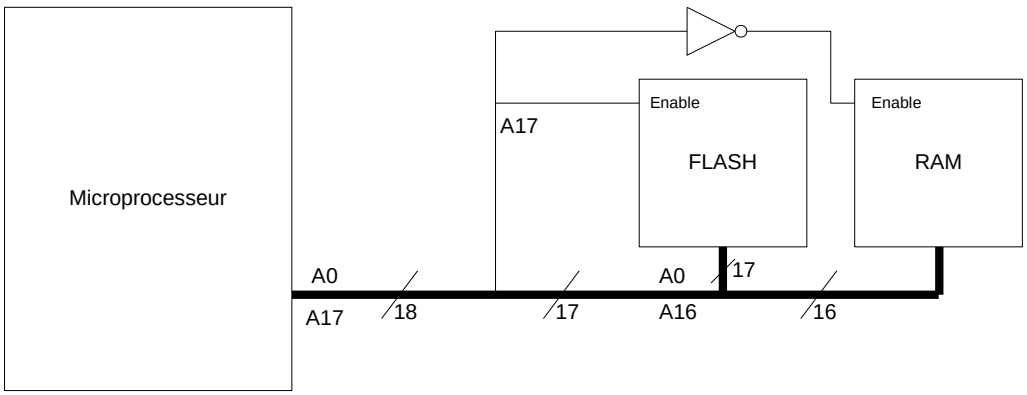
Fonction logique : $S = (A+(B \cdot C)) \cdot D$

Q1.17: Dans un système avec microprocesseur, pourquoi une architecture Harvard est-elle très avantageuse lorsque le microprocesseur a un pipeline d'instructions?

Le bus d'instruction et le bus de données indépendant de l'architecture Harvard permettent de lire une instruction tout en exécutant une autre instruction qui écrit ou lit une donnée.

Q1.18: Vous avez un système microprocesseur avec deux mémoires, une mémoire FLASH de 128K-mots et une mémoire RAM de 64K-mots. Dessinez le bus d'adresse de ce système avec le décodeur d'adresse. Notez que $K = 2^{10}$ et que vous devez déterminer les éléments non-établis du problème (exemple: l'adresse de base des mémoires) comme il vous plaira.

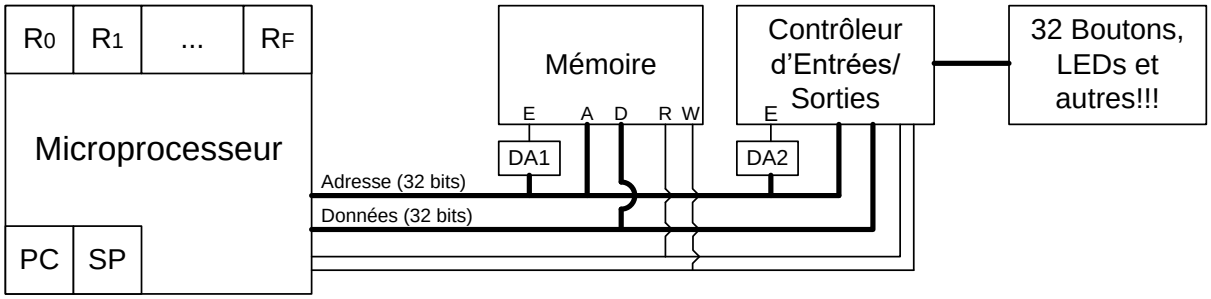
En raison de l'énoncé, il existe plusieurs solutions. Cependant, votre solution ne doit contenir qu'un seul bus d'adresse.



Q1.19: Les notes de cours décrivent sommairement le protocole NMEA 0183 pour représenter les fractions. Dites pourquoi ce protocole est peu utilisé dans l'industrie pour représenter les autres fractions.

Les fractions représentées avec des caractères ascii occupent énormément d'espace et elles sont très difficiles à manipuler (opération arithmétiques très complexes!)

Q1.20 (récapitulatif) Supposons le système suivant :



Dans ce système :

- ***Le microprocesseur fonctionne sur 32 bits : les registres sont 32 bits, les adresses sont 32 bits, le bus de données a 32 bits et toutes les instructions sont sur 32 bits.***
- ***Le microprocesseur a 16 registres tout usage utilisés pour faire des calculs.***
- ***Le microprocesseur a un registre PC, Program Counter, indiquant l'adresse de la prochaine instruction à exécuter. Le PC est automatiquement incrémenté de 4 après la lecture d'une instruction en mémoire.***
- ***Le microprocesseur a un registre SP, le Stack Pointer. Lors d'un PUSH, la donnée est mise sur la pile à l'adresse indiquée par SP, puis SP est automatiquement décrémenté de 4. Lors d'un POP, SP est incrémentée de 4, puis la donnée de l'espace mémoire indiqué par SP est stockée à l'endroit requis par l'instruction.***
- ***Lorsqu'une interruption se produit, le microprocesseur exécute automatiquement les tâches suivantes, dans l'ordre :***
    - ***Mettre la valeur de PC actuelle sur la pile (SP est automatiquement décrémenté après l'opération)***
    - ***Lire l'adresse 4*(numéro d'interruption).***
    - ***Prendre la valeur lue et la mettre dans PC : la table des vecteurs d'interruption commence à l'adresse 0…***
- ***Le microprocesseur supporte les instructions décrites ci-dessous (et plus!). L'opcode des instructions est sur 8 bits et les paramètres de chaque instruction sont sur 24 bits. Il faut 4 bits pour indiquer un des 16 registres tout-usage du microprocesseur.***

Opcode	Paramètres
bits 31 à 24	bits 23 à 0

Tableau 1. Instruction sur 32 bits.

- ***La mémoire est activée lorsque l'adresse, sur le bus d'adresse, est entre 0x00000000 et 0x0004FFFF : le décodeur d'adresse DA1 active la mémoire (Enable) lorsque l'adresse est entre ces valeurs.***
- ***Un contrôleur d'entrées sorties est activé lorsque les adresses sont 0x00060000 et 0x00060004. Ce contrôleur permet de lire ou d'écrire des entrées sorties digitales en fonction du registre de direction (0 = entrée, 1 = sorties). Le registre de valeur indique la valeur à mettre sur les sorties ou la valeur lue des entrées.***

Répondez aux questions en fonction des tables suivantes :

Mémoire	
Adresse	Valeur
0x00048100	0xFFFFFFFF
0x000480FC	0xFFFFFFFF
0x000480F8	0xFFFFFFFF
0x000480F4	0xFFFFFFFF
…	…
0x00040008	0x00000003
0x00040004	0x00000002
0x00040000	0x00000001
…	…
0x00000220	0x40FFFFFF
0x0000021C	0x21500000
0x00000218	0x21600000
0x00000214	0x117480FC
0x00000210	0x30765FFF
0x0000020C	0x10660004
0x00000208	0x10540008
0x00000204	0x20600000
0x00000200	0x20500000
…	…
0x00000008	0x00000400
0x00000004	0x00000200
0x00000000	0x00000100

Instruction	Opcode	Registre	Adresse
LOAD Rx, [Mem]	0x10	4 bits	20 bits
Store Rx, [Mem]	0x11	4 bits	20 bits
	Opcode	Registre	
PUSH Rx	0x20	4 bits	
POP Rx	0x21	4 bits	
	Opcode	Rdest	Rsrc1, Rsrc2
ADD Rdest, Rsrc1, Rsrc2	0x30	4 bits	8 bits
	Opcode		
Retour d'interruption	0x40		

Périphérique contrôlant 32 broches		
Registre	Adresse	Valeur initiale
Direction	0x00060000	0x00000000
Valeur	0x00060004	0x00000000

Valeur initiale des registres	
R4	0x00000004
R5	0x00000005
R6	0x00000006
R7	0x00000007

a) Si PC vaut 0x00000400 et SP vaut 0x000481000 lorsqu'une interruption #1 se produit, quelles seront les valeurs de PC et SP lorsque la première instruction de l'ISR (la routine traitant l'interruption) sera lue? Quelle valeur de la mémoire changera?

On retrouve 0x00000200 à l'adresse 4. PC vaudra donc 0x00000200. Par ailleurs, l'adresse de retour sera mise sur la pile (la mémoire à l'adresse 0x00048100 deviendra 0x00000400) et SP sera décrémenté de 4 (0x00048100FC).

b) Quelle est la première instruction exécutée dans l'ISR? Pourquoi retrouve-t-on habituellement cette instruction au début d'une ISR?

La première instruction est PUSH R5. On met les registres sur la pile pour éviter que ceux-ci soient changé lorsqu'on reviendra de l'interruption.

c) Décrivez les signaux qui apparaîtront sur les bus du système lorsque la troisième instruction de l'ISR sera lue, puis exécutée.

Lecture de l'instruction :

- Le PC vaut 0x00000208. Cette valeur est mise sur le bus d'adresse par le microprocesseur.
- Puis, la ligne de lecture de la mémoire est activée
- Ensuite, la mémoire met la valeur 0x10540008 sur le bus de données.
- Le microprocesseur lit l'instruction 0x10540008.

Exécution de l'instruction :

- Le microprocesseur décode l'instruction LOAD (10) R5 (5), [40008].
- Le microprocesseur met 40008 sur le bus d'adresse.
- Puis, la ligne de lecture de la mémoire est activée

- Ensuite, la mémoire met la valeur 0x00000003 sur le bus de données.
- Le microprocesseur lit la données sur le bus et met 3 dans R5.

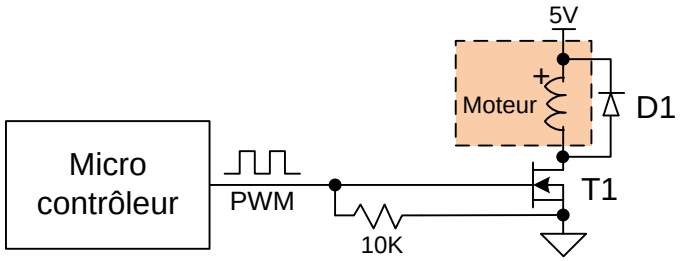
d) Quelles seront les valeurs des registres R5, R6 et R7 après l'addition?

R5 vaut 3.
R6 vaut la valeur retrouvée sur les broches du périphérique.
R7 vaut 3 + la valeur retrouvée sur les broches du périphérique.

e) Quelles valeurs retrouvera-t-on aux adresses de 0x000480F4 à 0x000480100 après l'exécution de l'ISR?

0x00048100 --- 0x00000400 (Adresse de retour de l'interruption)
0x000480FC --- 3 + la valeur retrouvée sur les broches du périphérique. (Instruction Store)
0x000480F8 --- 0x00000006 (PUSH de R6)
0x000480F4 --- 0xFFFFFFFF (aucun changement)
R6 vaut la valeur retrouvée sur les broches du périphérique.

Q1.21: On utilise presque toujours une diode en parallèle avec une inductance (ou un moteur) lorsque cette inductance est reliée à un interrupteur (habituellement un transistor; dans un power supply SMPS par exemple). La figure ci-dessous illustre ce propos avec un contrôleur typique de moteur DC. Expliquez à quoi sert cette diode (D1 sur la figure)?



Lorsque l'interrupteur est fermé et conduit le courant, la diode est inutile. Par contre, lorsque l'interrupteur s'ouvre, l'inductance produit un voltage qui s'oppose à la variation de courant : V = Ldi/dt. S'il n'y avait pas de diode, le voltage produit par l'inductance, instantanément très grand lorsque le courant passe de X à 0 subitement (l'interrupteur ouvre), causerait des arcs et brûlerait l'interrupteur. Avec une diode, la tension créée par l'inductance ne dépasse pas la tension Vf de la diode : le courant continue de circuler temporairement par D1 lorsque l'interrupteur ouvre.

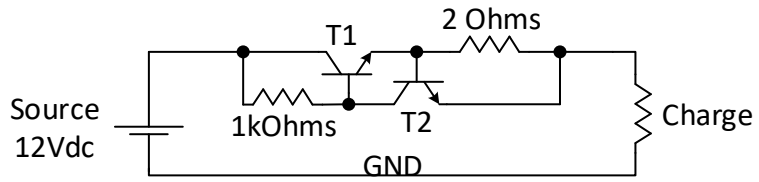
Q1.22 : Un programmeur écrit la ligne de code suivante en C: Toto = 3.2;. Toto est défini comme un float. Le code est compilé sur un microprocesseur ARM Cortex M4F ayant un FPU. Ce microprocesseur a 32 registres 32 bits S0 à S31 disponibles pour manipuler les fractions et tout un jeu d'instruction réservé à cet effet. Toutes les instructions utilisées pour manipuler des fractions commencent par V. Par exemple, on retrouvera VLDR pour lire une variable en mémoire et mettre sa valeur dans un registre de S0 à S31. Lorsque le programmeur regarde le code assembleur correspondant à Toto = 3.2;, il retrouve les instructions suivantes :

```
;Code pour Toto = 3.2;
LDR R0, [PC – 24]
LDR R1, [PC – 24]
STR R0, [R1]
```

Expliquez pourquoi ce code ne contient aucun registre relié aux fractions, aucune instruction reliée aux fractions et pas de référence explicite à Toto. En d'autres mots, expliquez comment le code fonctionne.

Le compilateur a mis 0x404ccccd (3.2 selon la norme IEEE754) comme une constante à une adresse 24 octets[1] avant le LDR R0, [PC-24]. Il a aussi mis l'adresse de Toto comme une constante à une adresse 24 octets avant le LDR R1, [PC-24]. Enfin, le store met 3.2 (temporairement dans R0) dans Toto (dont l'adresse est désignée par R1)…

Q1.23 : On retrouve le circuit suivant pour limiter le courant provenant de la source, lorsque la charge est trop grande (résistance trop faible ou court-circuit). Expliquez comment ce circuit fonctionne :



Lorsque le courant n'est pas élevé, T1 conduit et T2 ne conduit pas. La source 12Vdc produit une tension supérieure à celle de la charge (reliée à GND) et la tension Vbe de T1 est supérieur à 0.7. Par contre, la tension Vbe de T2 est égale au courant circulant dans la Charge multiplié par la résistance de 2 Ohms : cette tension n'est pas assez grande pour que T2 conduise.

Lorsque le courant est trop élevé, la tension Vbe de T2 devient assez grande pour que T2 conduise. En conduisant, T2 vient court-circuiter la base et l'émetteur de T1 : la tension Vbe de T1 devient plus petite de telle sorte que T1 conduit moins.

Lorsque T1 conduit moins, cela réduit le courant dans T2 qui conduit moins. Ainsi, T1 peut conduire plus, augmentant le courant dans la résistance de 2 Ohms qui, de nouveau, fait moins conduire T1. Il s'établit un équilibre. Le courant est limité à la valeur en ampères, qui, multipliée par 2 Ohms, fait conduire un peu T2…

[1] Ce n'est pas exactement 24 octets en raison du pipeline d'instruction du microprocesseur, mais plutôt 16 octets… Toutefois, vous pouvez considérer 24 octets pour comprendre la réponse.

Cours 2 Micro-Matériel

Q2.1 Énumérez 8 critères qui devraient guider la sélection d'un microprocesseur.

Voir la section 1.1 de Micro_Materiel.doc

Q2.2 Dites quels mots sont définis ci-dessous :

Définition	***Mot***
Circuit ayant pour fonction de faire un reset du microprocesseur lorsque celui-ci cesse de fonctionner normalement	Watchdog
Ensemble de programmes servant à évaluer les performances d'un microprocesseur	Benchmark
Se dit d'un contrôleur d'interruptions lorsque les adresses des routines d'interruptions sont configurables et lues dans une table d'adresses.	Vectorisé
Circuit permettant de transférer des données de la mémoire aux périphériques sans intervention du CPU	DMA
Circuit générant des ondes rectangulaires à fréquence variable et à largeur variable	PWM

Q2.3 Lorsque l'on conçoit un circuit électronique avec un microprocesseur, quels sont les circuits de base accompagnant le microprocesseur qui doivent être élaborés? (Exemple de circuit de base : les alimentations du microprocesseur…).

