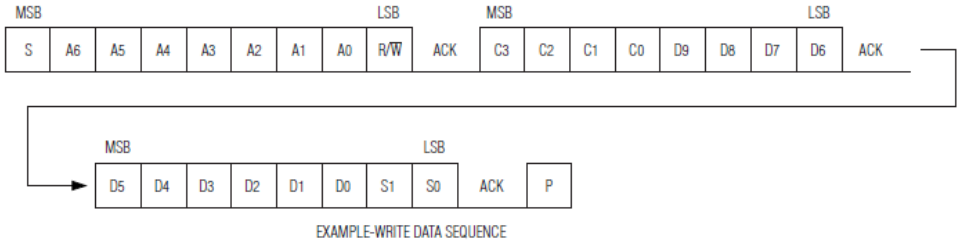
Examen 2012 – systèmes embarqués

J.-M Friedt, 20 janvier 2013

1 Transaction I2C sous GNU/Linux

Un programme de démonstration des fonctionnalités d'un convertisseur numérique-analogique connecté au processeur iMX27 (cœur ARM9) par bus I2C est proposé par Armadeus Systems sur plate-forme APF27 à `http://www.armadeus.com/_downloads/formations/formation_data/03_hello_i2c/hello.c`

Le composant utilisé au cours de cette démonstration est un MAX5821, convertisseur de 10 bits de résolution, dont un extrait de la *datasheet* [1] indique la séquence d'écriture (on notera que S1=S0=0 dans tous les cas)



EXAMPLE-WRITE DATA SEQUENCE

dont l'adresse (bits A0 à A6) est définie par un appel par `ioctl()` lors de l'initialisation du module noyau Linux `i2cdev` chargé des transactions I2C.

Dans l'appel à la fonction définissant la tension de sortie d'un des canaux, le code est

```
static void dac_set_channel_A(unsigned int value) {
  unsigned char buf[2] = { 0, 0 };
  buf[0] = (value & 0x3ff) >> 6;
  buf[1] = ((value & 0x3ff) << 2) & 0xfc;
  write(dac_fd, buf, 2);
}
```

1. I2C est un bus synchrone. En rappeler la principale caractéristique.
2. Quelle est la gamme des valeurs d'entrée de `value`, argument de la fonction définissant la tension de sortie ?
3. Que contiennent les deux octets formant le tableau `buf` ? En particulier, donner leur contenu si nous passons en argument la valeur 300.
4. L'auteur de ce code s'était à l'origine trompé en écrivant 0x3F à la place de toutes les occurrences de 0x3FF. Commenter quant au contenu de `buf[0]`. Quelle sera dans ce cas la sortie en tension du composant ?

2 Code de parité

Les codes de parité sont classiquement utilisés pour valider la cohérence d'une transmission. Dans le meilleur des cas ils permettent de corriger une erreur de transmission de données, mais en pratique ils ont principalement pour vocation de valider l'intégrité des données transmises. Un exemple trivial est le bit de parité dans une liaison asynchrone compatible RS232 : un bit additionnel est ajouté à la fin de la transmission de chaque octet pour valider que la valeur transmise vérifie certaines propriétés, à savoir que la somme des bits transmis et du bit de parité soit paire (cas de la parité paire, *Even*) ou impaire (cas de la parité impaire, *Odd*), selon le protocole retenu par convention entre les deux interlocuteurs.

1. quel est l'impact du bit de parité sur le débit des transactions, si par exemple les mots transmis sont codés sur 7 bits (protocoles dit 7E1 ou 7O1, le "1" final signifiant qu'un seul bit de STOP est communiqué) ?

2. quelle est la capacité de correction de ce bit de parité en cas de transmission sur un canal bruité ?
3. l'implémentation logicielle se fait sous forme de somme. Quelle pourrait être une implémentation matérielle (portes logiques uniquement) simple (qui justifie l'utilisation massive de cette méthode de validation d'intégrité des transactions numériques dans les circuits logiques) ?

En général, le code de parité n'agit pas sur chaque octet individuel mais sur l'ensemble du message transmis afin de réduire l'impact sur la bande passante, sous hypothèse que l'erreur de transmission du message est exceptionnelle et qu'une demande de re-transmission n'est pas excessivement pénalisante. Un exemple de code de parité est celui utilisé pour valider l'intégrité des numéros de carte de crédit ou codes SIRET [2] (à ne pas confondre avec les éléments de cryptographie à vocation de secret : le code de parité a uniquement pour vocation à valider l'intégrité d'une donnée et en ce sens n'a aucune prétention de confidentialité), aussi nommé code de Lund. Une description de cet algorithme qui définit la valeur des deux derniers chiffres, fourni dans [2], est

> "La clé de contrôle utilisée pour vérifier de l'exactitude d'un identifiant est une clé "1-2". Le principe est le suivant : on multiplie les chiffres de rang impair à partir de la droite par 1, ceux de rang pair par 2 ; la somme des chiffres obtenus doit être un multiple de 10."

