Microcontrôleur

Projet : SDCARDSPI – bibliothèque d'accès à une carte SD

Ce projet a pour objectif d'accéder au contenu d'une carte SD par la liaison SPI. Toutes les cartes SD (SDHC et SDXC) sont capables de commniquer sur SPI pour des accès à faibles performances. Accessoirement, seul la communication en mode SPI n'est pas soumise à des droits d'accès.

1) Ressources

Le module PmodSD permet de connecter une carte SD au microcontrôleur par le bus SPI. Sa documentation est disponible là :
https://digilent.com/reference/pmod/pmodsd/reference-manual

les spécifications détaillées sont disponibles gratuitement là :
https://www.sdcard.org/downloads/pls/pdf/?p=Part1_Physical_Layer_Simplified_Specification_Ver8.00.jpg&f=Part1_Physical_Layer_Simplified_Specification_Ver8.00.pdf&e=EN_SS1_8

En bref, les transferts se font par blocs de 512 octets, la fréquence du bus SPI est de 25MHz maximum.

2) Fonctionnalités attendues

a) fonctions de base

Les fonctions attendues sont :

- Sdcard_init : mise en place de l'environnement et initialisation de la carte SD
- SD_read_bloc : la fonction lit un bloc sur la carte, elle reçoit un numéro de bloc sur 32 bits, et un char * qui pointe vers une zone de 512 octets pour stocker les données récupérées
- SD_write_bloc : la fonction écrit un bloc de 512 octets sur la carte SD. Un numéro de bloc sur 32 bit ainsi qu'un char * vers les données à envoyer sont fournis en argument.

b) fonctions secondaires

Si le temps le permet, les fonctions suivantes seront aussi untéressantes :

- proposer une fonction d'effacement (utilisant la commande associée)
- utiliser des commandes de transfert multiples pour augmenter les performances sur des échanges de plusieurs blocs consécutifs.

3) Performances

La performance de l'ensemble de l'architecture est un critère important. Notamment, la fourniture de fonctions bloquantes est un minimum, mais dans l'idéal, l'usage du processeur doit être réduit au stricte nécessaire.

Concernant la consommation électrique, un module GPS est TRES gourmand, il est donc particulièrement intéressant de gérer son alimentation pour ne pas qu'il soit alimenté en permanence.
Un bilan des ressources matérielles utilisées doit être tenu à jour.

4) Sdcard Quickstart

a) Initialisation

Il existe deux versions pour le protocole. En effet, des commandes ont été ajoutées pour le protocole V2. Ces commandes permettent de gérer les cartes SDHC et SDXC, mais on peut tout de même trouver des cartes SD tout court qui utilisent le protocole V2.

1. s'assurer que la carte n'est pas séléctionnée (CS = 1)
2. attendre 20 ms
3. envoyer au moins 9 fois la valeur 0xFF, alors que la carte n'est pas selectionnée
4. abaisser le CS
5. envoyer la commande *init* (0x40, 0x00, 0x00, 0x00, 0x00, 0x95)
6. attendre une réponse de type R1
7. envoyer la commande *cmd8* (0x48, 0x00, 0x00, 0x01, 0xAA, 0x87)
8. attendre uen réponse de type R7
9. si la réponse est « illegal command », c'est une v1 (SD), sinon, c'est une V2.x : SD (peu probable), SDHC ou SDXC
10. attendre que la carte soit prête :
    1. envoyer la commande (0x77, 0x00, 0x00, 0x00, 0x00, 0x01)
    2. attendre une réponse de type R1
    3. envoyer la commande (0x69, 0x40, 0x00, 0x00, 0x00, 0x01)
    4. attendre une réponse de type R1
    5. si la réponse est non nulle, la carte n'est pas prête, on recommence
11. tester si on a une carte SD ou SDHC :
    1. envoyer la commande (0x7A, 0x00, 0x00, 0x00, 0x00, 0x01)
    2. attendre une réponse R7 (sur 32 bits / 4 octets)
    3. si le bit 30 vaut 1 : c'est une carte SDHC, sinon, c'est une carte SD en protocole V2
12. La carte est prête, mais on n'a aucune idée de sa capacité. Pour connaître sa capacité, il faut consulter un registre de la carte, le registre CSD sur la carte. Ce registre fait 128 bits et existe lui aussi en deux versions.
    1. Envoyer la commande (0x49, 0x00, 0x00, 0x00, 0x00, 0x01)
    2. attendre une réponse sur 128 bits + CRC sur 16 bits
    3. l'analyse du registre CSD se fera avec la datasheet…
13. La carte est prête...

b) réponses

Les réponses aux commandes ne sont pas immédiates, il faut souvent attendre avant de les obtenir. La réception de la valeur 0xFF signifie que la réponse n'est pas encore prête. Une valeur différente (0xFE) indique que la réponse arrive juste après.

R1 : 1 octet :

bit 0 : IDLE
bit 1 : ERASE_RESET
bit 2 : ILLEGAL COMMAND
bit 3 ; CRC ERROR
bit 4 : ERASE ERROR
bit 5:ADDR ERROR
bit 6 : PARAM ERROR

R7 : 4 octet :

on se fiche un peu de la valeur renvoyée, ce qui compte surtout, c'est si la commande est reconnue ou si elle est illégale.

c) Lire des données

Chaque bloc de données est identifié par une adresse sur 32 bits. Pour les cartes SD, chaque octet est désigné par une adresse sur 32 bits. Chaque bloc de 512 octets est identifié par l'adresse de son premier octet. Pour les cartes SDHC/SDXC, ce sont les blocs eux-mêmes à qui on a affecté une adresse sur 32 bits.

Pour lire un bloc de 512 octets de données il faut :

1. envoyer la commande : 0x51 + adresse (MSB first) + 0x01
2. attendre la réponse de type R1.
3. s'il n'y a pas d'erreur, une autre réponse sur 512 octets (précédée de 0xFE) arrive au bout d'un certain temps. Ce bloc de données est suivit d'un CRC sur 16 bits.

Pour écrire, il faut

1. envoyer la commande 0x58 + adresse (MSB first) + 0x01
2. attendre la réponse de type R1
3. envoyer le bloc de 512 octets précédé de 0xFE
4. attendre la validation de la carte (0x05 ou 0xE5)
5. la carte est alors occupée le temps d'écrire les données. Tans qu'elle sera occupée, elle enverra 0x00, quand elle devient libre, elle recommence à répondre 0xFF