Alimentation, reset, horloge, mémoires, programmation du code initial exécuté par le mocroprocesseur, entrées/sorties et périphériques.

Q2.4 Pour quelle(s) raison(s) les microcontrôleurs opèrent-ils habituellement entre 10MHz et 100MHz alors que les microprocesseurs opèrent à des vitesses beaucoup plus grandes?

Parce que les temps d'accès aux différents types de mémoires sont de l'ordre de 10ns à 50ns.

Q2.5 Qu'est-ce qu'une Nested Vectored Interruption?

Les interruptions imbriquées (nested) permettent aux routines d'interruption d'être mises de côté pour un traitement ultérieur si une routine d'interruption plus prioritaire est en cours d'exécution. Une interruption vectorisée (vectored) indique que le microprocesseur utilise un vecteur, c'est-à-dire un pointeur de fonction afin de déterminer l'adresse de la séquence d'instructions correspondant à l'interruption.

Q2.6 Pourquoi est-il déconseillé de faire une horloge indiquant le temps à l'aide de l'oscillateur interne d'un microcontrôleur? Comment peut-on faire une telle horloge avec le kit microcontrôleur du cours?

L'oscillateur interne d'un microcontrôleur est souvent très imprécis, ce qui peut décaler d'une seconde à toutes les 2 minutes. Il est possible de configurer les timers du kit du cours en RTC (Real Time Clock, horloge temps réel), ce qui peut faire ce travail, mais il faut utiliser l'horloge externe qui est plus précise...

Q2.7 Quel est l'intervalle entre les différentes tensions pour un ADC 12 bits dont la tension d'entrée varie entre -3,3 V et 3,3 V ?

$(3{,}3V - (-3{,}3V))/2^{12} = 1{,}61$ mV

Q2.8 Pourquoi les microprocesseurs ont-ils besoin de plusieurs broches d'alimentation?

Le coeur d'un microprocesseur nécessite généralement une tension différente des entrées/sorties. Les entrées/sorties peuvent nécessiter différentes tensions, par exemple dans le cas d'un ADC où l'alimentation doit être stable afin de ne pas fausser la mesure. Il peut aussi y avoir une alimentation pour un RTC qui doit toujours être alimenté à partir d'une pile lorsque le système est éteint.

Q2.9 Dites pourquoi certaines broches de microcontrôleur on parfois le rôle d'adresse et de données, multiplexé temporellement.

Cela permet de diminuer le nombre de broches requises et d'adresser un plus grand nombre de périphériques.

Q2.10 Qu'est-ce qu'un MAX232?

Il s'agit d'un circuit intégré permettant de convertir un signal d'un port série RS232 vers un signal convenable pour la logique TTL.

Q2.11 Dans un microcontrôleur moderne, quelle composante limite habituellement la vitesse de lecture et d'exécution des instructions? Quelles stratégies sont utilisées pour contournée les limites de cette composante?

La mémoire FLASH limite habituellement la vitesse de lecture des instructions entre 50MHz et 80MHz. Il existe plusieurs stratégies pour contourner le problème : lire les in

Q2.12 Quelles stratégies peuvent être utilisées pour réduire la consommation de puissance d'un microcontrôleur?

Réduire la fréquence d'horloge, réduire le voltage d'alimentation, réduire le nombre de transistors alimentés.

Cela peut se traduire par des modes de sommeil (pendant lesquels l'horloge du cœur est coupée par exemple), par le délestage de périphériques inutilisés, par une réduction de la puissance de calcul en réduisant la fréquence sortant de la PLL, par une alimentation DC du cœur la plus basse possible, par…

Q2.13 Dans un contexte de microcontrôleur, que signifient les acronymes BGA, QFP, DIP?

BGA = Ball Grid Array
QFP = Quad Flat Package
DIP = Dual Inline Pins

Il s'agit de format des boitiers et broches de circuits intégrés.

Q2.14 Pourquoi l'identifiant unique d'un microcontrôleur n'est-il pas habituellement écrit dans la FLASH du microcontrôleur?

Parce qu'on veut pouvoir effacer la FLASH afin de reprogrammer le microcontrôleur sans perdre l'identifiant unique.

Q2.15 Pourquoi permet-on de commencer l'exécution de code après un reset à plusieurs adresses, habituellement selon certaines broches du microcontrôleur?

- Exécution du code en FLASH pour exécuter le code de l'usager
- Exécution du code en ROM pour reprogrammer la FLASH
- Exécution du code en RAM pour tester ou déverminer un code en RAM sans changer le contenu de la FLASH
- Exécution du code en RAM pour accélérer la reprise des activités après un reset (si la RAM n'a pas été effacée!)

Cours 3 Micro-Logiciel

Q3.1 Vous retrouvez les instructions et données suivantes dans la mémoire d'un microcontrôleur. Assumez qu'un reset vient de se produire et que l'adresse du vecteur d'interruption traitant le reset est 0x00000004 et répondez aux questions suivantes (le registre R15 est le Program Counter).

Note 1: Le code qui suit peut contenir des erreurs sur le -8 et le -6 des LDR R1, [R15 - 8] et LDR R1, [R15 - 6] en raison du pipeline d'instruction du microcontrôleur. Afin de répondre aux questions, considérez que le PC (R15) ne varie pas pendant l'exécution d'une instruction.

Note 2: Assumez que le microprocesseur opère en little endian. Si vous retrouvez 0x12345678 à l'adresse 0x00001000, cela signifie qu'il y a 0x78 à l'adresse 0x00001000, 0x56 à l'adresse 0x00001001, 0x34 à l'adresse 0x00001002 et 0x12 à l'adresse 0x00001003.

Adresse	***Valeur***
0x00000000	***0x00000000***
0x00000004	***0x00001004***
…	
0x00001000	***0x12345678***
0x00001004	***0x20000000***
0x00001008	***LDR R1, [R15 – 8]***
0x0000100A	***LDR R2, [R15 – 6]***
0x0000100C	***LDR R3, [R2+2]***
0x0000100E	***ADD R1, R2, R3***
0x00001010	***ADD R1, R1, 0x1234***
…	
0x20000000	***0xAAAA1111***
0x20000004	***0x2222BBBB***

1. ***On utilise six octets de mémoire pour mettre 0x12345678 dans le registre R1 du microprocesseur. Peut-on optimiser le code pour faire la même opération avec moins d'octets? Si oui, comment? Si non, pourquoi?***
2. ***À quelle adresse devrait se retrouver l'instruction suivant cette séquence d'instruction?***
3. ***Donnez la valeur du registre modifié par chaque instruction de la séquence. Dites quelle sera la valeur de R1 après le premier LDR, la valeur de R2 après le second LDR et ainsi de suite.***

1. Il faut quatre octets pour représenter la valeur 0x12345678. Il faut aussi au moins un octet pour l'opcode et quelques bits pour désigner R1. Donc, il faut au moins six octets pour faire cette opération.
2. L'instruction ADD R1, R1, 0x1234 est certainement une instruction 32-bits du jeu d'instruction Thumbs-2. L'adresse de la prochaine instruction sera donc 0x00001010 + 4, soit 0x00001014.
3.

0x00001008	LDR R1, [R15 – 8]	R1 = 0x12345678
0x0000100A	LDR R2, [R15 – 6]	R2 = 0x20000000
0x0000100C	LDR R3, [R2+2]	R3 = 0xBBBBAAAA
0x0000100E	ADD R1, R2, R3	R1 = 0xDBBBAAAA
0x00001010	ADD R1, R1, 0x1234	R1 = 0xDBBBBCDE

Q3.2 Qu'est qu'une banque de registre ou une banque de mémoire? Pourquoi utilise-t-on des banques de registres ou de mémoire?

Une banque de registre ou de mémoire apparaît lorsque plusieurs copies du registre ou de la mémoire coexistent mais qu'une seule copie à la fois de la banque est visible de l'extérieur de la banque. Une seule copie est active et la façon d'accéder à la copie ne change pas, peu importe qu'elle copie est active.

Les banques de mémoire ou de registre sont très utiles pour accélérer les changements de contexte de programme.

Q3.3 Qu'est-ce qu'une architecture Load/Store?

Une architecture Load/Store est une architecture où les seules instructions permettant d'accéder à la mémoire de données (pile exclue) sont les instructions Load et Store (LDR et STR pour le ARM Cortex M3 ou M4).

Q3.4 À quoi sert le bit-banding du ARM_Cortex-M3? Quelle séquence d'instruction est optimisée par le bit-banding? Donnez un exemple de code assembleur.

Le bit banding sert à optimiser les manipulations de bits. Un exemple de séquence d'instructions optimisée est :

//Exemple de Set bit 0x04 de l'octet désigné par *Adresse*
LDR Rx, [*Adresse*]
OR Rx, Rx, 0x04
STR Rx, [*Adresse*]

Cette séquence pourrait être remplacée par une ou deux instructions si *Adresse* est entre 0x20000000 et 0x20000000+1Mo. L'adresse de la région de bit-band à écrire est calculée comme suit :

//Voir CortexM3_TRM_r2p0.pdf, p.93 pour plus de détails… ou réfléchir à la question!

bit_word_offset = (byte_offset x 32) + (bit_number × 4)
bit_word_addr = bit_band_base + bit_word_offset

bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number × 4)

MOV R1, 1
STR R1, [0x22000000 + (*Adresse* – 0x20000000)*32 + 2*4]

Q3.5 Quelle séquence d'opérations effectue le microprocesseur ARM Cortex M4 lorsqu'il entre ou sort d'une interruption?

Voir sections 4.3 et 4.4 du fichier Micro_Logiciel.pdf

Q3.6 Nommer les différents modes d'adressage du ARM Cortex M4?

- Mémoire/Registre
--- L'adresse de l'instruction courante + une constante (PC + immediate)
--- Le dessus de la pile + une constante (SP + immediate)
--- La valeur d'un registre + une constante (indirect indexed)
--- La somme de deux registres
- Registre/Registre
-Constante/Registre

Q3.7 Quelle(s) composante(s) matérielle(s) place(nt) les mémoires et les périphériques aux adresses indiquées dans le « memory map »?

Le(s) décodeur(s) d'adresse.

Q3.8 Dites si les énoncés suivants sont vrais ou faux. S'ils sont faux, dites pourquoi.

#	Énoncé	V/F
A	Dans un système PMIO, il y a une adresse 0 pour les périphériques et une adresse 0 pour les mémoires. Dans les systèmes MMIO, il y a une seule adresse 0.	V
B	Toutes les mémoires ont un registre qui détermine la première adresse de la mémoire dans le système.	F
	Toutes les mémoires ont une broche Enable qui permet au décodeur d'adresse de d'activer ou de désactiver la mémoire en fonction de l'adresse.	
C	Dans tous les systèmes MMIO, les instructions qui accèdent à la mémoire accèdent aussi aux périphériques. Dans tous les systèmes PMIO, les instructions qui accèdent aux périphériques activent des signaux de contrôle qui permettent de décoder une adresse de périphérique.	V
D	Il faut que chaque adresse du système désigne un mot de mémoire ou un périphérique pour que le système fonctionne.	F
	Le microprocesseur lit et exécute des instructions. S'il adresse un mot qui n'existe pas, le microprocesseur lira des données « aléatoires » en fonction du design des bus.	

Q3.9 Qu'est qu'un alias d'adresses?
Il s'agit de deux ou plusieurs adresses qui désignent le même mot de mémoire ou registre de périphérique.

Q3.10 Nommez et décrivez deux stratégies utilisées dans le cœur ARM pour diminuer le temps de latence moyen avant d'entrer dans une interruption.
Voir la section sur la latence des interruptions dans les notes de cours C3.

Q3.11 Pourquoi la table des vecteurs d'interruptions du cœur ARM commence-t-elle par le vecteur de reset et par la valeur initiale du stack pointeur?
Pour accélérer le démarrage du microcontrôleur et permettre les interruptions dès la première instruction.

Q3.12 [Enrichissement] Le ARM Cortex M4 contient une banque de registres pour le pointeur de pile. Cela permet de sauver beaucoup de mémoire RAM lorsqu'un système d'exploitation préemptif est utilisé. Expliquez comment et pourquoi.
Avec un système d'exploitation préemptif, chaque processus en mémoire doit avoir une pile qui lui est propre. Le système d'exploitation change le pointeur de pile lorsqu'il change l'application exécutée après un quantum (temps de CPU fixe alloué à un processus avant que l'OS ne décide le prochain processus exécuté).

Lors de l'exécution d'un processus, plusieurs évènements reliés à la pile peuvent survenir : sauvegarde de données sur la pile, appel de fonction, interruption, utilisation de la pile pour faire des calculs, etc. Le système empile des informations sur la pile et les dépile dans chacun de ces cas et la taille de la pile doit être calculée afin de contenir toutes les informations pertinente :

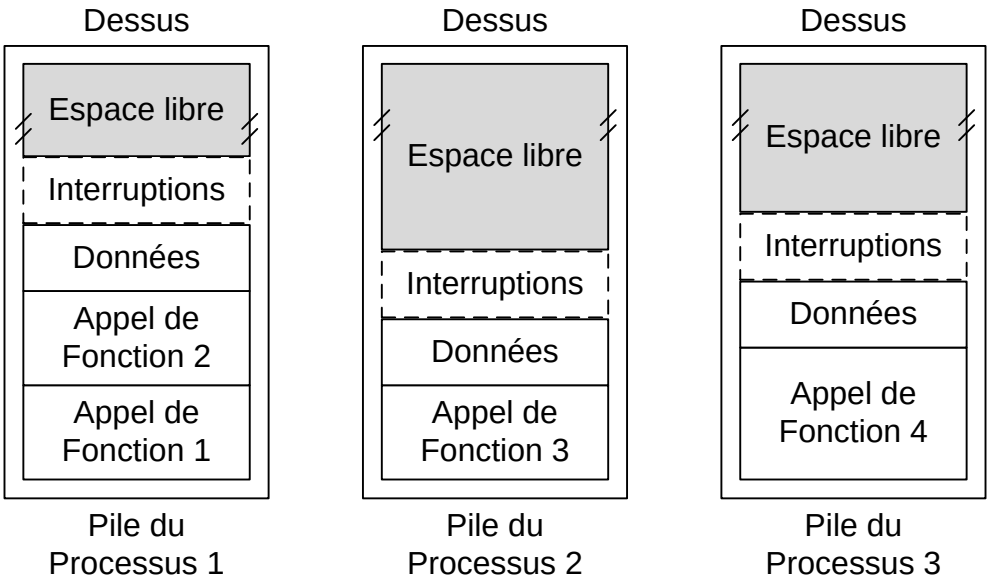


Illustration de piles de processus avec un OS préemptif, pointeur de pile unique

Habituellement, les interruptions des périphériques appartiennent au système d'exploitation : le OS implémente les fonctions traitant les interruptions (ISRs) et offre une interface aux processus de l'usager pour accéder au matériel… Par ailleurs, les interruptions de périphérique ont habituellement une priorité plus haute que celle de l'OS (un timer périodique) : pendant une interruption de périphérique, le processus en cours d'exécution ne peut pas changer. Cela implique que seul le processus actif utilise peut contenir des données reliées aux interruptions.