4. dans le cas de la carte bleue ci-dessus, est-ce que la clé est vérifiée (attention à la notion de *chiffres* dans la somme) ?
5. que se passe-t-il si le premier chiffre est incrémenté de 1 lors d'une erreur de communication ?
6. que se passe-t-il si les deux premiers chiffres sont inversés lors d'une erreur de transcription ?
7. commenter sur la capacité d'identification d'erreur par rapport au bit de parité vu auparavant. Quel peut être alors l'intérêt de ce code de validation des transactions ?

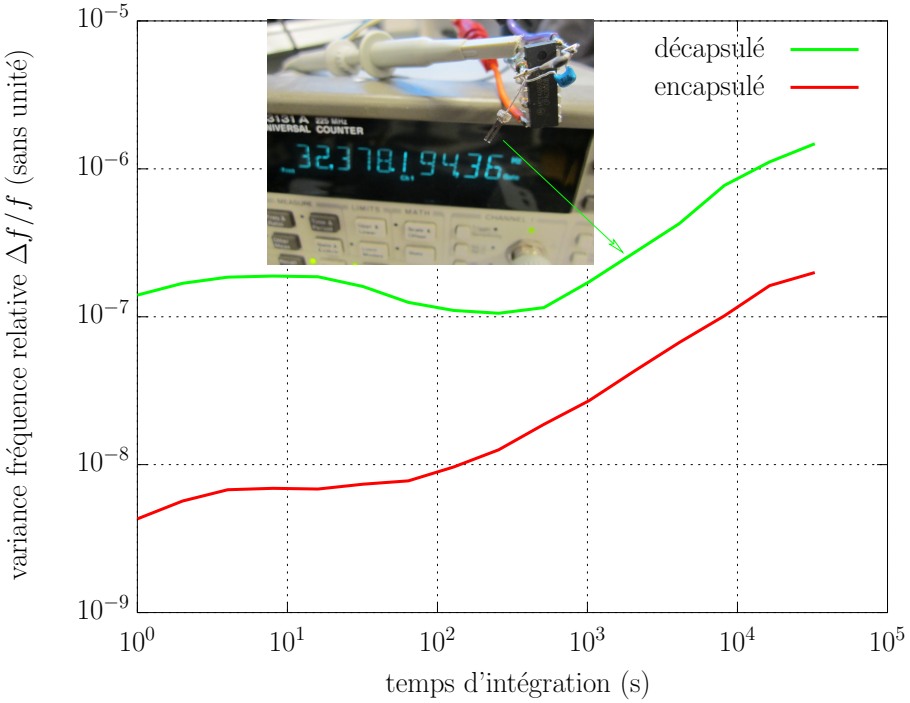
3 Horloge et liaison série asynchrone

La variance d'Allan est un outil classique de caractérisation de stabilité d'un oscillateur, qui fournit la fluctuation de fréquence relative (ordonnée) sur une durée d'intégration donnée (abscisse). Ainsi, une fluctuation aléatoire se traduit par une baisse de la variance avec le temps d'intégration (décroissance de la variance avec la racine du nombre d'échantillons contenus dans la moyenne glissante), et une dérive lente (dérive thermique par exemple) se traduit par une croissance de la variance.

L'exemple ci-dessous propose une mesure sur un diapason horloger tel que ceux souvent utilisés pour cadencer les microcontrôleurs, d'une part encapsulé sous vide (courbe verte) et d'autre part dans le cas très défavorable d'un résonateur décapsulé et soumis aux pertes viscoélastiques avec l'air (facteur de qualité dégradé). Dans cet exemple, un circuit d'oscillation trivial est mis en œuvre en plaçant le résonateur dans un circuit d'entretien basé sur un composant 4060 [1], suivant une architecture classiquement retenue dans les circuits numériques (l'amplificateur d'entretien de l'oscillation est une porte

1. `http://www.fairchildsemi.com/ds/CD/CD4060BC.pdf`

inverseuse tandis que deux condensateurs fournissent la rotation de phase nécessaire à vérifier les conditions de Barkhausen de l'oscillateur).



