Jacob Fontaine
Philippe Spino

Spip2401
Fonj1903

Rapport App2

Présente à :
JEAN LAVOIE

January 30, 2018

Contents

1 Introduction.

Ce présent rapport a pour but de présenter les éléments de conception du système d'acquisition de données(SAD) développé par notre équipe. Le SAD a été conçue à l'aide d'un processeur embarqué LPC1768, soit la coquille de base du SAD. Le contenue de ce rapport explique les ressources mémoire et du processeur utilisées, les algorithmes de gestion des acquisitions, la synchronisation des tâches utilisées, une analyse des problèmes de conception et de déverminage lors du montage du prototype, une description des méthodes concernant la validation des fréquences d'échantillonnages des données et une analyse du prototype le plus optimal. De plus, une discusion sur les temps d'éxecution des codes base pour un inclinomètre. à la demande de la compagnie metrologie numérique.

2 Utilisation des ressources matérielles.

Pour ce présent prototype deux LCP1768 sont utilisés. Un des microcontrôleur est programmé pour emêttre de façon aléatoire des données soit numériquement ou bien analogiquement. Il est nommé DataSpammer. L'autre microcontrôleur utilise l'interface principale du SAD. Le DataSpammer, est réglé avec une fréquence d'opération mobile, c'est-à-dire qu'elle peut etre changé selon l'utilisateur. Pour le montage, nous avons 2 mbed qui sont branchés ensembles et le DataSpammer envoi ses données aléatoires via 2 port GPIO et la fréquence d'opératiopn passe de 10ms a 500ms. Sur le microcontrôleur qui a l'interface primaire, les led 1 et 2 servent a indiquer un changment de voltage sur les pattes analogiquer p19 et p20. Les led 3 et 4 servent a indiquer un événement numérique. Les pattes p21 et p22 sont utiliser.

L'interface principale utilise 4 tâches pour faire la gestion des acquisitions, une tâche pour chaque type d'entrée, une tâche pour l'horloge temps-réel(RTC) et une tâche pour la gestion de la propagation. Pour propager les données acquisent, une structure, nommée Event, est utilisé. Cette structure contient un byte non signé, un entier de 32bit et une struct temporelle. Pour propager les Event, une file globale, qui contient des Event, de type First In First Out(FIFO) est utilisé. Pour la gestion de l'accès mémoire de cette file, un mutex global est utilisé. La file est utilisé par les deux tâches d'acqusitions numérique et analogique. Par la suite, pour l'acquisition du temps lors d'une capture de données, une file globale FIFO est utilisé pour une demande du temps. un mutex est aussi utilisé pour la gestion de l'accès a cette file.

3 Algorithmes de gestion des acquisitions de données.

L'algorithme dévéloppé pour l'aquisition des données est divisé en 4 tâches différents. Numérique, Analogique et le RTC. La tâche pour l'acquisition numérique étant en priorisation haute, l'analogique en priorisation moyenne et le reste, soit la tâche pour le RTC, en priorisation basse.

3.1 Numérique

La tâche pour l'acqusition des données numérique vérifie s'il y a un changement de bit. Le temps pris pour qu'un bit puisse changer et être considéré un changement est de l'ordre des 50 ms. La fréquence d'opération de cet algorithme est 100ms. Pour ce faire, une tâche::wait est utilisé. cela permet d'être assuré que le tâche vas être à une fréquence de 100ms. Une boucle itérative sur 20 tours regarde les valeurs des pattes d'entrées digitales à toutes

les 5ms, lorsque les valeurs rester inchangé après une itération, un conteur est incrémenté. Si dès que le conteur atteind 10(10 * 5ms = 50ms),l'algorithme a fait une demande au mutex pour assurer l'accès a la file et évité les interlockage. Ensuite, une réquisition du temps est fait à la tâche du RTC. Le mutex du temps est acquis et un Event est crée avec le changement et le temps, et on l'ajoute à la file des Event. Une fois, l'Event dans la file, les mutex sont libérés.

3.2 Analogique

La tâche pour l'acquisition des données analogique vérifie s'il y a un changement de la moyenne des 5 dernière lecture. La définition d'un changement est lorsque la nouvelle moyenne est $\pm$ 12.5% de l'ancienne moyenne. La lecture des pattes analogique opère à 50ms et envoi la moyenne recalculer au 250ms. Une boucle itérative sur 5 tours additionne les valeurs lu et attend 50ms. Une fois 5 échantillions additionné ensemble, soit 250ms plus tard, la comparaison est fait. Si une nouvelle valeur est accepté, une acquisition du mutex de la file des Event est appelé et le mutex du temps est acquis. un Event est crée et inséré dans le file. Ensuite les mutex sont relâchés.

3.3 L'horloge temps-réel (RTC)

La tâche pour l'acquision de l'étampe de temps s'exécute a une fréquence d'opération de 50ms. Elle demande l'accès à la file du temps en tentent d'acquérir le mutex de la file du temps. Ensuite, elle crée un variable time_t et l'insère dans le file de temps. L'algorithme relâche ensuite le mutex de la file de temps.

4 Synchronisation des tâches CPU.

Pour la synchronisation des ressources et des tâches, des mutexs sont utilisé. Les tâches ont un ordre de priorité allans de haute à basse. Pour s'assurer que les tâches seront bien synchrone au départ cest de fixer le processus crée par la fonction main à une haute priorité, partir les tâches avec leur priorité respective et ensuite remettre le processus de la fonction main à une priorité normale. Cela permet de bien s'assurer que les opération critiques, tel que le démarage des tâches soit bien exécuté dans la fonction main. Ensuite, dans les tâches, l'utilisation de la fonction Thread::wait(ms). Cela nous permet de fixer la fréquence d'opération au valeur que l'on veut obtenir. elle permet aussi de réguler l'accès aux mutex

5 Déverminage

Il est certain que l'utilisation de l'oscilloscope était primordiale pour déverminer cet APP. Une difficulté se présentait donc devant le fait que nous devions toujours travailler dans un local contenant cet outil, car nous voulions valider nos données en entrés et en sortie. De plus, nous avons remarqué qu'il était difficile de savoir ce que contenait les files. Par exemple, nous avions un problème avec un timestamp, car il changeait de valeur entre 2 thread après être passé par une file. Nous ne pouvions voir que l'addresse du du timestamp, et non sa valeur. Pour faciliter les choses, il serait intéressant d'utiliser plus le sytsème d'expression de Keil, qui permet d'analyser le résultat d'une série de variable.

6 Méthode de validation des fréquences d'échantillonage.

Les fréquences d'échantillonages étant de 100ms et de 250ms pour le numérique et l'analogique respectivement, les informations trop vite, ce qui pourrait par exemple créer des évènement invalide alors qu'ils sont bien valide. En plus de faire le bon nombre de wait, comme expliquer dans notre algorithme plus haut, nous pouvions vérifier nos échantillions en allumant les leds sur réception d'un échantillion valide. Par contre, cela étant peu précis, nous avons également développer un autre programme qui permettais d'envoyer les données à un rythme précis. Cela, coupler à notre invite de commande, permettais de valider que nous avions bien reçu, par exemple, les 3 échantillions valide envoyés. Ainsi, nous avons pu confirmer le fonctionnement de nos fréquences.

7 Analyse de performances des Codes de Métrologie Numérique.

8 Conclusion.

9 Schémas.