Lorsqu'une interruption (ou plusieurs interruptions) de périphérique se produit, de l'information (comme l'adresse de retour) est sauvegardée sur la pile. Même si une seule pile peut contenir les données des interruptions, chaque processus doit avoir une pile assez grande pour sauvegarder les informations reliées à celles-ci, s'il n'y a pas de banque de pointeur de pile.

Par contre, s'il y a une banque de pointeur de pile, un pointeur pile peut être utilisé pour une pile réservée aux interruptions et l'autre pointeur de pile peut être utilisé pour les piles des processus :

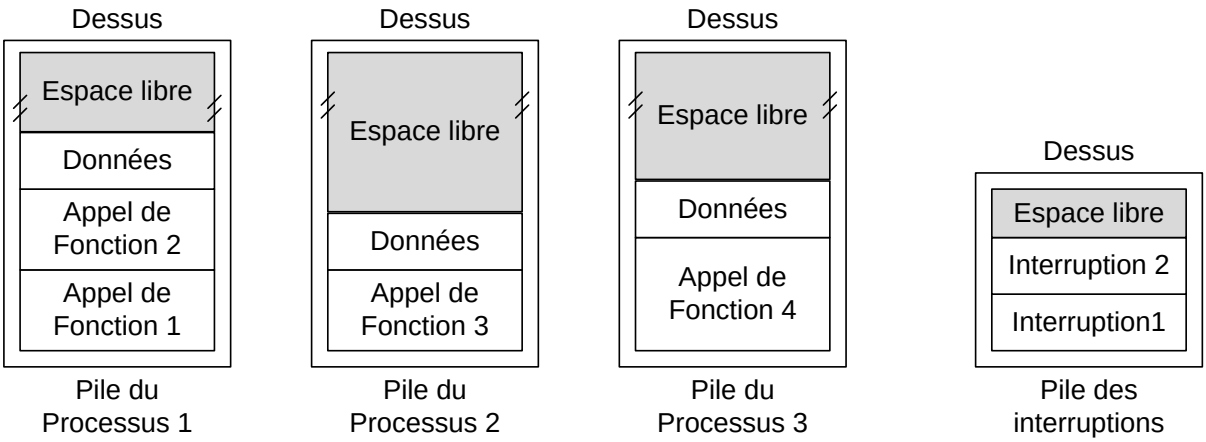


Illustration de piles de processus avec un OS préemptif, avec banque de pointeurs de pile

Cours 4 Mémoires, Matériel

Q4.1 Énumérez les signaux nécessaires afin de lire une mémoire ROM et indiquez le rôle de chacun des signaux.

Les signaux suivants ou leurs équivalents apparaîtront :

- Chip Enable : est utilisé par le décodeur d'adresse pour activer la mémoire et connecter celle-ci au bus de données.
- Output Enable : est relié au bus de contrôle pour signaler et permettre la lecture de la mémoire
- Bus d'adresse : désigne le mot de mémoire qui sera lu
- Bus de données : données provenant de la mémoire

Q4.2 Expliquez le fonctionnement de la mémoire SRAM.

Le bit
Le bit de DRAM est deux inverseurs dos-à-dos (4 transistors) et de deux inverseurs pour activer le bit.

Le mot
Les bits de DRAM sont regroupés en mots : chaque mot contient N bits qui sont tous activés (via les transistors) par le même signal.

Organisation de la mémoire
Les mots de la mémoire SRAM sont organisés en vecteur à une dimension. Un mot est relié au bus de données (à travers un buffer) en fonction de l'adresse de mémoire choisie.

Interface
Pour accéder à la mémoire SRAM, il faut contrôler au moins les signaux suivants, ou leur équivalent : nCE, les adresses, les données, nOE et nWE.

Q4.3 Expliquez le fonctionnement de la mémoire DRAM.

Le bit
Le bit de DRAM est constitué d'un transistor et d'un condensateur.
La tension aux bornes du condensateur est comparée avec une tension spécifique pour savoir si le bit vaut 1 ou 0.
Un ampli permet de charger le condensateur lorsqu'on écrit la mémoire.
Comme le condensateur se décharge par des résistances de fuite, il faut rafraîchir la mémoire régulièrement : recharger le condensateur.

Le mot
Les bits de DRAM sont regroupés en mots : chaque mot contient N bits qui sont tous activés (via le transistor) par le même signal.

Organisation de la mémoire
Les mots de la mémoire DRAM sont organisés en matrice à deux dimensions. Il y a des rangées de mots et des colonnes de mots.
Pour lire ou écrire la DRAM, on choisit d'abord une rangée : tous les mots de la rangée choisie deviennent connectés (via le transistor) à leur colonne. On choisit en ensuite la colonne : un multiplexer (ou équivalent) relie la colonne choisie au bus de données à travers un tampon.

Interface
Pour accéder à la mémoire DRAM, il faut contrôler au moins les signaux suivants, ou leur équivalent : nRAS, nCAS, nCE, les adresses, les données et une broche de lecture/nÉcriture.

Q4.4 Expliquez pourquoi les mémoires DRAM ont, pour une même quantité de broches, plus de capacité que les mémoires SRAM qui ne font pas multiplexage adresses/données.

Les adresses de rangée et de colonne sont multiplexées temporellement pour la mémoire DRAM. Avec N broches, on détermine 2^2N adresses.

Q4.5 Les chronographes suivants sont-ils des chronographes de lecture ou d'écriture de la mémoire? Expliquez pourquoi?

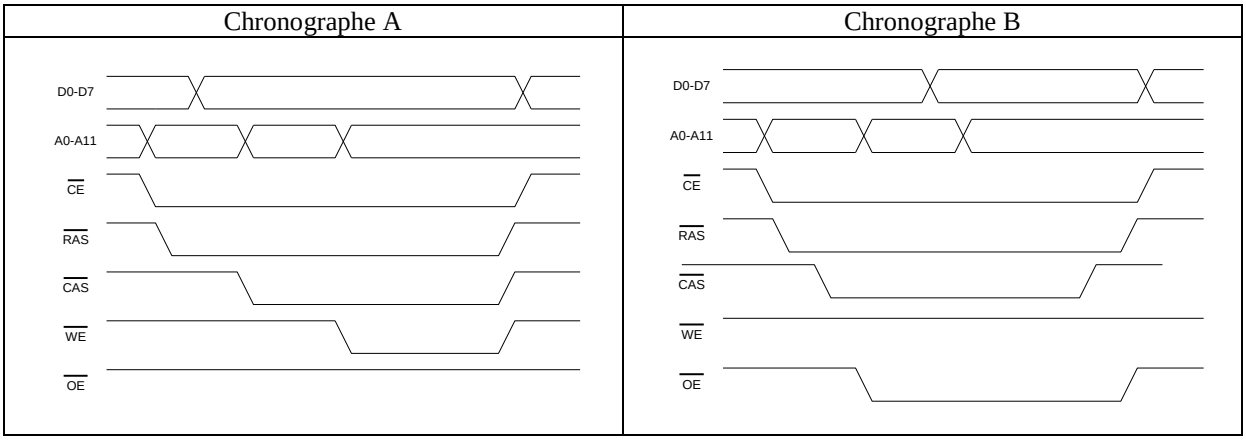


Figure 1 - Accès à la mémoire DRAM

a) Il s'agit d'une écriture de la mémoire: la ligne nWE est activée et les données sont changées par le CPU avant que l'adresse de la mémoire ne soit donnée complètement (avant l'activation de nCAS)
b) Il s'agit d'une lecture de la mémoire: la ligne nOE est activée et les données sont changées par la mémoire après que l'adresse de la mémoire soit donnée complètement (après l'activation de nCAS).

Q4.6 Parmi ces mémoires non-volatiles (FLASH, EEPROM et FRAM), indiquez celle que vous utiliseriez dans les applications suivantes (la meilleure mémoire est celle qui coûte le moins cher, vous avez une quantité illimitée de mémoire non-volatile!). Indiquez pourquoi.

1. ***Vous mesurez la température à toutes les 100ms et vous devez sauvegarder toutes les températures lues dans le dernier 48 heures dans une mémoire non-volatiles. Pour chaque échantillon, vous avez 8 octets de données (la température et le temps).***
2. ***Votre système embarqué affiche une page web avec plusieurs images et des informations qui varieront peu, mais qui seront lues fréquemment.***
3. ***Vous échantillonnez du son à 20kHz – 2 octets par échantillons et vous voulez sauvegarder plusieurs bandes sonores de 2 minutes maximum pour les relire ensuite. Votre système a assez de mémoire RAM pour stocker 8Mo d'information.***

Dans le cas 1, j'utiliserais de la mémoire EEPROM. Il s'agit de la mémoire idéale pour emmagasiner de petits échantillons qui changent fréquemment. Par ailleurs, le temps d'écriture de la EEPROM est < à 100ms.
Dans le cas 2, j'utiliserais de la mémoire FLASH : il y a peu d'écritures et c'est le choix le plus économique.
Dans le cas 3, j'utiliserais encore de la mémoire FLASH. Les données sont mises temporairement en RAM, puis copiées en FLASH…

Q4.7 Le MMU et le contrôleur de DMA sont étroitement reliés ensemble. Expliquez pourquoi.

Le contrôleur de DMA prend le contrôle du bus système pour accéder à la mémoire, or c'est le MMU qui est l'interface entre le microprocesseur et le bus système…

Q4.8 Comment se fait le DMA dans LM3S9B92? Quelles sont les contraintes d'accès à la mémoire de données par le cœur du microcontrôleur lors d'un transfert par DMA?

Voir la Datasheet du microcontrôleur pour la première partie de la question. On retrouve également la réponse suivante à la deuxième question dans la datasheet: "The µDMA controller's usage of the bus is always subordinate to the processor core, so it never holds up a bus transaction by the processor. Because the µDMA controller is only using otherwise-idle bus cycles, the data transfer bandwidth it provides is essentially free, with no impact on the rest of the system."

Q4.9 Quelle est la différence principale entre les mémoires DRAM et FRAM?

La mémoire FRAM est similaire, en construction à de la mémoire DRAM. Touteefois, la couche diélectrique de la cellule DRAM est remplacée par une couche ferroélectrique, ce qui la rend non-volatile.

Q4.10 Il faut interfacer une mémoire de 1 méga-octet sur un bus d'adresses 32 bits (on assume le bus A31-A0 avec A31 MSB). La première adresse de la mémoire doit être 0x10E08000. Quelle est la fonction logique permettant d'activer la mémoire en mettant le bit 'ENABLE' de la mémoire à '0' à partir des lignes du bus d'adresse?

R : NOTE : Plusieurs réponses possibles.
On a 1024*1024 = 1 048 576 octets = 0x100000 en hexadécimal.
La plage d'adresses est de 0x10E08000 à 0x10F07FFF, donc :
A31.....A0
0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
à
0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Donc, A31 = 0; A30 = 0; A29 = 0; A28 = 1; A27 = 0; A26 = 0; A25 = 0; A24 = 0; A23 = 1; A22 = 1 A21 = 1 sont des conditions respectées en tout temps. On ne doit plus être à 0 lorsque A20 = 1 et A15 = 1, donc à partir de 0x10F08000.

D'où E = A31 + A30 + A29 + A27 + A26 + A25 + A24 + !(A28·A23·A22·A21) + (A20·A15) + (A20·(A15 + A16 + A17 + A18 + A19))

Q4.11 Combien de temps un bit de mémoire FLASH garde-t-il sa valeur lorsque la mémoire n'est pas alimentée?

Habituellement, les manufacturiers spécifient 20 ans : la décharge de la grille flottante à travers le diélectrique non conducteur se fait très lentement.

Q4.12 Le contenu des mémoires non-volatiles est habituellement protégé par un CRC ou un checksum. Pourquoi? Quels évènements peuvent corrompre les données ou instructions contenues dans ces mémoires?

Un ESD pendant la lecture ou une faute d'alimentation pendant l'écriture des données peuvent corrompre le contenu de la mémoire.

Cours 5 : Périphériques, Général + GPIO

Q5.1 Lorsque deux composantes opèrent à des vitesses différentes, il faut un tampon de données et des drapeaux d'état pour gérer la communication entre les deux composantes. Expliquez à l'aide d'un exemple.

Supposons un microprocesseur qui peut lire des instructions à toutes les millisecondes et un disque dur qui peut produire une instruction à toutes les secondes en moyenne, mais qui peut prendre jusqu'à 5 secondes pour produire une instruction. Dans le meilleur cas, le disque dur produit une instruction en 2 milliseconds, mais c'est un cas très rare.

Si le microprocesseur lit les instructions du disque dur à toutes les millisecondes, il exécutera chaque instruction mille fois, ce qui peut occasionner des erreurs. Il faut un drapeau pour indiquer au microprocesseur que la prochaine instruction est prête.

Il est possible que le microprocesseur exécute d'autres actions pendant que le disque dur gratte. Si c'est le cas et si une autre action cette autre action dure plus de quatre millisecondes, il est possible que le disque produise 2 instructions pendant que le microprocesseur exécute une autre action. Dans ce cas, la première instruction produite par le disque dur sera perdue à moins qu'il n'y ait un tampon.

Q5.2 Quelles adresses faudra-t-il lire et/ou écrire afin de mettre un "1" sur la broche 10 du microcontrôleur LM3S9B92?

La broche 10 du LM3S9B92 est PD0. Comme la fonction de la broche 10 est GPIO par défaut, le rôle de la broche n'a pas à être changer (GPIOAFSEL). Il faut configurer la broche en sortie et écrire un "1" comme valeur de sortie.

Selon la datasheet, l'adresse du registre de direction du port D (GPIODIR) est 0x4000.7000 + 0x400 (APB) ou 0x4005.B000 + 0x400 (AHB) en fonction du bus choisit pour écrire le registre de direction. Comme il s'agit du bit 0 du port D, il faut mettre le bit 0x01 à 1 pour ce registre.

Selon la datasheet, l'adresse du registre de valeur du port D est 0x4000.7000 + 0x000 (APB) ou 0x4005.B000 + 0x000 (AHB) en fonction du bus choisit pour écrire le registre de valeur. Comme il s'agit du bit 0 du port D, il faut mettre le bit 0x01 à 1 pour ce registre.

Finalement, il faut activer PD0 en écrivant GPIODEN (GPIO Digital Enable). L'adresse du registre d'activation du port D est 0x4000.7000 + 0x51C (APB) ou 0x4005.B000 + 0x51C (AHB) en fonction du bus choisit pour écrire le registre d'activation. Comme il s'agit du bit 0 du port D, il faut mettre le bit 0x01 à 1 pour ce registre.

Q5.3 Dites pourquoi il y a habituellement un microcontrôleur juste pour contrôler un LCD dans un système microprocesseur et interfaces.

Un LCD a beaucoup de broches… il faut un microcontrôleur simple (peu dispendieux) capable de gérér une matrice de points énorme.

Q5.4 Expliquer comment fonctionne l'interface entre un contrôleur de LCD comme le HD44780 et un autre microprocesseur (le LCD du laboratoire 3 a un contrôleur HD44780).

Le microprocesseur connecté au HD44780 envoie des commandes par une interface parallèle au HD44780. Ces commandes régissent l'affichage : configurer le LCD, effacer le LCD, écrire un caractère à la position du curseur, déplacer le curseur, etc.