Nous analyserons cette courbe en considérant que la fluctuation relative de fréquence Δf d'un diapason encapsulé (courbe rouge) autour de sa fréquence de fonctionnement $f = 32768$ Hz est de $\Delta f/f = 4 \times 10^{-9}$, et pour un diapason décapsulé (courbe verte) de $\Delta f/f = 1 \times 10^{-7}$ sur une seconde d'intégration.

Le débit maximum des transactions sur une liaison asynchrone est déterminé par l'intervalle de temps pendant lequel les deux interlocuteurs restent synchronisés malgré les fluctuations de leurs horloges locales [3].

On suppose que dans un premier temps la multiplication de fréquence du quartz de 32768 Hz vers la fréquence de communication des bits sur le bus asynchrone par boucle à vérouillage de phase (PLL) d'un facteur P maintient la fluctuation relative de fréquence constante. En effet pour une PLL idéale, les fluctuations de fréquence sont multipliées par P ainsi que la fréquence de travail, donc le quotient reste inchangé. Comme la durée de communication est inférieure à la seconde, nous prendrons comme fluctuation de fréquence la variance d'Allan $\Delta f/f$ à 1 s.

1. Quelle est la fluctuation de temps de transfert d'un bit lorsque la fréquence relative des horloges cadençant les deux interlocuteurs varie de $\Delta f/f$, f étant la fréquence d'horloge cadençant le microprocesseur au cours de transactions au débit de $P \times f$ bits/s ?
2. Quelle est la durée de transaction de N bits sur liaison cadencée à fréquence $P \times f$? Quelle est la fluctuation du temps de transfert lorsque N bits successifs sont transmis après le bit de synchronisation qu'est le START bit ?
3. Quelle est la condition pour maintenir l'intégrité des communications entre deux interlocuteurs sur une liaison asynchrone ?
4. En déduire le débit maximal de communication série asynchrone supporté entre deux interlocuteurs cadencés par de telles horloges en supposant que chaque transaction porte sur N bits transmis au débit de $P \times f$ bits/s, la mesure de stabilité portant toujours sur l'oscillateur local de fréquence f.
5. Application numérique sur le débit accessible dans le cas où $N = 65535 \times 8$ (cas de la taille maximale d'un paquet IP, soit 65536 octets) : quelle est la stabilité d'horloge nécessaire pour la transaction d'un tel paquet ? Est-elle vérifiée dans notre cas ?

On suppose maintenant que dans le cas d'une PLL réaliste, la bruit relatif de fréquence est multiplié par P, donc le bruit croît quadratiquement avec le facteur multiplicatif de la PLL (après multiplication par P, les fluctuations relatives de fréquence ne sont plus $\Delta f/f$ mais $P\Delta f/f$).

Reprendre la séquence des questions précédentes avec cette nouvelle hypothèse, en exprimant les conditions sur le débit de communication en fonction de la stabilité de l'oscillateur de référence.

6. Application numérique sur le débit maximum accessible dans le cas où $N = 65535 \times 8$ (cas de la taille maximale d'un paquet IP, soit 65536 octets) pour une transaction commandée par un processeur cadencé par le diapason dont la fréquence est multipliée.
7. Que peut-on en déduire pour une liaison ethernet qui se fait à 10 Mbits/s ? Commenter.
8. Bonus : quelle solution peut-on envisager pour augmenter le débit malgré l'utilisation d'oscillateurs locaux de qualité médiocre tels que ceux présentés ici ?

4 Encore une fois ...

1. Exprimer la valeur hexadécimale 0x7B en décimal.
2. Exprimer la valeur 123 en hexadécimal.
3. Combien de bits faut-il pour coder la valeur 1023 dans un format binaire ?
4. Quelle est la durée théorique de fonctionnement d'un circuit consommant 1 A en activité et 1 mA en veille, activé pendant 15 s toutes les heures, sachant qu'il est alimenté par une pile de capacité 3717 mA.h ?
5. Quelle serait la conséquence de réduire la durée de fonctionnement du circuit précédent de 15 à 0,117 s ?
6. Quelle commande permet d'effacer un fichier sous unix ?

