GIF-3002 Périphériques, Real Time Clock et Interface Parallèle

Ce document présente les RTCs, le temps dans les systèmes microprocesseurs et les interfaces parallèles. Il s'inscrit dans la séquence de cours sur les périphériques communs des systèmes microprocesseurs.

1 Real Time Clock (RTC) et temps des microcontrôleurs

1.1 RTC

Un RTC (Real Time Clock, Horloge temps réel en français!) est un circuit intégré qui compte le temps. À travers des interfaces variées, un RTC donne les secondes, les minutes, le jour, le jour de la semaine, le mois et l'année en cours.

Un RTC est habituellement un circuit qui consomme très peu d'énergie. Il doit être alimenté par une batterie (ou un condensateur) pour continuer de compter le temps lorsque l'alimentation principale est éteinte. Il doit aussi avoir un circuit d'horloge très précis afin d'éviter des erreurs qui s'accumulent avec le temps.

Supposons par exemple, une RTC ayant une horloge de 32.768kHz nominale opère à une température de 35C à la place de 25C. Sachant que l'horloge varie de 20ppm maximum à 35C, de combien de minutes par année peut se tromper le RTC dans le pire cas? Dans une année, il y a 365jours/année*24heures/jours*60minutes/heures, soit 525600 minutes. 20 ppm est 20 partie par million, soit 20/1000000 d'erreur. Donc, sur un an, il y aura 525600*20/1000000 = 10.512 minutes d'erreur, dans le pire cas.

Un RTC peut être intégré dans un microcontrôleur ou être externe. Les RTCs intégrés sont plus simple à interfacer, mais il faut souvent ajouter des broches externes au microprocesseur pour une alimentation suppléante (une batterie) et une horloge indépendante. Les RTCs externes sont plus difficiles à interfacer, mais la gestion de la batterie et de l'horloge indépendante est plus simple.

> *Lorsque les microcontrôleurs avec RTC internes opèrent en mode d'économie d'énergie (veille, sommeil, inactif), le RTC continue presque toujours d'être alimenté. Ainsi, certains microprocesseurs n'ont pas de broche externe pour la batterie : on assume que la batterie est branchée directement sur les broches d'alimentation du microprocesseur, qu'une faute d'alimentation sera détectée (générant une interruption NMI par exemple) et que le programme roulant dans le microprocesseur mettra le microprocesseur en veuille ou inactif si la batterie est utilisée.*

1.2 RTC et Timers

Le circuit de base pour mesurer les temps dans les microcontrôleurs est le Timer. Les timers sont beaucoup plus utilisés que les RTC et, parfois, se substituent au RTC lui-même. Les timers sont plus facile à interfacer (RTC externe), ils ne requièrent pas de batteries externes et ils sont basés sur l'horloge du microprocesseur…

Par contre, les timers ont quelques désavantages sur les RTCs lorsqu'il faut compter du temps réel: ils sont moins précis (l'horloge du microcontrôleur n'est pas toujours une horloge précise), ils peuvent contenir un nombre limité de valeurs (un timer 32 bits ne contient que 4Go valeurs, si l'unité de temps est 1ms, combien de jours roulera le timer avant de déborder?) et ils ne donnent pas le temps réel (ce ne sont que des comptes!). Par ailleurs, il est rarement possible de descendre la consommation de courant/puissance d'un timer au même niveau que celle d'un RTC.

1.3 Format du temps

Il y a plusieurs façon de représenter le temps en binaire. Le nombre d'octets utilisés pour noter un temps varient d'une représentation à l'autre. Voici deux représentations communes du temps :

1.3.1 Représentation de base

Pour représenter le temps, chaque unité de temps peut être associée à un ou plusieurs octets en mémoire :

- 2 octets de 0 à 999 pour les millisecondes
- 1 octet de 0 à 59 pour les secondes
- 1 octet de 0 à 59 pour les minutes
- 1 octet de 0 à 23 pour les heures
- 1 octet de 0 à 31 pour les jours du mois
- 2 octet de 0 à 65535 pour les années

1.3.2 Temps POSIX

Le POSIX est un standard pour UNIX et le temps POSIX fait partie de ce standard. Le temps POSIX est écrit sur quatre octets ou chaque valeur vaut une seconde. Il s'agit du nombre de secondes écoulées depuis le 1er janvier 1970 au fuseau horaire 0 (UTC = Universal Time Clock = temps au fuseau 0).