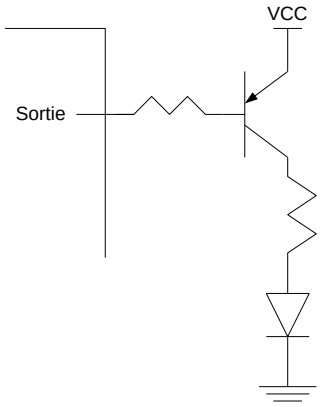
*Q5.5 Combien de broches de microprocesseur sont nécessaires afin de contrôler un clavier ayant 72 touches? Pourrions-nous relier chaque broche à une touche? Si on assume que le clavier est une matrice 8*9, serait-il possible de lire le clavier avec 14 broches? Comment?*

Si le clavier est une matrice 8*9, il faudra 17 broches. Il est possible dans ce cas, d'utiliser 14 broches si on change le rôle des broches dans le temps : 9 broches servent pour identifier les colonnes, 4 broches servent à lire les rangées 0 à 3 ou 4 à 7 en fonction de la 14^{e} broche…

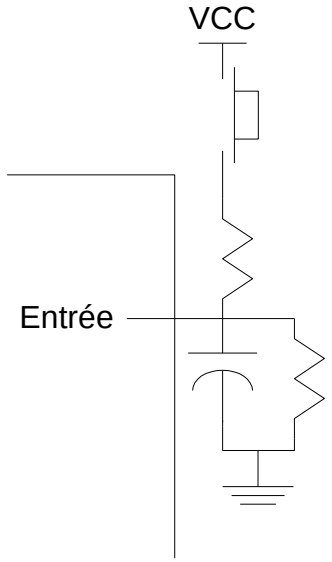
Q5.6 Illustrez une sortie totem pole avec des transistors (BJT ou CMOS au choix) et illustrez une sortie open drain (ou open collector, au choix).

L'inverseur (NOT) illustré dans le fichier "Introduction-Composantes analogiques.ppt » est une sortie totem pole. Une sortie open drain est un NMOS dont la grille est relié au contrôle et la source, au ground.

Q5.7 Dessinez le circuit pour qu'une LED nécessitant 10mA soit allumée avec un "0" lorsque la sortie du microprocesseur ne peut fournir que 4mA.



Q5.8 On veut lire un "1" lorsqu'un bouton est appuyé et un « 0 » sinon. Si le contact du bouton oscille énormément et si un debounce matériel est requis, dessinez le circuit permettant de lire ce bouton avec l'entrée digitale d'un microcontrôleur.



Q5.9 Quel problème peut survenir s'il n'y a pas de résistances de pull-up ou de pull-down sur les lignes de lecture d'un clavier?

La ligne peut ainsi être en état de haute impédance, ce qui implique que s'il est lu, les données reçues peuvent être invalides, étant donné que cet état correspond à du bruit.

Q5.10 Lors de l'interfaçage d'un LCD, dans quel cas utilise-t-on l'instruction de lecture du busy flag plutôt que d'attendre un délai défini?

R : Si le programme doit s'exécuter dans un temps très critique le plus rapidement possible, il est plus efficace d'attendre que le périphérique soit prêt plutôt que d'attendre un délai qui pourrait être beaucoup plus long que la période où le périphérique est occupé.

Q5.11 Nommer les trois sortes de registres qu'on retrouve généralement permettant de contrôler les GPIO et à quoi sert chacun?

Direction, valeur, fonction.

Q5.12 Pourquoi le registre de direction est toujours configuré comme des entrées à un reset?
Cela permet d'éviter des collisions de données, par exemple si un périphérique est actif sur cette ligne et tente d'écrire '0' ou '1'.

Q5.13 Pour quelle raison un condensateur peut être mis en parallèle avec une entrée (connecté entre l'entrée et la masse)?
Cela permet de réduire les oscillations et autres instabilités du signal et diminue donc le risque d'une lecture erronée.

Q5.14 Votre ami arrive avec un petit écran tactile et vous demande si son écran tactile est capacitif ou résistif. Comment ferez-vous pour lui donner une réponse juste?
Si vous mettez des gants et que l'écran ne fonctionne plus, vous avez un écran capacitif.

Q5.15 Pourquoi utilise-t-on souvent deux circuits intégrés différents afin de contrôler les gros LCDs TFT (Thin-film-transistor liquid-crystal display)?
Le nombre de pixel à gérer devient trop grand.

Un circuit intégré/microcontrôleur gère l'image à afficher. Il prend une copie de l'image en RAM et gère les crystaux liquides pour afficher cette image.

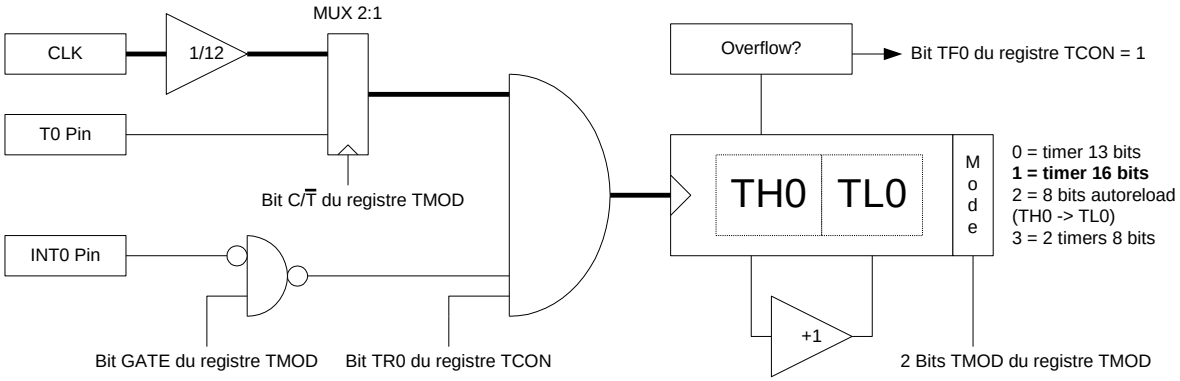
L'autre circuit intégré donne du support pour modifier l'image avec des instructions simples (déformation d'image, filtres simples, scrolling, etc.), tracer des formes automatiquement ou afficher des caractères avec diverses fontes.

Ainsi, le microcontrôleur principal n'a qu'à demander d'afficher « Hello Word » dans un rectangle à telle position pour que cela s'affiche, sans devoir gérer

Q5.16 Quelles sont les différences entre une interface 8080 et 6800 pour le contrôle d'un LCD avec un bus parallèle?
Plutôt que d'avoir WR et RD strobe comme le 8080, l'interface 6800 implémente une ligne Strobe et une ligne R/W…

Cours 6: Périphériques, PWM, Timer, ADC et DAC

Q6.1 Le site web suivant détaille le fonctionnement des timers pour le 8051 d'Intel (le cœur de microprocesseur le plus vendu sur le marché mondial!) : http://www.mikroe.com/eng/chapters/view/65/chapter-2-8051-microcontroller-architecture/. Expliquez le fonctionnement du timer 0 du 8051.



Q6.2 Quels registres du LM3S9B92 doit-on écrire pour que le timer 0 déclenche une interruption à toutes les millisecondes? Quelles valeurs doit-on donner à ces registres si l'horloge du coeur fonctionne à 32MHz.

Supposons que l'on opère le timer 0 en mode 16 bits, périodique et que l'on utilise la partie A pour compter le temps, exécuter la section 11.4.3 (16-Bit One-Shot-Periodic Timer Mode) de la datasheet permet d'obtenir la configuration du timer désirée.

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the GPTM Configuration Register (GPTMCFG) with a value of 0x0000.0004.
3. Set the TnMR field in the GPTM Timer Mode (GPTMTnMR) register:
a. Write a value of 0x1 for One-Shot mode.
b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the TnSNAPS, TnWOT, TnMTE and TnCDIR bits in the GPTMTnMR register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale Register (GPTMTnPR).
6. Load the start value into the GPTM Timer Interval Load Register (GPTMTnILR).
7. If interrupts are required, set the appropriate bit in the GPTM Interrupt Mask Register (GPTMIMR).
8. Set the TnEN bit in the GPTM Control Register (GPTMCTL) to enable the timer and start counting.
General-Purpose Timers
9. Poll the GPTMRIS register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 the appropriate bit of the GPTM Interrupt Clear Register (GPTMICR).
If the TnMIE bit in the GPTMTnMR register is set, the RTCRIS bit in the GPTMRIS register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

Pour avoir un compte à toutes les 1ms, il y a plusieurs possibilités. On pourrait, par exemple, mettre GPTMTAPR à 31, ce qui donnerait un compte de timer à toutes les us. Ensuite, il suffirait de reloader le timer à tous les 1000 comptes…

Q6.3 Énumérez quatre applications dans lesquelles on retrouve un PWM.

Moteurs.
Communications.
Power supplies.
Gradation de puissance fournie à une charge.

En fait, les PWMs sont présents dans énormément d'applications électroniques.

Q6.4 Peut-on convertir un signal de PWM en niveau DC dépendant des paramètres d'opérations du PWM? Si oui, comment? Si non, pourquoi? (Voir http://www.esorol.com/assets/files/PDFs/Spectral%20analysis%20of%20a%20PWM%20signal.pdf pour de l'aide!!!).

Oui, l'onde du PWM a un contenu DC et plusieurs harmoniques dont la fréquence dépend qui dépendent de la fréquence du PWM. Pour faire une sortie DC, il faut générer une onde PWM haute fréquence et filtrer tout ce qui n'est pas DC. Le contenu DC sera directement proportionnel au duty cycle.

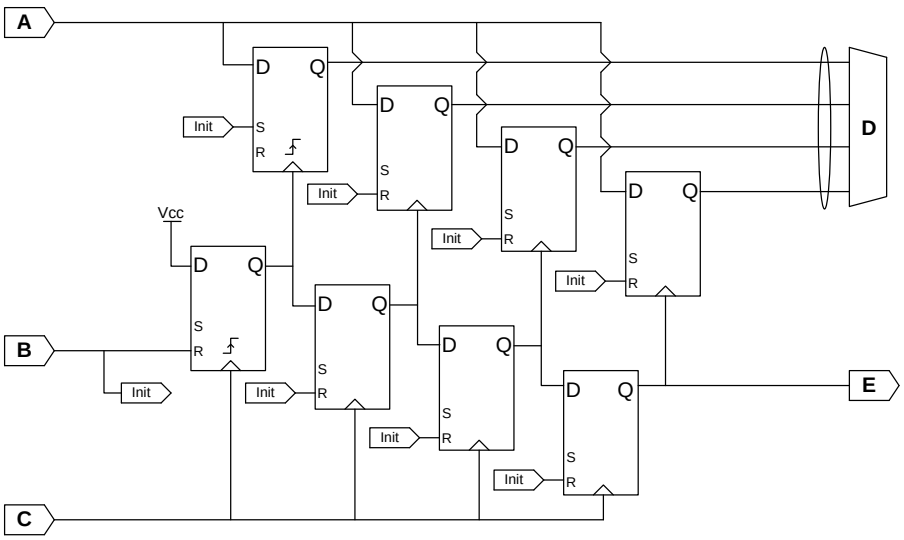
Q6.5 Peut-on produire un sinus avec un PWM? Comment?
Oui, avec un filtre passe-bande.

Q6.6 Donnez un exemple d'application où la largeur des pulses d'un PWM est modulée.
Encodage de signal avec largeur de pulse (PWM!). Par exemple un pulse court pourrait être un 0 et un pulse long, un 1.

Q6.7 Expliquez comment fonctionne un DAC R-2R.
Une tension de référence est divisée par 2, par 4, par 8, par 16, par 2^n avec le réseau de résistances. Toutes les tensions résultantes sont additionnées ou ne sont pas additionnées en fonction des bits du mot digital à convertir.

Q6.8 Expliquez comment fonctionne un ADC par approximation successives?
Un DAC produit une tension qui est comparée avec la tension analogique à convertir. À toutes les itérations, la tension de sortie du DAC est ajustée en fonction de la sortie du comparateur jusqu'à ce que la tension à convertir soit près de la sortie du DAC. La conversion est finie à ce moment : l'entrée du DAC est la valeur convertie.

Q6.9 La figure suivante illustre le « Successive Approximation Register (SAR) » d'un ADC 4-bits. Indiquez quels signaux internes de l'ADC sont branchés sur A, B, C, D et E?



A--- Sortie du Comparateur ayant pour entrées l'échantillon à convertir et la sortie du DAC.
B--- Signal débutant la conversion.
C--- Horloge interne de l'ADC

D--- Entrée digitale du DAC
E--- Signal indiquant la fin de la conversion

Q6.10 Expliquez ce qui limite la vitesse d'échantillonnage et la résolution des DACs/ADCs. Utilisez le DAC R-2R et/ou l'ADC par approximations successives au besoin.

La vitesse d'échantillonnage des DACs et des ADCs est toujours limitées par la vitesse de propagation des signaux dans leurs composantes électroniques. Plus il y a de portes entre l'entrée et la sortie, plus le temps de conversion est long. La vitesse d'échantillonnage est aussi souvent limitée par la méthode de conversion : dans les approximations successives, par exemple, on compare N valeurs pour un ADC N bits…

La résolution est toujours limitée par la résolution de la tension de référence. Elle est aussi limitée par la résolution des composantes internes de l'ADC ou du DAC. Dans le DAC R2-R par exemple, si la Résistance a une résolution de 1% , la résolution du DAC sera 2%, au mieux.

Q6.11 Pourquoi devrait-on toujours utiliser un filtre passe-bas entre le signal échantillonné et l'entrée d'un ADC lorsqu'on interface avec un ADC?

Pour éliminer le recouvrement spectral. Chercher fréquence de Nyquist sur Google pour plus d'info… ou relire vos manuels de traitement de signal!

Q6.12 Décrivez les différentes manières d'utiliser un Timer?

- Libre (free run) : Timer incrémenté ou décrémenté continuellement même s'il déborde
- Un coup (one shot) : Timer qui incrémente ou décrémente pendant une période définie et qui génère par la suite une interruption.
- Générateur d'horloge : (periodic, auto-reload) : Timer qui génère des interruptions à un intervalle régulier, définie par une période d'incrémentation ou de décrémentation.

Q6.13 Un timer 16 bits en mode auto-reload incrémente à tous les coups de l'horloge du timer. Sachant que l'horloge du système est 16MHz, sachant qu'il est possible de diviser l'horloge du système par 4, 8 et 16 afin de générer l'horloge du timer, quelle doit être la valeur de chargement automatique (auto-reload) du timer et quelle doit être l'horloge du timer pour générer une interruption à toutes les millisecondes. S'il existe plusieurs réponses possibles, donnez-les toutes!

Si l'horloge du timer vaut 1MHz (16MHz/16), il faut 1000 comptes pour faire 1ms. Donc, la valeur de chargement automatique de l'horloge devrait être 65536 (un timer 16 bits!) – 1000, soit 64536. Ainsi, le timer débordera (et génèrera une interruption) après avoir été incrémenté de 1000 comptes.

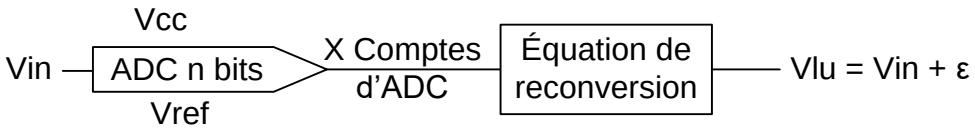
Horloge = 1MHz –> Auto-reload = 64536 -> interruption 1ms
Horloge = 2MHz –> Auto-reload = 63536 -> interruption 1ms
Horloge = 4MHz –> Auto-reload = 61536 -> interruption 1ms

Q6.14 Vous voulez utiliser un timer pour générer une interruption périodique. Malheureusement, ce timer n'a pas de mode auto-reload. Expliquez comment vous pourriez générer une interruption périodique en dépit de l'absence de l'auto-reload.