Références

[1] disponible à `http://datasheets.maximintegrated.com/en/ds/MAX5821.pdf`

[2] H. Le Grand, *Normalisation des codes SIREN - SIRET* (2001), disponible à `http://www.dsi.cnrs.fr/conduite-projet/phasedeveloppement/technique/etude-detaillee/modele-de-donnees/regles-SIREN-SIRET.pdf`

[3] *The Secret Life of Machines – the fax machine*, disponible à `http://www.youtube.com/watch?v=IaCfs5Xb-EI`

5 Solutions

5.1 Transaction I2C sous GNU/Linux

1. Un bus synchrone partage une horloge commune entre les interlocuteurs.
2. Le DAC est sur 10 bits donc les valeurs admissibles en entrée sont comprises entre 0 et 1023. Toute valeur en dehors de cette gamme sera tronquée lors des masquages (`&3ff`).
3. `buf` contient les 10 bits définissant la tension de sortie du DAC selon $V_{DAC} = V_{REF} \times buf/1024$. Cependant l'octet de poids faible étant complété des bits S0 et S1, au lieu d'avoir 8 et 2 bits dans les octets de poids faible et fort respectivement, on a 6 et 4 bits respectivement. Noter que `buf[1]` contient l'octet de poids faible et `buf[0]` l'octet de poids fort. Si la valeur est 300=0x12C, alors `buf[0]=4` et `buf[0]=0xB0`.
4. `0x3f` ne conserve que les 6 bits de poids faible. Or le décallage à droite de 6 places des 6 bits de poids faible donne toujours 0, donc on a toujours `buf[0]=0`. Pour ce qui est de `buf[1]`, on conserve 6 bits, qu'on décalle de 2 places vers la gauche, puis masque avec les bits 8 à 2, donc on ne conserve finalement que les 6 bits de poids faible du mot `value` qui avait été passé en argument : la gamme de tensions accessible est fortement réduite, puisque seuls les 6 bits de poids faible sur les 10 bits disponibles sont exploités.

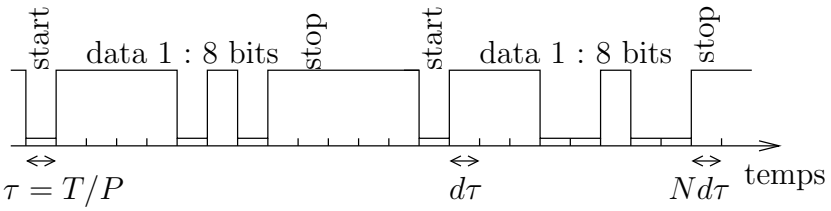
5.2 Code de parité

1. la transmission d'un mot de 7 bits ne nécessite plus 9 bits (START/7 bits de données/STOP) mais 10, donc le débit est réduit de 10%.
2. ce bit additionnel ne permet de détecter qu'une unique inversion d'état (ou un nombre impair d'inversions d'état sur un canal très bruité), mais ne permet pas de savoir quel bit a été transmis de façon erronée. Un nombre pair d'erreurs ne sera pas détecté.
3. XOR implémente le bit de parité : de façon récursive, une chaîne de $N-1$ portes XOR permet de tester la parité de N bits, en partant des deux bits de poids faible (si les deux bits sont identiques la sortie est 0, et si les deux bits sont différents la sortie est 1, donc parité paire). Pour la parité impaire, on ajoute une porte inverseuse (NOT).

   > "La clé de contrôle utilisée pour vérifier de l'exactitude d'un identifiant est une clé "1-2". Le principe est le suivant : on multiplie les chiffres de rang impair à partir de la droite par 1, ceux de rang pair par 2 ; la **somme des chiffres obtenus** doit être un multiple de 10."

4. oui la somme est vérifiée, si on prend soin de noter que la somme des *chiffres* signifie que lorsque $2 \times 9 = 18$ on somme $1 + 8$.
5. une valeur erronée se traduira par une somme fausse (le modulo est 10, chaque chiffre est codé sur une valeur entre 0 et 9, donc la somme est nécessairement fausse).
6. l'inversion de deux valeurs successives se traduit par une somme erronée puisque une valeur sur 2 est doublée avant sommation...
7. ... cette méthode est donc plus robuste que le bit de parité vu auparavant qui n'est pas capable de voir un nombre pair d'erreurs (qui est le cas d'une inversion de bits). Cette correction simple préliminaire évite d'engager les procédure cryptographiques de validation du code banquaire (nécessitant transaction des valeurs fournies par le client au site d'achat et activation des bases de données banquaires) si le numéro transmis est trivialement erroné du fait d'une erreur de saisie lors de la copie du numéro de carte bleue lors d'un achat (à tester sur le web ...).