1.4 Le temps et l'homme…

Dans un système microprocesseur, il est généralement recommandé de garder un système de temps réel aussi simple que possible. En effet, plusieurs éléments peuvent complexifier l'horloge temps réel : les fuseaux horaires (recommandé de toujours travailler en UTC), le Daylight Saving Time (DST, heure avancée et heure reculé, recommandé ne pas en tenir compte), les leap seconds (des secondes insérées pour compenser les variations du jour solaire, recommandé ne pas en tenir compte)… La conception d'un circuit d'horloge temps réelle peut rapidement devenir très complexe…

Habituellement, les horloges temps réels des systèmes microprocesseurs sont resynchronisées régulièrement à partir d'une interface externe.

Certaines données (comme le jour de la semaine) peuvent être calculées assez facilement par le RTC si le RTC supporte cette option et si on lui fournit les données adéquates (un jour de référence par exemple).

2 Interfaces Parallèles

Une interface parallèle est un circuit intégré qui gère un bus de donnée, un bus d'adresse et un bus de contrôle afin de lire ou écrire des données en parallèle (8 bits, 16 bits, 32 bits, 64 bits…).

Le premier rôle des interfaces parallèles dans les systèmes microprocesseurs est indubitablement de permettre un accès à la mémoire. Cet accès est habituellement entièrement géré par le matériel (le microprocesseur lui-même ou une interface de bus) et il ne requiert pas d'instructions ou de séquences d'instructions pour se faire[1].

> *La plupart des interfaces parallèles permettent de lire des instructions d'une mémoire d'instructions : il est fondamental d'avoir des interfaces parallèles qui sont contrôlées automatiquement par le matériel.*

Les interfaces parallèles servent aussi à contrôler, lire ou écrire les périphériques. Il est commun de relier ces interfaces à des ADCs, des DACs, des bus parallèles, des afficheurs ou autre périphérique.

2.1 Définitions

Voici quelques définitions reliées aux timings des interfaces parallèles :

Temps d'accès (Accès Time) : Le temps pris pour accéder à une donnée en lecture ou en écriture. Peut différer selon l'opération.

Temps de cycle : Le temps minimum entre deux opérations consécutives de lecture ou d'écriture.

Latence : Temps requis avant l'apparition des premières données. La latence nCAS par exemple est le temps pris entre la descente/activation de nCAS et l'apparition des données d'une mémoire SDRAM.

Largeur de bande : Vitesse maximale de transfert des données.

2.2 Accès à la mémoire

Les broches suivantes, communes à la plupart des mémoires peuvent habituellement être gérées par l'interface parallèle en fonction de registres de configurations:

Broche	Description	Usage
ALE	Address Latch Enable : indique une adresse valide sur le bus d'adresse	Multiplexage des broches d'adresses et de données.
nCS	Chip Select : active un circuit de mémoire	Décodage d'adresse et activation des mémoires

[1] En dehors des instructions de configuration de l'interface évidemment, faites avant l'utilisation de l'interface parallèle.

nOE	Output Enable : active la lecture de la mémoire	Lire la mémoire. La mémoire écrit des données.
nWE	Output Enable : active l'écriture de la mémoire	Écrire la mémoire. La mémoire lit des données.
nLB nHB	Sélection des octets lus ou écrits à l'intérieur d'un mot	Pour les mémoires ayant des mots de plusieurs octets
Data	Données	Données
Address	Adresse	Adresse

Pour accéder aux mémoires, les interfaces parallèles doivent respecter des timings bien précis (voir le cours sur les mémoires et les chronographes). De ce fait, plusieurs registres permettent de configurer les vitesses et les temps de latence des interfaces parallèles.

> *Les **wait states** apparaissent lorsque l'interface parallèle (ou le microprocesseur) doit attendre un ou plusieurs coups d'horloge pour que la mémoire produise ou écrive une donnée.*

2.2.1 Gestion de mémoire DRAM ou SDRAM

Quelques interfaces parallèles, plus avancées, gèrent de la mémoire DRAM ou la mémoire SDRAM. Ces interfaces contrôlent des broches additionnelles permettant de contrôler ces mémoires:

DRAM :

Broche	Description	Usage
nCAS	Column Address Strobe	Indique que les lignes d'adresses désignent une colonne
nRAS	Row Address Strobe	Indique que les lignes d'adresses désignent une rangée
R/nW	Lecture ou écriture	Détermine si DRAM est lue ou écrite
Address	Adresse	Adresse
Data	Données	Données

SDRAM :

Broche	Description	Usage
CKE	Clock Enable : active l'horloge du bus	Active ou désactive l'horloge, peut aussi mettre la mémoire en veille
CLK	Clock : horloge du bus synchrone	Horloge de la mémoire pour communication
DQM	Data Q Mask : Désactive les broches de données	Fournit une méthode de gestion des collisions d'écriture et de lecture
nCAS	Column Address Strobe	Décrit la commande envoyée à la mémoire…
nRAS	Row Address Strobe	
nWE	Write Enable : active l'écriture de la mémoire	

nCS	Chip Select	
BA0 BA1	Bank select	

Si le temps le permet, le chronographe de l'interface parallèle du LM3S9B92 pour la lecture d'une mémoire SDRAM sera présenté en classe.