Lorsque l'interruption du timer se produit, il s'agit de recharger manuellement le compte du timer.

Pour avoir une plus grande précision sur la fréquence des interruptions, la valeur de recharge manuel du timer devrait tenir compte du temps pris pour entrer dans l'interruption et du temps nécessaire pour recharger le timer dans l'interruption. De plus, l'interruption du timer devrait avoir la plus haute priorité du système.

Q6.15 On retrouve, dans la littérature, plusieurs équations pour calculer la valeur mesurée par un ADC à partir de la lecture de l'ADC :



Parmi les équations de conversion suivantes, laquelle donnera, en moyenne, l'erreur de conversion la plus petite si l'entrée est totalement aléatoire?

a) $Vlu = \frac{X\ counts}{2^n\ counts} * Vref$

b) $Vlu = \frac{X\ counts}{(2^n - 1)\ counts} * Vref$

c) $Vlu = \frac{(X + 0.5)\ counts}{2^n\ counts} * Vref$

d) Aucune de ces réponses

L'équation c) est la meilleure. Voir l'annexe A pour plus de détails!

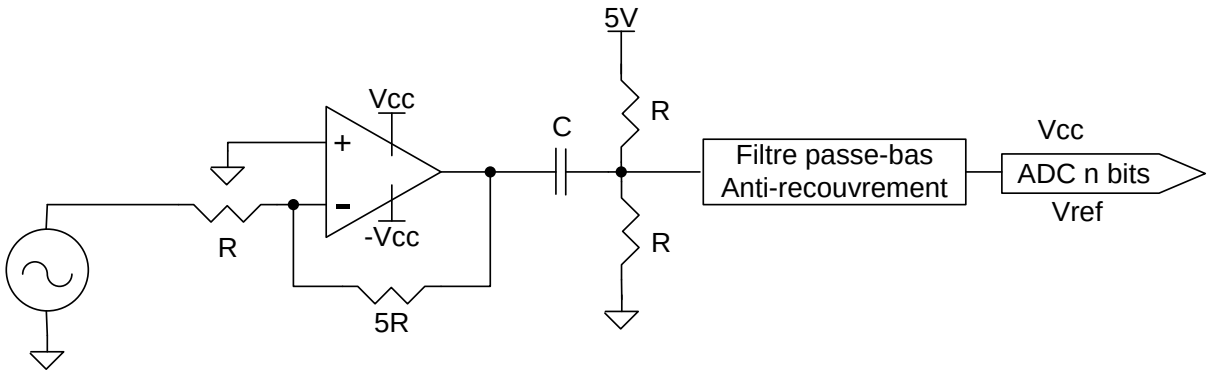
Q6.16 Lorsqu'un ADC 10 bits mesure 513 (valeur binaire retournée par l'ADC), quelle est la tension analogique à l'entrée de l'ADC, si on considère une référence de 2.5V et une alimentation de 3.3V?

On estimer l'entrée de l'ADC à $2.5V*(513+0.5)/2^{10}=1.2537V$

Q6.17 Si on met 1.25V à l'entrée d'un ADC 12 bits ayant un voltage de référence à 3.3V : quelle valeur digitale donnera l'ADC (combien de comptes)? Lorsque le nombre de compte d'ADC sera reconvertit en valeur analogique pour calculer l'entrée, quelle sera l'erreur de mesure en fonction de la méthode de calcul (appliquer les trois équations de conversion énumérée précédemment)?

Nombre de compte = arrondi inf$(1.25/3.3*2^{12})$ = 1551
Valeur d'entrée estimée = 1551.5/4096*3.3V = 1.25V

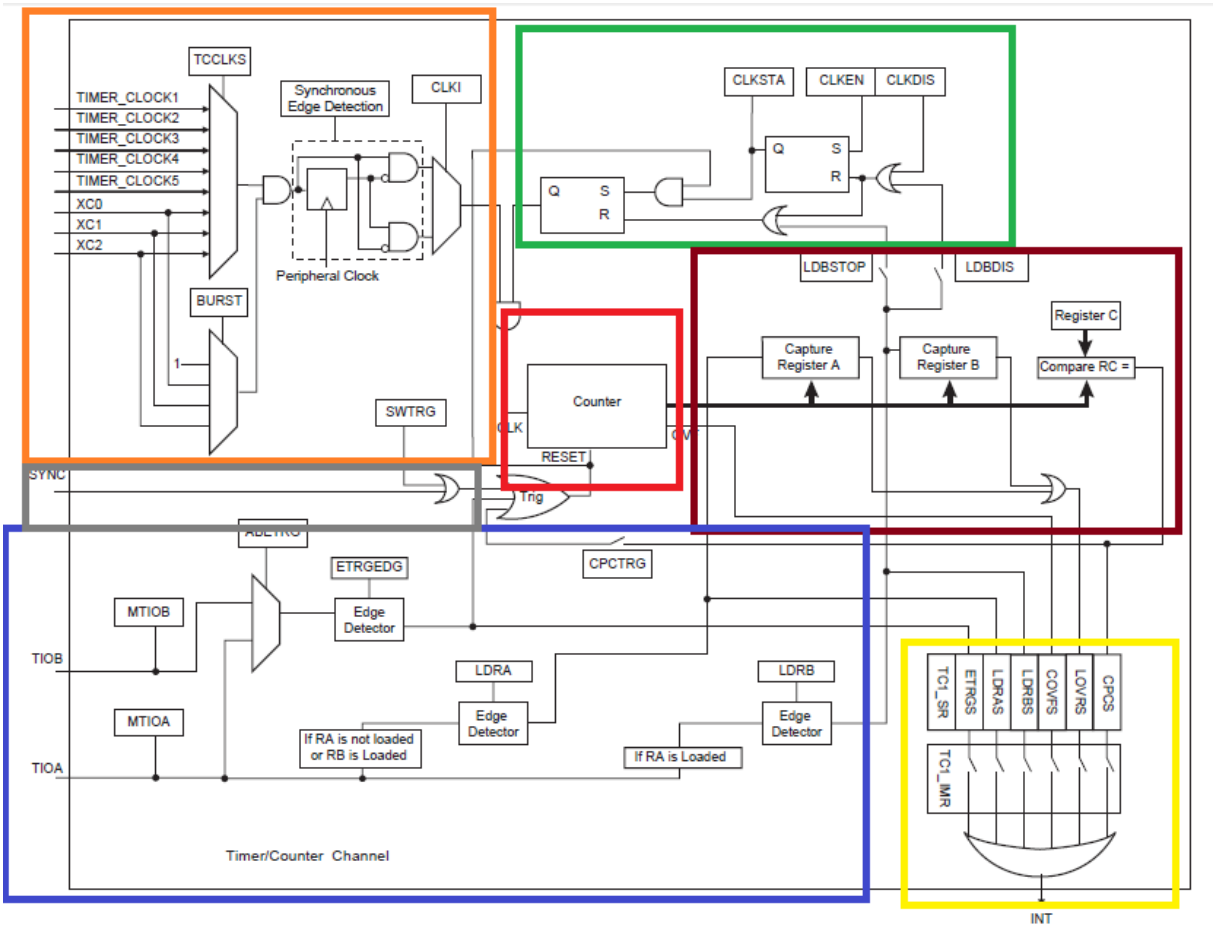
Q6.18 Un signal analogique haute-fréquence varie de -0.5V à 0.5V. Quel circuit électrique devrait être utilisé pour lire ce signal avec un ADC lisant de 0V à 5V (référence à 5V)?



Q6.19 Un DAC 8 bits a une tension de référence à 5V. Si on veut sortir 1.66V du DAC, comment de comptes doit-on mettre à l'entrée du DAC? Quelle est l'erreur par rapport à la sortie désirée?

Nombre de comptes = arrondi au plus près $(1,66/5*2^8)=85$
La sortie sera 85/256*5V = 1.66V

Q6.20 La page 859 de http://www.atmel.com/images/atmel-11100-32-bit%20cortex-m4-microcontroller-sam4s_datasheet.pdf illustre le fonctionnement des timers su SAM4S, un microcontrôleur similaire à celui des laboratoires du cours. Détaillez le rôle de chaque section du contrôleur de timer qu'on retrouve sur la figure de cette page.



En haut à gauche, en orange : Sélection de l'horloge du timer et filtre sur les transitions
En haut à droite, en vert : Section d'activation/désactivation du timer
Au Centre, en rouge : Le timer lui-même
À gauche, en gris : signal de synchronisation du timer avec les autres timers
À droite, en marron : registres de capture et de comparaison du timer
En bas à gauche, en bleu : utilisation d'entrées externes pour contrôler le compteur (gate, reset) et la capture
En bas à droite, en jaune : Évènements qui cause des interruptions de timer

Cours 7: Périphériques, Interfaces Séries

Q7.1 Quelle est la différence entre UART et RS232?

UART est Universal Asynchronuous

Q7.2 Quelle interface série est utilisée dans chaque application suivante :

a) Cartes SD?
b) Programmation JTAG?
c) Circuit de contrôle des jets d'eau et alarmes d'incendies dans un immeuble à 30 étages?
d) Mémoires séries?
e) ADC/DAC externes?
f) Système de communication dans un autobus.
g) Système de surveillance dans une usine

a) I2C principalement.
b) Le JTAG est très près du SPI.
c) Le RS485 semble le meilleur choix : robuste, peut communiquer sur de longues distances.
d) SPI ou I2C : SPI si une très grande vitesse est requise, I2C si on veut utiliser deux broches du micro seulement.
e) Habituellement SPI parce que les vitesses d'échantillonnage des ADC/DAC sont autour de 1MHz.
f) CAN : très robuste.
g) Le RS485 semble le meilleur choix : robuste, peut communiquer sur de longues distances.

Q7.3 Pour les quatre interfaces séries suivantes, donnez les caractéristiques générales :

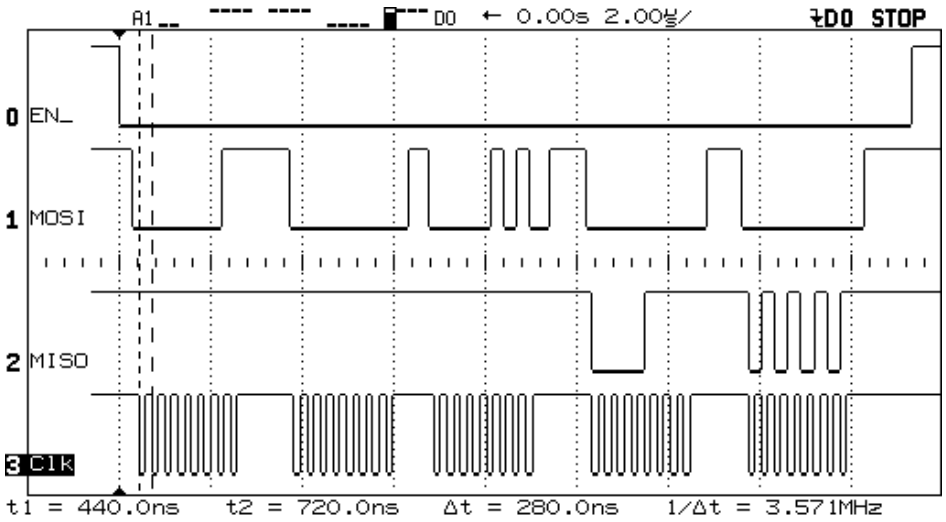
Caractéristique	***RS232***	***RS485***	***SPI***	***I2C***
Architeure physique du réseau	Point-à-point	Multipoint Toute topologie sans boucles	Multipoint	Multipoint
Architecture logique du réseau	DTE-DCE	Non défini, habituellement master slave	Master-Slave	Master-Slave, plusieurs masters supportés
Sens de la communication	FULL-DUPLEX	FULL-DUPLEX ou HALF-DUPLEX	FULL-DUPLEX	HALF-DUPLEX
Synchrone ou Asynchrone	Asynchrone	Asynchrone	Synchrone	Synchrone
Vitesse max (bps)	~115kHz[2]	~115kHz[2]	~5MHz[2]	~500kHz[2]
Distance max (m)	10 mètres ou plus[3]	100 mètres ou plus[3]	~30 centimètres[3]	~30 centimètres[3]
Nombre de fils minimum	2[4]	2 (HALF-DUPLEX)[4]	4[4]	2[4]
Type de sortie requise pour écrire un bit	Totem-pole	Totem-pole avec tri-state	Totem-Pole	Open-Collector
Matériel de support requis	MAX232 ou équivalent	MAX485 ou équivalent Résistances de fin de ligne au besoin	Résistances de fin de ligne au besoin	Pull-ups et résistances de fin de ligne au besoin
Décennie de création	1960-1970	1980-1990	1980-1990	1980-1990

[2] Dépend de la longueur des fils et du bruit
[3] Dépend de la qualité/nature des fils et de la vitesse de communication
[4] Le ground ou la référence peut être requis également, en fonction de l'application.

Q7.4 Le RS485 utilise parfois 4 fils pour la communication, mais souvent 2 : il faut moins de fils! En contrepartie, la communication sur deux fils est HALF-DUPLEX, différentielle et bidirectionnelle. Expliquez pourquoi la plupart des systèmes de communication RS485 ont une architecture maître-esclaves en raison de ces caractéristiques.

Sur lien de communication half-duplex, un seul appareil de communication peut signaler des informations sur le bus à un moment donné. La façon la plus facile d'arbitrer l'accès au bus est d'avoir un maître de bus qui initie et contrôle toutes les communications.

Q7.5 La figure suivante illustre une communication SPI entre un maître et son esclave. Répondez aux questions reliées à la figure :



A. Qui génère les différents signaux de communication? Le maître ou l'esclave?

MOSI, EN_ et CLK sont contrôlés par le maître.
MISO est contrôlé par l'esclave.

B. Combien d'octets sont transmis et reçus lors de la communication? Quelles sont les valeurs de ces octets?

Octet (dans l'ordre de transmission)	Transmis	Reçu
1	0x03	0xFF
2	0x00	0xFF
3	0x0A	0xFF
4	0x00	0x0F
5	0x00	0x55

C. À quelle fréquence sont transmis les bits lors de cette communication SPI? Quelle est la vitesse de communication effective?

La fréquence des bits est de 3.571MHz. Cependant, il y a des pauses entre chaque octet et il faut environ 16 us pour transmettre 40 bits. Donc, la vitesse de communication est d'environ 40bits/16us, soit environ 2.5 Mbps.