5.3 Horloge et liaison série asynchrone



1. la période est l'inverse de la fréquence : $T = \frac{1}{f} \Rightarrow \left|\frac{df}{f}\right| = \left|\frac{dT}{T}\right|$. Par ailleurs, la période entre deux symboles (bits) est $\tau = T/P \Rightarrow \frac{df}{f} = \frac{dT}{T} = \frac{d\tau}{\tau}$
2. la durée est $N \times \tau = N \times T/P$ ...
3. ... et comme l'horloge est réinitialisée à chaque transaction par le START bit, la fluctuation d'horloge au cours de cet intervalle est $N \times d\tau$.
4. Il faut que au bout des N bit transmis, la variation d'intervalle de temps soit inférieur à 1/2 intervalle de temps (temps de transfert d'un bit)...
5. ... donc $N \times d\tau < \tau/2 \Leftrightarrow \frac{df}{f} < \frac{1}{2N}$. Cette relation est indépendante de P puisque la PLL est supposée idéale : la montée en fréquence n'induit pas de fluctuation additionnelle de la base de temps. Pour $N = 65536 \times 8 + 2$ (le START et STOP bits sont ici négligeables devant la taille du message), il suffit d'avoir un oscillateur local de stabilité meilleure que $1/1048578 \simeq \times 10^{-6}$ qui est trivialement vérifié dans nos exemples. Cependant, ce résultat est surprenant (pour l'auteur tout au moins), et indique que la débit de communication n'est *pas* limité par la stabilité d'un oscillateur idéalement multiplié par PLL, puisque le temps de transaction est d'autant plus court que la vitesse est élevée, ce qui compense l'augmentation des fluctuations de l'oscillateur avec la montée en fréquence (le facteur multiplicatif P s'annule au numérateur et au dénominateur). D'où la seconde hypothèse (cas suivant, quelque peu artificiel certes) d'une fluctuation qui croît de façon quadratique avec P.
6. Si maintenant les fluctuations croient de façon quadratique avec le facteur de multiplication P, on a toujours $\frac{dT}{T} = \frac{df}{f}$ (définition de la période) mais cette fois $\tau = T/P$ induit $\frac{d\tau}{\tau} = P \times \frac{df}{f}$ (puisque τ est une période d'un oscillateur multiplié P fois). Dans ce cas, la condition d'intégrité de la communiction devient $d\tau < \tau/2 \Rightarrow \frac{d\tau}{\tau} < \frac{1}{2N} \Leftrightarrow P\frac{df}{f} < \frac{1}{2N}$ et finalement on trouve la condition sur la stabilité de l'oscillateur de référence $\frac{df}{f} < \frac{1}{2NP}$.

   Si $N = 65536 \times 8 + 2$ (on n'envoie que un START et STOP bit en début et fin de transaction), alors $\frac{1}{2NP} \simeq 1/P \times 10^{-6}$ et l'oscillateur le meilleur que nous présentions ici présentant $\Delta f/f = 4 \times 10^{-9}$, alor $P < \frac{10^{-6}}{4 \times 10^{-9}} = 250$. Une transaction présente donc un débit maximam de $32768 \times 250 = 8.2$ Mb/s.
7. Le diapason encapsulé est presque capable de respecter cette condition, le diapason décapsulé en est incapable, il est 30 fois trop instable. Pourtant les liaisons 10 Mb/s sont largement dépassées et les débits de communication sur réseau ethernet dépassent le Gb/s : il ne s'agit pas d'une simple liaison asynchrone mais d'un codage Manchester dans lequel le codage est conçu de façon à maximiser le nombre de transitions d'états entre bits successifs pour permettre de resynchroniser les horloges des deux interlocuteurs par boucle à vérouillage de phase.
8. Bonus : toutes ces discussions portent sur un codage sur deux états (bit à 1 ou 0). Les codages plus subtils (sur la phase ou l'amplitude du signal) permettent de coder de nombreux états pour chaque symbole, et donc de réduire d'autant la fréquence de transition de ces symboles. C'est ainsi que wifi est par exemple capable de coder jusqu'a 64 états par symbole transmis et donc d'atteindre 54 Mb/s en faisant transiter des signaux bien plus lentement que 54 MHz.

5.4 Encore une fois ...

1. 0x7B=123
2. 123=0x7B !
3. $\log_2(1023) \leq 10 \Rightarrow 10$

4. $1000 \times 15/3600 + (3600 - 15)/3600 \times 1 = 5.1625$ mA.h et pile de 3717 mA.h donc l'autonomie est $3717/5.1625 = 720$ h.
5. Si on réduit la durée d'activité du circuit précédent à moins d'une seconde, alors $1000 \times 0,117/3600 + (3600 - 0,117)/3600 \times 1 \simeq 1$ mA.s et l'autonomie est de 3600 h ou 150 jours. C'est donc bien la durée d'activité qui limite la durée de vie de ce circuit.
6. `rm fichier`