2.3 Accès aux périphériques

Les interfaces parallèles sont souvent utilisées pour communiquer avec plusieurs périphériques : ADC, DAC, afficheurs, imprimantes, interfaces pour bus de communication, FPGA (Field Programmable Gate Array), CPLD (Complex Programmable Logic Device) ou autres…

Une interface parallèle est préférée à un groupe de GPIO dans plusieurs circonstances. En fait, l'interface parallèle peut exécuter certaines tâches automatiquement et libérer le CPU. De plus, l'interface parallèle peut souvent être plus rapide que le CPU lisant des instructions pour manipuler un groupe de GPIO : elle n'a pas à lire et décoder des instructions entre chaque transition de broche.

Les principales forces d'une interface parallèle par rapport à un groupe de GPIO sont la configuration automatique des séquences d'activation/désactivation des broches de l'interface, la possibilité de mettre des données dans un tampon pour ajuster la vitesse entre le périphérique et le CPU et le DMA.

2.3.1 Files de gestion des octets (FIFO)

Une file est souvent utilisée lors de transferts de données d'un circuit intégré à un autre circuit intégré. La file est une structure de donnée contenant N mots où les premiers mots qui entrent sont les premiers mots qui sortent (First In, First Out).

Les points suivants sont à considérer lorsque l'on utilise des files :

- Quelles sont les vitesses d'entrées et les vitesses de sorties? La vitesse moyenne de sortie doit être plus grande que la vitesse moyenne d'entrée!
- Combien d'entrées sont nécessaires pour garantir, statistiquement, que la file ne sera jamais pleine?
    - Cela peut donner des calculs très élégants. Supposons par exemple une interface parallèle qui emmagasine des échantillons d'ADC dans une file à une vitesse de 1MHz. Supposons que la file ait 16 entrées, de telle sorte que l'interface parallèle remplisse la file en 16us. Supposons aussi que le microprocesseur vide la pile en 0.1us, lorsque l'interruption de l'interface parallèle survient (cette interruption se produit dès qu'un échantillon nouveau est acquis). Si le microprocesseur a 32 sources d'interruptions ayant une priorité égale à celle de l'ADC, que ces interruptions arrivent aléatoirement à une fréquence de 10MHz en moyenne (pour toutes les sources d'interruption ensemble) et que le traitement de ces interruptions

prend 1us en moyenne, quelle est la probabilité pour que la file se remplisse une fois à l'intérieur d'une journée?

- Si la file est pleine et qu'une nouvelle entrée arrive, que se passe-t-il?
    - La première entrée est effacée et toutes les entrées sont décalées vers la sortie pour faire une place.
    - La dernière entrée est écrasée par la nouvelle entrée (je ne connais pas de file qui gère les entrées superflues ainsi!!!).
    - La nouvelle entrée est perdue…
- Si la file est vide et qu'on demande une entrée, que se passe-t-il?

2.3.2 Interface parallèle et DMA

Habituellement, on peut configurer un canal de DMA pour transférer des données du périphérique connecté à l'interface parallèle vers la mémoire ou de la mémoire vers le périphérique.

2.4 Programmation d'interface parallèle

Pour programmer une interface parallèle, il faut généralement écrire les registres de contrôle suivant :

- Activation de l'interface parallèle
- Établissement du rôle des broches de l'interface
- Détermination de l'horloge de l'interface parallèle
- Détermination du mode d'opération de l'interface.
- Configuration des broches de l'interface en fonction de leur rôle
- Détermination des adresses attribuées à l'interface parallèle.

Lorsque tous ces paramètres sont établis, les accès à une mémoire via l'interface parallèle devraient se faire de façon transparente : un microcontrôleur lit une adresse attribuée à la mémoire connectée à l'interface parallèle comme il lit une adresse interne.

Lorsque tous ces paramètres sont établis, les accès aux périphériques via l'interface parallèle sont gérés en grande partie par l'interface elle-même. Le microprocesseur consulte ou écrit des registres de l'interface parallèle (comme le nombre d'éléments dans la file de l'interface) pour opérer le périphérique connecté à l'interface.