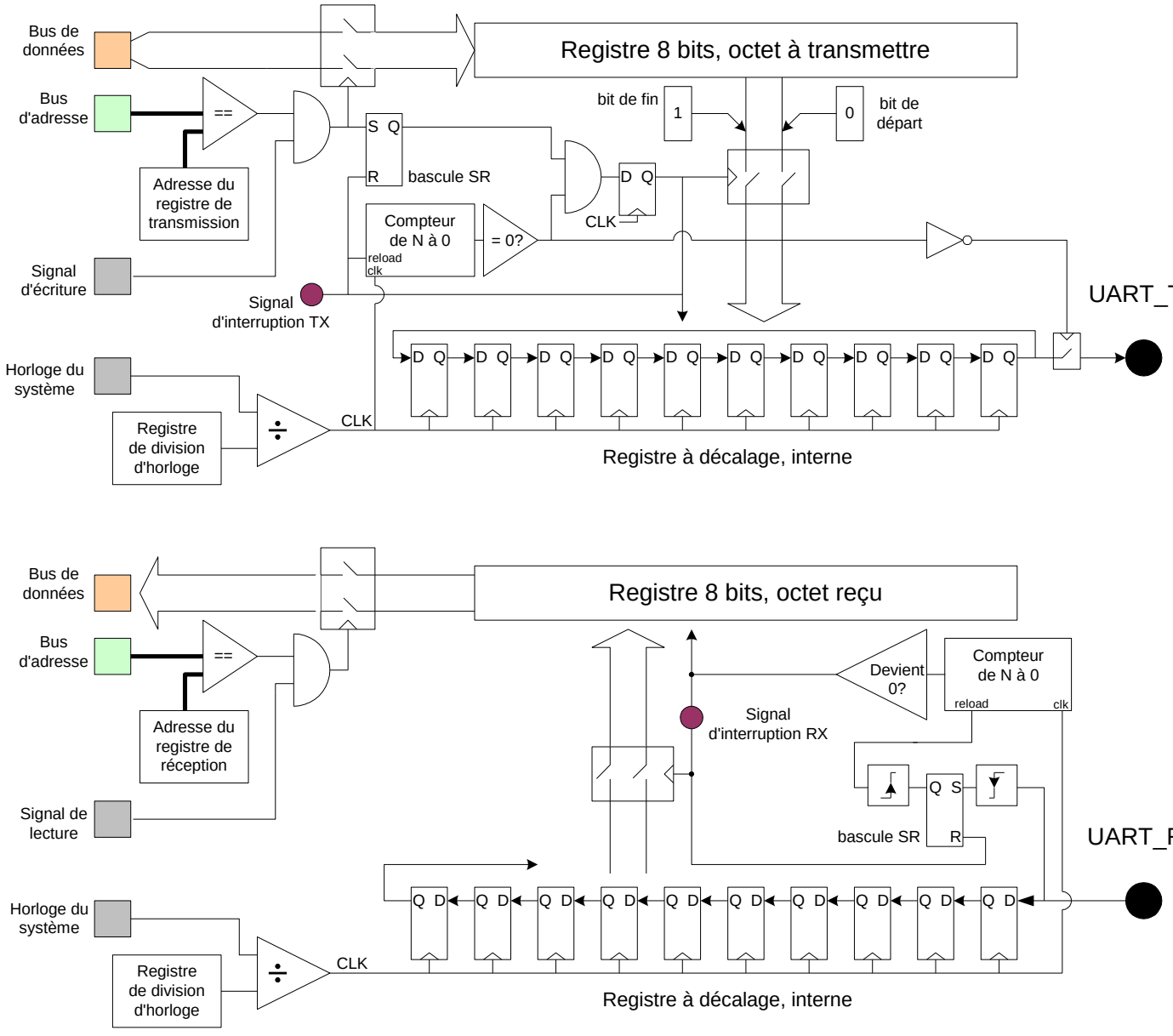
Chaque octet est transmis par le module d'E/S du SPI. Les pauses entre les transmissions sont causées par le délai de communication entre le module d'E/S, le processeur et les instructions lues/exécutées afin de gérer le module d'E/S.

D. Quelles sont la phase et la polarité du signal SPI?

La phase : les données sont mises sur la ligne de donnée avant la première transition du signal d'horloge.
La polarité : l'horloge commence HIGH.

Avec cette phase et cette polarité, les données sont échantillonnées/lues lors de descente d'horloge et mises-à-jour lors de montées d'horloge.

Q7.6 La figure suivante illustre un module d'E/S pour un UART[5]. À partir de cette illustration de principe, expliquez le fonctionnement général de ce périphérique :



Transmission : La transmission se fait avec un registre d'octet à transmettre, un registre à décalage (shift register) contenant l'octet en cours de transmission et un compteur. Lorsqu'une instruction écrit le registre d'octet à transmettre, la valeur est automatiquement transférée dans le registre à décalage et le compteur commence à compter. À chaque coup de l'horloge, un bit est transmis sur la broche de UART_TX jusqu'à ce que tous les bits soient transmis. À ce moment, la transmission cesse. Elle recommencera quand un autre octet sera écrit dans le registre d'octet à transmettre.

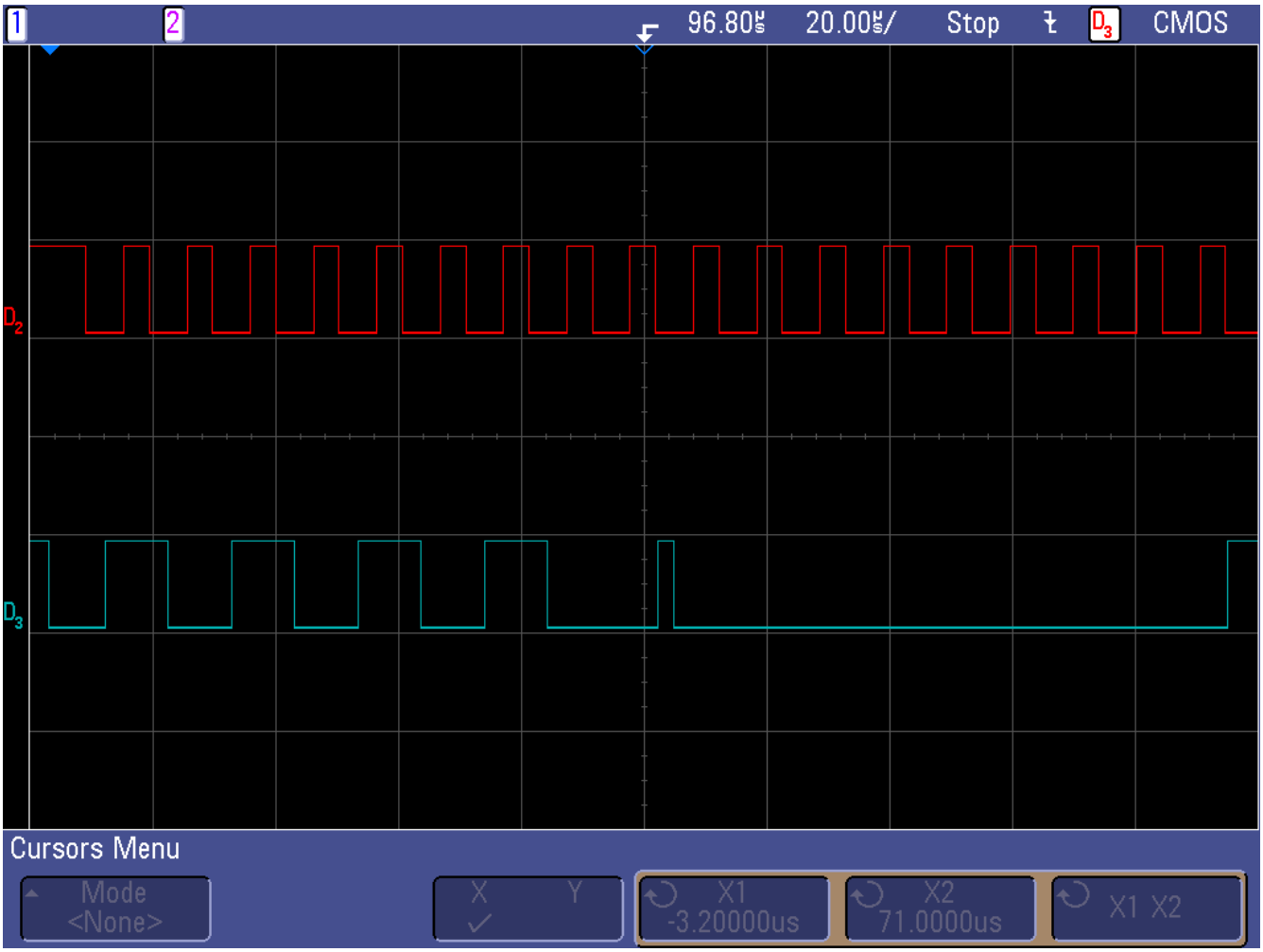
[5] Le module d'UART présenté dans ces exercices est fictif. Malgré quelques erreurs possibles (par exemple, le nombre de bits du registre à décalage dans un UART est souvent plus petit que 10 bits!) et malgré le nombre très limité de caractéristiques qu'il illustre, ce dessin permettra au lecteur de comprendre le fonctionnement global d'un UART.

Si un octet est écrit dans le registre à transmettre pendant la transmission, cet octet sera transféré dans le registre à décalage lorsque l'octet en cours de transmission sera fini. Ainsi, il est possible de transmettre en continu avec une intervention du CPU à tous les octets : il suffit d'écrire chaque octet à transmettre pendant la transmission de l'octet précédent. Habituellement, un signal d'interruption est disponible pour indiquer que les données du registre d'octet à transmettre ont été transférées dans le registre à décalage et qu'il est désormais possible de réécrire le registre de données à transmettre sans compromettre la transmission.

Réception : La réception se fait avec un registre d'octet reçu, un registre à décalage contenant le l'octet en cours de réception et un compteur. Lors d'une descente de voltage sur la broche UART_RX (les bits de départs sont des 0!!!), le module de réception commence à compter jusqu'à ce que tous les bits de l'octet à recevoir soit reçus. Les bits sont reçus à chaque coup d'horloge dans le registre à décalage. Ils sont transférés dans le registre d'octet reçu quand le compte de bits se termine.

Habituellement, transférer un octet reçu du registre à décalage vers le registre d'octet reçu cause une interruption si le contrôleur d'interruption est configuré à cette fin. La valeur contenue dans le registre d'octet reçu ne changera pas tant qu'un nouvel octet complet n'est pas reçu.

Q7.7 La capture d'oscilloscope suivante représente le début d'une communication I²C (le signal d'horloge commence au début du chronogramme). Le signal D2 est l'horloge (SCL) et le signal D3 le canal de données (SDA). Vu horizontalement, la première moitié du signal contient l'octet de contrôle, et la seconde moitié le début des données.



a) Est-ce le maître ou l'esclave qui initie la communication? Comment?

Réponse : C'est toujours le maître qui initie la communication par l'envoi d'un START bit.

b) Quelle est l'adresse du périphérique visé par la requête?
Réponse : 0x55 (attention au start bit : l'adresse fait 7 bits). Visuellement, il peut être plus facile de partir de la fin de l'octet de contrôle (juste avant le bit de read/write) et de lire l'adresse à l'envers.

c) Considérant la taille de cette adresse, combien de périphériques peuvent-ils être adressés sur un bus I²C?
Puisque la taille de l'adresse est de 7 bits, alors 128 périphériques peuvent en théorie être branchés sur le même bus. En pratique, le nombre de périphériques est limité par la capacité totale tolérée sur le bus (400 pF selon l'implémentation de référence).

d) Après l'envoi du premier octet de synchronisation, est-ce que c'est le maître ou l'esclave qui écrit sur le bus?
Le bit de r/w est à 0 (ce bit est envoyé tout de suite après l'adresse), donc c'est le maître qui écrira le/les prochains octets.

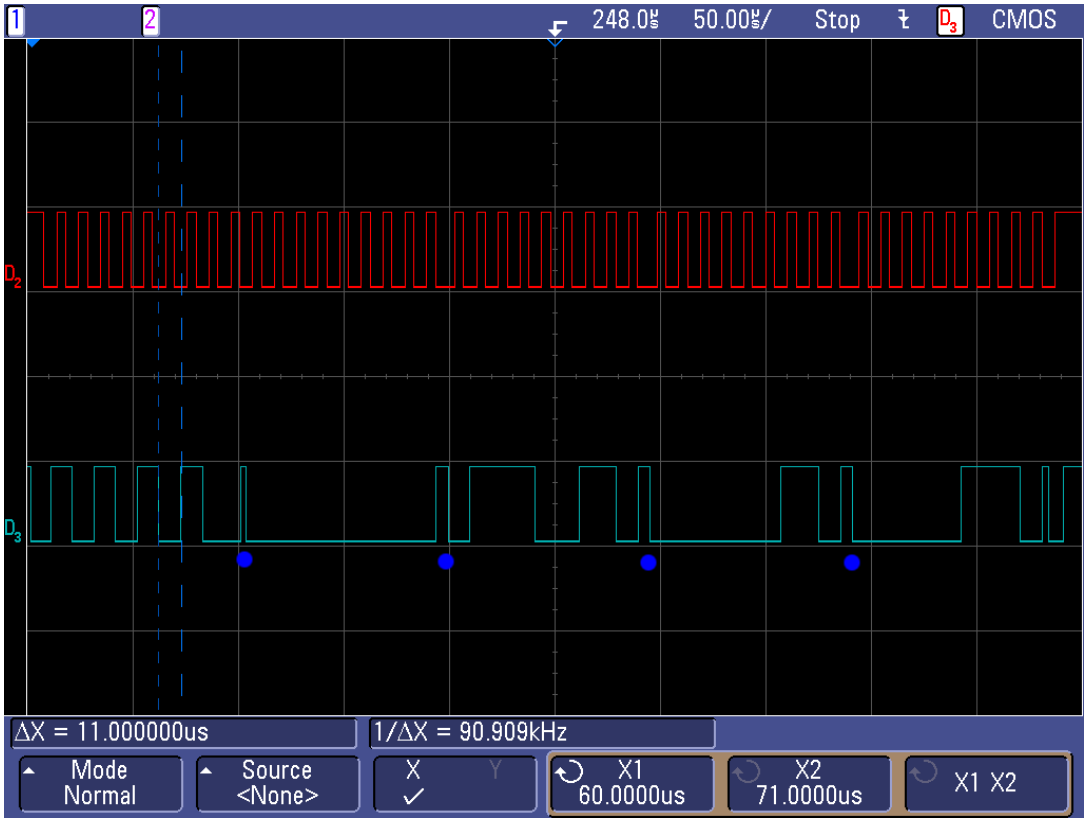
Q7.8 Pour lire un octet sur une mémoire EEPROM en I²C, le protocole est le suivant :
- ***Envoi d'un octet de contrôle pour sélectionner l'esclave***
- ***Envoi de l'adresse (2 octets)***
- ***Envoi d'un autre octet de contrôle sélectionnant le même esclave***
- ***Lecture des données à partir de l'esclave***

a) Pourquoi l'octet de contrôle est-il envoyé deux fois alors que le canal de communication ne change pas (e.g. c'est le même esclave qui reste activé)?
Réponse : parce que la direction de la communication doit être inversée (le maître doit d'abord écrire sur le bus pour envoyer l'adresse, puis c'est l'esclave qui doit envoyer les données provenant de l'adresse indiquée). En I²C, le seul moyen est de renvoyer un octet de contrôle pour changer la valeur du bit r/w, même si l'adresse reste la même.

b) Pour chacun de ces octets envoyés, qui génèrera le bit de ACK? Qui génèrera l'horloge?
Réponse :
Pour le bit de ACK :
Octet 1 : Esclave (octet de contrôle)
Octet 2 : Esclave
Octet 3 : Esclave (seconde partie de l'adresse)
Octet 4 : Esclave (nouvel octet de contrôle)
Octet 5 : Maître (écriture de l'esclave)

L'horloge est toujours générée par le maître, quelle que soit la direction.

Q7.9 La capture d'écran suivante représente l'écriture de deux octets dans une mémoire de type flash (EEPROM). La mémoire est initialement vide (toutes les pages ont été effacées). Afin de faciliter la lecture du chronogramme, la largeur d'un bit est identifiée par les deux curseurs, et le début de chacun des octets de données est identifié par un point bleu. Le premier octet est l'octet de contrôle, les deux suivants représentent l'adresse (le premier octet contenant les MSB), et les deux derniers les données envoyées.



a) Quelle est l'adresse à laquelle les données seront écrites?

Réponse : 0x0073 (115 en décimal)

b) Si on fait par la suite une lecture à l'adresse 0x0074 (116 en décimal), quel sera le premier octet reçu? Le deuxième?

Réponse : le premier octet sera 0x07 (soit la valeur du second écrit ici, à la fin du chronogramme) et le second aura la valeur 0xFF (la valeur par défaut d'une mémoire flash).

c) D'après ce que vous savez du protocole I²C et de son implémentation matérielle, que représentent les petites transitions 0 → 1 → 0 sur SDA, que l'on peut par exemple voir au-dessus du premier point bleu?

Réponse : Lors de l'envoi du bit de ACK par l'esclave, le maître doit lui laisser le contrôle du SDA. Les canaux d'un bus I²C sont reliés à des pull-ups (leur état par défaut est 1). Une fois que l'esclave prend le contrôle du SDA pour écrire son ACK, il y impose 0. Par la suite, il peut s'écouler un petit délai entre le moment où l'esclave relâche le bus et où le maître en reprend le contrôle, d'où la petite transition (qui est ignorée, puisque la lecture d'un bit se fait sur la partie haute (1) de SCL).

d) Tenant compte du temps de transmission d'un bit (identifié par les curseurs), et en négligeant le temps requis pour l'envoi de l'octet de contrôle, quelle est la vitesse de transmission effective (en octets/s) de ce lien I²C?

Réponse : le temps de transmission d'un bit est d'environ 11 us. De là, on déduit le temps de transmission d'un octet, qui sera de 9*11 = 99 us (ne pas oublier le bit de ACK!), soit environ 0.1 ms. La vitesse de transmission est donc d'environ 10 000 octets/s, soit 80 kbits/s. À titre de référence, la vitesse nominale du bus I²C (en mode standard, utilisé ici) est de 100 kbits/s.

Cours 8, RTC, temps et interfaces parallèles

Q8.1 Qu'est-ce qu'un RTC? Pourquoi utiliser un RTC au lieu d'un compteur interne?

Un RTC (real-time clock) est un circuit intégré pour compter le temps. Un RTC peut être préféré à un compteur interne pour plusieurs raisons : il est plus précis, il est généralement plus économique en consommation de puissance et il fonctionne sans intervention du microprocesseur. Concernant le dernier point, cela veut dire qu'il peut fonctionner lorsque le microprocesseur est en mode d'économie d'énergie et qu'il n'est pas limité à un compte de valeur (contrairement à un compteur).

Q8.2 Combien d'octets <u>au minimum</u> sont nécessaires pour représenter une date de l'année actuelle à la milliseconde près?

Approche 1 : ne donne pas le minimum!

10 bits pour les millisecondes, 6 bits pour les secondes, 6 bits pour les minutes, 5 bits pour les heures, 5 bits pour les jours du mois, 11 bits pour l'année, pour un total de 43 bits.

Étant donné qu'il est impossible de représenter un octet partiellement, il faut donc 6 octets (48 bits).

Approche 2 : solution exacte!

Dans une année, il y a 1000 millisecondes/seconde * 60 seconde/minute * 60 minutes/heure * 24heure/jour * 366 jours/année (une année bissextile!). Il y a donc 0x075CD78800 millisecondes par année et il faut 35 bits pour représenter ce nombre, donc un total de 5 octets.

Q8.3 Nommez deux avantages et deux inconvénients à utiliser une interface parallèle au lieu d'une interface série.

Avantages : bande passante supérieure pour une fréquence de communication donnée, simple à interfacer, simple à décoder.

Inconvénients : augmentation de la complexité d'un design matériel (plus de traces sur un PCB), émission de bruit électromagnétique accru, problème d'interférence entre les lignes (diaphonie/crosstalk), longueur limitée des liens câblés, problème de différence de longueurs des traces à très haute fréquence.

Q8.4 Décrivez le temps d'accès, le temps de cycle, la latence et la bande passante lorsque ces mots sont employés dans un contexte d'interface parallèle?

Le <u>temps d'accès</u> est le temps nécessaire pour accéder à une donnée.
Le <u>temps de cycle</u> est l'intervalle de temps minimum devant être alloué entre deux opérations consécutives.
La <u>latence</u> est le temps passé entre la demande d'une donnée et son arrivée au demandeur.
La <u>bande passante</u> est le débit de données pouvant être atteint dans un laps de temps donné.

Q8.5Pourquoi est-il important de donner la latence d'un périphérique?

Un périphérique est souvent très lent, ce qui implique que les données ne pourront y être écrites ou lues rapidement. Donner la latence permet donc au programmeur de pouvoir ajouter des temps d'attente dans son programme afin de bien utiliser le périphérique.

Q8.6 À quoi servent habituellement les broches ALE, nCS, nOE et nWE d'une interface parallèle.

Address Latch Enable (ALE) : Permet d'indiquer la présence d'une adresse valide sur le bus d'adresses.
Chip Select (nCS) : Permet d'activer le circuit du périphérique.
Output Enable (nOE) : Permet d'activer le périphérique en mode lecture.
Write Enable (nWE) : Permet d'activer le périphérique en mode écriture.

Q8.7 L'interface de communication avec une mémoire NAND-FLASH est souvent une interface utilisant essentiellement 8 lignes pour les adresses et les données. Comment peut-on adresser une mémoire NAND-FLASH de 1Mo?

Avec du multiplexage temporel : les 8 lignes entre la mémoire et le microprocesseur donnent l'adresses en plusieurs morceaux dans le temps.

Q8.8 Les interfaces parallèles servent souvent à communiquer avec des écrans ou des LCDs. Pourquoi?

Les pixels d'un écran ou d'un LCD sont souvent gérés de la façon que la mémoire : on écrit à une adresse de « mémoire » pour changer la couleur d'un pixel.

Q8.9 Si vous utilisez une interface parallèle pour accéder à une mémoire SDRAM, quelle instruction exécutée par le microcontrôleur permettra de contrôler les broches nRAS et nCAS?

Aucune. C'est l'interface parallèle qui gèrera tous les signaux d'accès à la mémoire si elle est configurée adéquatement.

Cours 9: Programmation de périphériques

Q9.1 Pourquoi un registre de statut existe-t-il pour les interruptions d'un périphérique?

Une interruption d'un périphérique peut avoir été déclenchée pour plusieurs raisons. Le registre de statut permet de déterminer la ou les raisons qui a poussé le périphérique à générer l'interruption, puis de faire le traitement approprié.

Q9.2 Nommez et décrivez les modes de transfert habituels des contrôleurs de DMA. Donnez ensuite un exemple de situation où chaque mode de transfert est utilisé.

- Transfert unique : permet de transférer un seul bloc de données. Utilisé par une application lors d'un transfert ponctuel dans le temps. Exemple : transfert de données du disque dur vers la mémoire requis par le système d'exploitation lors d'une faute de page…
- Transfert ping-pong : permet un transfert impliquant deux sources-destinations, en inversant ceux-ci entre le premier et le second transfert. Il s'agit d'une forme de double buffering : il est possible de travailler dans un tampon pendant la préparation de l'autre tampon puis inverser les rôles lorsque le travail est accompli dans l'un et la préparation est terminée dans l'autre. Exemple : traitement d'un signal continu en blocs. Un calcul comme une FFT est effectué sur le buffer ping pendant le remplissage du buffer pong et vice versa.
- Scatter-gather : permet de regrouper plusieurs petits transferts de plusieurs sources vers une destination ou de plusieurs destinations vers une source en un seul transfert DMA. Exemple : Mise en mémoire, dans un buffer contigu, de données provenant de plusieurs groupes de GPIOs.

Q9.3 L'utilisation du DMA et l'utilisation de caches dans le système de mémoire amène un nouveau problème dans les SMIs. Quel est ce problème?

L'utilisation du DMA peut amener un problème de cohérence avec les différents caches du système, c'est-à-dire qu'un transfert DMA peut faire en sorte que les données en mémoire ne concordent plus avec les données en cache. Dans le cas d'une lecture, le transfert DMA peut lire une valeur obsolète; dans le cas d'une écriture, le cache peut contenir une valeur qui n'est plus valide.

Q9.4 On retrouve le mauvais exemple suivant dans les notes de cours :

```
__irq void UART_ISR(void)
{
if( UnOctetEstRecu() )
        {
Tableau[NombreOctetsRecusUART] = OctetRecu();
NombreOctetsRecusUART++;
        }
        …
}

void Main(void){
init();
while(1){
        …
        if(NombreOctetsRecusUART > 0)
        {
                OctetAtraiter = Tableau[0];
                TraiteOctet(OctetAtraiter);
                DécaleOctetsDuTableauVersIndice0();
                NombreOctetsRecusUART--;
}       }       }
```

Il est dit un peu plus loin que pour corriger l'exemple, on pourrait appliquer 3 solutions différentes : désactiver les interruptions, utiliser un sémaphore ou éliminer la section critique en dupliquant des ressources. Changez le code pour du mauvais exemple pour illustrer ces trois solutions.

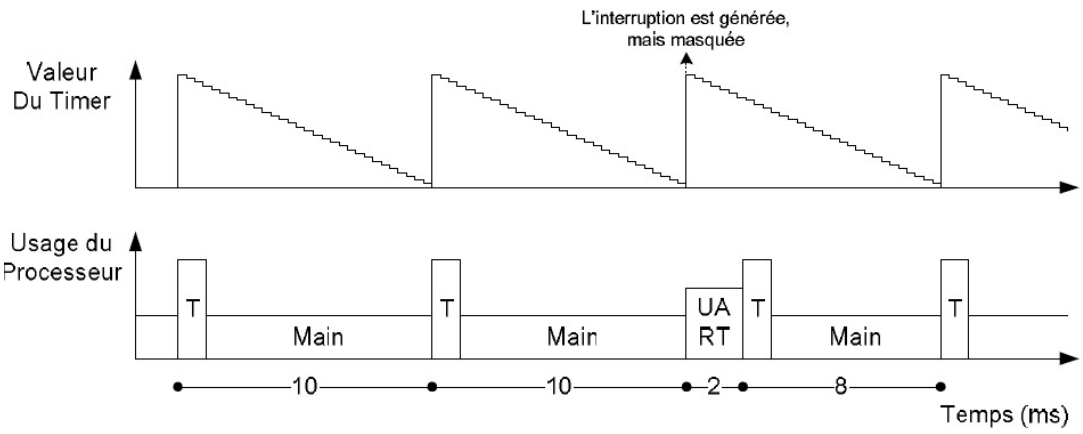
Désactivation des interruptions	Éliminer la section critique
__irq void UART_ISR(void) ***{*** ***if(UnOctetEstRecu())*** ***{*** ***Tableau[NombreOctetsRecusUART] = OctetRecu();*** ***NombreOctetsRecusUART++;*** ***}*** ***…*** ***}*** ***void Main(void){*** ***init();*** ***while(1){*** ***…*** DisableInterrupts(); ***if(NombreOctetsRecusUART > 0)*** ***{*** ***OctetAtraiter = Tableau[0];*** ***TraiteOctet(OctetAtraiter);*** ***DécaleOctetsDuTableauVersIndice0();*** ***NombreOctetsRecusUART--;*** ***}*** EnableInterrupts(); ***}*** ***}***	***__irq void UART_ISR(void)*** ***{*** ***if(UnOctetEstRecu())*** ***{*** ***Tableau[IndexTeteEcriture++] = OctetRecu();*** ***If(IndexTeteEcriture == TAILLE_DU_TABLEAU)*** ***IndexTeteEcriture = 0;*** ***}*** ***…*** ***}*** ***void Main(void){*** ***init();*** ***while(1){*** ***…*** ***if(IndexTeteLecture != IndexTeteEcriture)*** ***{*** ***OctetAtraiter = Tableau[IndexTeteLecture++];*** ***If(IndexTeteLecture == TAILLE_DU_TABLEAU)*** ***IndexTeteLecture = 0;*** ***TraiteOctet(OctetAtraiter);*** ***DécaleOctetsDuTableauVersIndice0();*** ***NombreOctetsRecusUART--;*** ***}*** ***}*** ***}***

Utiliser un sémaphore ne permet pas de régler le problème dans cet exemple, parce que l'interruption n'est pas rappeler par un système d'exploitation et parce que les deux tâches n'ont pas la même priorité.

Q9.5 Un timer est programmé en mode Auto-Reload pour générer une interruption basse priorité à toutes les 10 millisecondes. Dans le même programme, une interruption du UART survient lorsqu'un octet est reçu. Cet octet est traité dans la routine de l'interruption du UART pendant 2 millisecondes. Sachant que l'interruption du UART a une priorité supérieure à celle du timer, répondez aux questions suivantes :

1. ***Si une interruption de UART se produit juste avant l'interruption de timer, qu'arrive-t-il avec l'interruption du timer?***
2. ***Quand surviendront les prochaines interruptions de timer par rapport à celle avant l'interruption du UART?***

1. Le traitement de l'interruption du timer sera retardée de 2 millisecondes
2. Le timer, comme les autres périphériques, est un circuit matériel indépendant :



Q9.6 Pour quelle raison le code suivant peut-il conduire à une attente, dans la fonction Wait, beaucoup plus longue que prévu? Comment corriger simplement la fonction?

```
void Main(void)
{
	ConfigureTimerToGenerateIntAtEveryMS();
	ConfigureNVICToGenerateTimerInt();
	while(1)
	{
		…
		if(Event())
		{
			Wait(100);
		}
	}
}

void Wait(int TimeToWait)
{
	DecrementingCounter = TimeToWait;
	while(DecrementingCounter != 0)
	{};
}

int DecrementingCounter;

void Timer_ISR(void)
{
	DecrementingCounter --;
}
```

Si le « Main » ne s'exécute pas pendant deux interruptions de timer et que DecrementingCounter passe de 0 à -1 pendant ces interruptions, le délai d'attente sera 2_{32} interruptions de timer.

On corrige ainsi :

```
void Timer_ISR(void)
{
	if(DecrementingCounter != 0)
		DecrementingCounter --;
}
```

Q9.7 Pour la question précédente, proposez au moins une autre implémentation de la fonction Wait. Vous pouvez également changer le code dans Timer_ISR.

```
void Wait(int TimeToWait){
	unsigned int TimeStamp;
	TimeStamp = IncrementingCounter;
	while(IncrementingCounter - TimeStamp <= TimeToWait)
	{};
}

unsigned int IncrementingCounter;
void Timer_ISR(void){
	IncrementingCounter ++;
}
```

On notera que l'overflow de timer est géré automatiquement par la soustraction en notation complément 2.

Q9.8 (pour experts!) Dans le code suivant, on veut entreprendre une action 5 millisecondes après la réception d'un octet sur le SPI : on considère que le message est fini et on le traite. Dans le système, une interruption de systick timer se déclenche périodiquement à toutes les millisecondes. Par ailleurs, une interruption de SPI se produit lorsqu'on reçoit un octet. Lors de la réception, on note aussi le temps. Toutes les interruptions ont la même priorité. Voici le code:

Boucle principale	*Interruption de SPI*	*Interruption de timer*
void main(void){ ***while(1){*** ***if(MessageUnderRx)*** ***{*** ***if(MScount – LastRxTime > 5)*** ***{*** ***TraiteMessageSPI();*** ***MessageUnderRx = 0;*** ***}}}};***	***volatile unsigned int LastRxTime;*** ***void SPI_Rx_ISR(void)*** ***{*** ***MessageUnderRx = 1;*** ***LastRxTime = MScount;*** ***AjouteOctetRecuAuMessage();*** ***}***	***volatile unsigned int MScount;*** ***void Systick_Ms_ISR(void)*** ***{*** ***MScount++;*** ***}***

Malheureusement, le code n'attend pas toujours 5 millisecondes après la fin d'un message avant le traiter. À de rares occasions, il traite un message en plein milieu de sa réception. Bizarrement, le phénomène se produit lorsqu'on compile avec un compilateur, mais pas avec un autre compilateur. Sachant que les deux compilateurs compilent correctement (l'erreur est dans le code), pourquoi le problème survient-il?

Le problème se situe dans la ligne de code ***if(MScount – LastRxTime > 5)***. En C, il s'agit d'une seule ligne, mais il s'agit de plusieurs instructions pour le microprocesseur :

1. Lire MSCount (exemple : load R1, [adresse de MScount])
2. Lire LastRxTime (exemple : load R2, [adresse de LastRxTime])
3. Soustraire les deux valeurs (exemple : SUB R3, R1, R2)
4. Comparer à 5 et faire un branchement (exemple : SUB R3, R3, #5, BG R3, TraiteMessage)

Si la séquence suivante d'évènements se produit (tous ces évènements surviennent aléatoirement l'un par rapport à l'autre dans le temps) :

A. On lit MSCount dans le main (Exemple : MSCount vaut 100)
B. L'interruption de timer survient (Exemple : MSCount vaut maintenant 101)
C. Un octet SPI est reçu pendant l'interrution de timer qui est traitée après
    a. LastRxTime est mis égal à MSCount (Exemple: LastRxTime vaut maintenant 101)
D. On lit LastRxTime dans le main (Exemple : LastRxTime vaut maintenant 101)
E. On soustrait et compare, ce qui mène à l'erreur parce que MSCount est plus petit que LastRxTime (Exemple : 100 – 101 > 5 est vrai pour des unsigned int)

Pour régler le problème, il suffit d'inverser les actions 1 et 2 : lire LastRxTime avant MSCount… L'ordre de lecture des deux opérandes d'une opération arithmétique est habituellement laissé à la discrétion du compilateur!

Q9.9 Pour quelle raison l'accès au contrôleur SPI est parfois programmé directement, sans utiliser d'interruption ou de de DMA comme illustré ci-dessous?

```
void main(void)
{
        …
        ConfigureSPIPourEtreMaitreEtParlerAvecSlaveSPI();
        while(1)
        {
                …
                if(JeDoisEnvoyerUnOctetSPI())
```

```
                {
                        EnvoieOctetSurSPI(OctetAEnvoyer);
                }
                …
        }
}

void ConfigureSPIPourEtreMaitreEtParlerAvecSlaveSPI(void)
{
        REGISTRE_CONTROLE_SPI = VALEUR_POUR_ETRE_MAITRE;
        REGISTRE_CONTROLE_SPI += VALEUR_DE_POLARITE_ET_PHASE_DESIREE;
        REGISTRE_PRESCALE_SPI = VALEUR_POUR_FREQUENCE_SCLK_DESIREE;
        REGISTRE_CONTROLE_PERIPHERIQUE = VALEUR_POUR_ACTIVER_SPI;
        REGISTRE_ROLE_DES_BROCHES = VALEUR_POUR_UTILISER_BROCHES_POUR_SPI;
}

void EnvoieOctetSurSPI(unsigned char OctetTX)
{
        while(REGISTRE_STATUT_SPI != PRET_A_TRANSMETTRE)
        {};
        REGISTRE_TRANSMISSION_SPI = OctetTX;
}
```

La vitesse d'horloge du SPI (entre 1MHz et 10MHz) peut être proche de celle du microprocesseur (entre 10MHz et 168MHz).

Q9.10 En utilisant du pseudo code comme dans la question précédente, indiquez comment on pourrait refaire l'accès au SPI en utilisant des interruptions, c'est-à-dire sans attendre dans la fonction EnvoieOctetSurSPI.

Dans @void ConfigureSPIPourEtreMaitreEtParlerAvecSlaveSPI(void)@, Ajouter :
REGISTRE_CONTROLE_SPI += VALEUR_POUR_PERMETTRE_INT_SPI;
REGISTRE_CONTROLE_SPI += VALEUR_DECLENCHER_INT_SPI_APRES_TX;

Ajouter la fonction @void ConfigureNVICetTableVectIntPourIntDuSPI(void)@ dans le main :

```
void ConfigureNVICetTableVectIntPourIntDuSPI(void)
{
        REGISTRE_CONTROLE_NVIC = VALEUR_POUR_ACTIVER_INT_SPI;
        REGISTRE_PRIORITE_NVIC = VALEUR_POUR_PRIORITE_INT_SPI;
        TABLE_INT_VECT[INDEX_PERIPH_SPI] = MonISRSPI();
}

void MonISRSPI(void)
{
        REGISTRE_TRANSMISSION_SPI = ProchainOctetATransmettre();
}

void EnvoieOctetSurSPI(unsigned char OctetTX)
{
        AjouteOctetAListeDoctetsATransmettre();
        if(PasDeTransmissionSPIEnCours())
        {
                REGISTRE_TRANSMISSION_SPI = ProchainOctetATransmettre();
        }
}
```

Cours 10, Production et environnement

Q10.1 Expliquez pourquoi l'affirmation suivante est fausse : un ingénieur en électronique doit livrer ses appareils rapidement et, de ce fait, consacre la plus grande partie de son temps à implémenter des programmes embarqués ou implémenter des circuits électroniques.

Vous êtes toujours mieux de livrer un produit de qualité que de livre un produit rapidement. Un produit livrer rapidement risque de revenir ou de vous faire perdre un client… Dans cette optique, il faut prendre le temps de bien établir les exigences du client et, surtout, de bien concevoir le produit. Le temps pris pour le design est souvent récupérer en temps d'implémentation : penser avant d'agir permet souvent d'éviter des efforts inutiles.

Dans le même ordre d'idée, la qualité d'un produit étant très importante, les tests du produit devraient prendre une part importante du temps alloué au projet. Les tests, les tests et encore les tests sont nécessaires à tous les niveaux pour produire du matériel ou du logiciel de qualité.

Un programmeur qui ne fait pas ses tests unitaires aura rapidement une réputation de mauvais programmeur. Un designer qui ne fait pas ses tests système aura rapidement une réputation de mauvais designer. Un gestionnaire de projet qui ne fait pas de tests de produit formels perdra rapidement ses clients…

Pour toutes ses raisons, le temps d'implémentation est souvent beaucoup plus court que celui de design ou celui de test.

Q10.2 Donnez au moins six (6) raisons qui peuvent pousser un ingénieur de production à changer une pièce recommandée par un ingénieur de recherche et développement.

Voir la section de « Pré-production », sous-section « Revue du design matériel », dans les notes de cours.

Q10.3 Expliquez ce qu'est le JTAG.

Voir la section sur le JTAG dans les notes de cours.

Q10.4 Décrivez la norme RoHS et évaluez son impact en SMI.

Il s'agit d'un standard européen limitant les quantités de substances toxiques présentes dans les équipements électriques et électroniques, visant le plomb, le cadmium, le mercure, le chrome hexavalent, les PBB et les PBDE. Le plomb étant beaucoup utilisé auparavant pour des SMI, surtout pour les soudures, cette norme a eu un impact important.

Q10.5 Un manufacturier évalue la MTBF d'un modèle de disque dur à 90 000 heures. Un client se procure deux disques durs de ce modèle pour son serveur et les configure en redondance. Quelle est la probabilité que le montage fonctionne après 10 ans?

Il est impossible de répondre à cette question sans connaître la fonction de densité de probabilité, puisque la MTBF n'est qu'une moyenne!

Anecdote : en ce moment, la grappe de calcul de l'Université Laval comporte 1 048 disques durs de 1 To, chacun avec une MTBF estimée à 700 000 heures (près de 80 ans) par le manufacturier. Malgré cela, il y a entre 1 et 3 défaillances par mois dans le bassin de disques de la grappe de calcul...

Q10.6 Les mémoires FLASH et EEPROM supportent un nombre maximum de cycles d'écriture. Nommez quelques stratégies que vous pouvez utiliser pour maximiser la vie utile de ces mémoires.

- Uniformiser les écritures à travers les adresses de la mémoire afin de ne pas toujours écrire aux mêmes endroits.
- Lancer les écritures selon le format favorisé par la mémoire, par exemple par pages plutôt qu'en octets.
- Exploiter l'utilisation de caches afin de lancer des opérations d'écritures en blocs.
- Lancer des opérations d'écriture seulement lorsque c'est nécessaire! Cela peut sembler simpliste comme approche, mais vous seriez étonné d'apprendre l'inefficacité de certains programmes en utilisant un analyseur de performance (profiler)...

Astuce : les nouveaux disques durs à base de transistors (SSD, solid-state drive) comportent un micro-contrôleur afin d'appliquer les trois premières stratégies ci-haut. Les systèmes d'exploitation récents s'ajustent pour appliquer la quatrième.

Q10.7 Nommez quatre stratégies pour limiter la consommation en puissance d'un système embarqué.

- Désactiver les modules du microprocesseur et les périphériques rattachés lorsqu'ils ne sont pas utilisés.
- Utiliser les fréquences et les tensions les plus basses possibles pour les circuits intégrés, puisque la puissance dynamique du circuit augmente proportionnellement à la fréquence d'opération et au carré de la tension d'opération.
- Utiliser des dispositifs demandant une faible consommation de courant.
- Utiliser une source d'alimentation plus efficace.

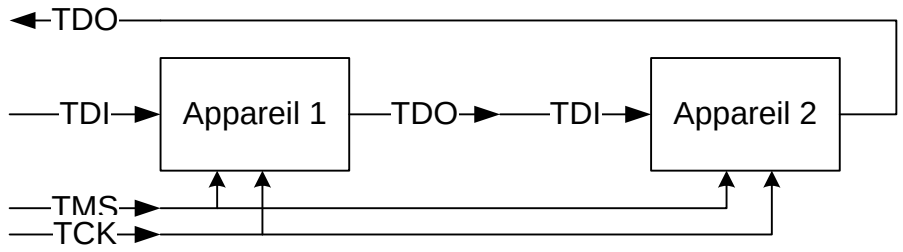
Astuce : il est important d'analyser les besoins avant de se lancer dans des alimentations complexes : bien qu'une alimentation linéaire soit moins efficace qu'une alimentation à découpage (SMPS), cette dernière peut s'avérer beaucoup plus coûteuse, tant en ressources matérielles qu'humaines, pour les bénéfices rapportés.

Q10.8 Quelle norme européenne établit les règles de mise au rebut des équipements électrique et électroniques?

WEEE

Q10.9 Lorsque voulez tester la soudure des broches de deux circuits intégrés par JTAG, comment les broches TDO, TDI, TMS et TCK sont reliées à ces deux circuits?

TMS et TCK sont reliées en parallèle. TDO et TDI sont reliées en série :



Annexe A Équations de conversion des ADCs

Problème

Lorsqu'un ADC n bits mesure X counts avec une tension de reference Vref, on retrouve, dans la littérature, deux équations de conversion de X en valeur analogique lue.

Equation 1, Conversion de counts d'ADC vers Voltage lu, approximation 1:

$$Vlu = \frac{X\ counts}{2^n\ counts} * Vref$$

Equation 2, Conversion counts d'ADC vers Voltage lu, approximation 2:

$$Vlu = \frac{X\ counts}{(2^n - 1)\ counts} * Vref$$

Quelle équation devrait-on prendre?

Solution

Par construction, la plupart des ADCs avec comparateurs convertissent la tension analogique d'entrée de la manière suivante :

Equation 3, Conversion Voltage lu vers counts d'ADC

$$X\ count = Partie\ entière\ de\left(\frac{Vlu}{Vref} * 2^n\ counts\right) \text{ ou } X\ count = \left\lfloor\frac{Vlu}{Vref} * 2^n\ counts\right\rfloor$$

Les comparateurs à l'intérieur de l'ADC (ADC FLASH ou Approximation successive ou autre), forcent un arrondissement vers le bas du nombre de counts lus. Pour convertir correctement et éviter de perdre ½ bit de précision, l'équation de conversion inverse devrait être :

Equation 4, Conversion counts d'ADC vers Voltage lu, correcte

$$Vlu = \frac{(X + 0.5)\ counts}{2^n\ counts} * Vref$$

Dans l'équation 4, le 0.5 counts vient contrebalancer l'arrondissement inférieur de l'équation 3. On convertit la sortie de l'ADC à la valeur moyenne des entrées pouvant donner cette sortie.

Les équations 1 et 2 sont des approximations de l'équation 4, pouvant être pratiques afin de simplifier les calculs (qui veut s'embarrasser d'ajouter 0.5 counts dans un monde numérique!!!).

Quelle approximation est la meilleure, l'équation 1 ou l'équation 2?

L'erreur pour l'approximation 1 est la suivante (équation 4 – équation 1):

Equation 5, erreur pour l'approximation 1

$$erreur = \frac{(X + 0.5)\ counts}{2^n\ counts} * Vref - \frac{X\ counts}{2^n\ counts} * Vref$$

$$erreur = \frac{0.5\ counts}{2^n\ counts} * Vref$$

L'erreur pour l'approximation 1 est de ½ LSB…

L'erreur pour l'approximation 2 est la suivante (équation 4 – équation 2):

Equation 6, erreur pour l'approximation 2

$$erreur = \frac{(X + 0.5)\ counts}{2^n\ counts} * Vref - \frac{X\ counts}{(2^n - 1)\ counts} * Vref$$

$$erreur = \frac{(0.5 * 2^n - X - 0.5)\ counts}{2^n(2^n - 1)\ counts} * Vref$$

L'erreur dépend du nombre de counts lus : X étant de 0 à $2^n - 1$.

Si X = 0, l'erreur est de :

$$erreur = \frac{0.5(2^n - 1)\ counts}{2^n(2^n - 1)\ counts} * Vref$$

$$erreur = \frac{0.5\ counts}{2^n\ counts} * Vref$$

Si X = $0.5*(2^n)$, (Vin = Vref/2):

$$erreur = \frac{(0.5 * 2^n - 0.5 * 2^n - 0.5)\ counts}{2^n(2^n - 1)\ counts} * Vref$$

$$erreur = \frac{-0.5\ counts}{2^n(2^n - 1)\ counts} * Vref$$

Si X = 2^n -1, (Vin >= Vref):

$$erreur = \frac{(0.5 * 2^n - 2^n - 1 - 0.5)\ counts}{2^n(2^n - 1)\ counts} * Vref$$

$$erreur = \frac{-0.5\ counts}{2^n} * Vref$$

Conclusion

Convertir des counts d'ADC avec l'équation 2 (diviser la valeur lue par 2^n-1) permet d'approximer adéquatement l'entrée de l'ADC, surtout lorsque cette entrée est proche de la moitié de la plage d'entrée de l'ADC et lorsque le nombre de bits de l'ADC, n, est grand. Dans ces cas communs, l'écart entre le meilleur estimé de l'entrée que l'on peut faire et l'estimé obtenu est presque nul (proportionnel à $1/2^{2n+1}$).

Par exemple, si on suppose un ADC 8 bits ayant une référence de 3.2V (0.0125V/count). Si la tension d'entrée est entre 1.6V (inclus) et 1.6125V (exclus), l'ADC lira 128 counts. Autrement dit, si l'ADC lit 128 counts, il est juste de considérer que l'entrée de l'ADC était, en moyenne, 1.60625V (une moyenne des entrées possibles). C'est cette estimation qui donne le moins d'erreur. Si on avait divisé 128 par 256 et multiplié par Vref (approximation 1), on aurait obtenu Vin = 1.6V, soit ½ bit d'erreur statistiquement. Or, si on divise 128 counts par 256-1 = 255 puis si on multiplie par Vref, on obtient 1.60637V. L'erreur entre approximation 2 et l'approximation idéale est de 1.60625V – 1.60637V = presque rien…