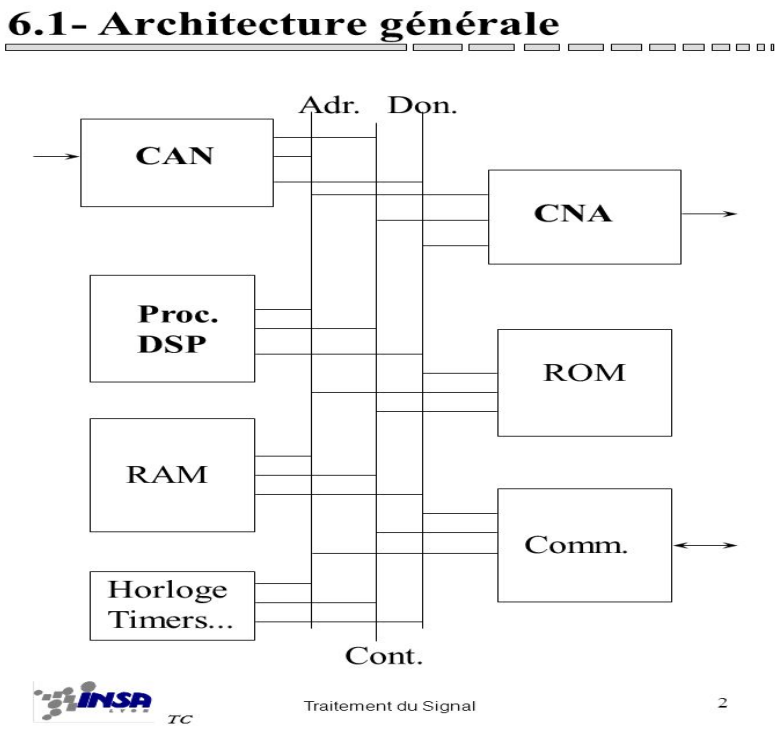
Architecture des DSP



1. Architecture d'un processeur

Un processeur est un composant intégré (en technologie C-MOS) ou une partie de composant susceptible d'exécuter des instructions selon un programme d'instructions préétabli sur des données numériques. Le programme, et parfois les données, sont enregistrés dans des mémoires numériques.

L'architecture est ce qui détermine d'emblée et de manière définitive les principales caractéristiques du processeur en particulier la rapidité d'exécution des instructions. Elles sont liées au nombre de bus internes, qui sont des liaisons parallèles à *N* bits, à la valeur même de *N*, aux opérations possibles sur les données transitant sur les bus et enfin à la puissance de calcul de l'unité centrale.

Les bus sont reliés à des registres temporaires pour stocker provisoirement les codes d'instructions, des adresses ou des données. D'autres types de registres existent:

1- les registres de contrôle ou d'état, qui ne sont pas stockés en mémoire car ils sont adressés directement par une instruction, par mesure de sécurité. Ces registres sont essentiels car ils déterminent le déroulement précis d'une instruction. Certains d'entre eux sont en lecture seule, d'autres en écriture seule.

2- les registres définissant les conditions de fonctionnement d'un port (série, DMA, hôte ...), des interfaces avec des mémoires de sortie ou du *Timer*, qui ont une adresse en mémoire de données. Les instructions s'exécutant au cœur du processeur appelé unité centrale, comportant dans le cas le plus classique, une seule unité arithmétique de traitement des données. À l'entrée des unités de calcul constituant l'unité centrale, les données sont chargées dans des registres temporaires qui, lors de la réalisation de l'instruction de calcul, vont être reliés à l'unité de calcul ; le résultat de l'opération sera

disponible dans un registre (temporaire) de sortie. L'architecture est conçue pour que les instructions puissent :

1- Charger les données dans un registre temporaire d'entrée en lisant le contenu d'une mémoire vive à une adresse donnée, les données transitant par un bus de données et les adresses pour la lecture par un autre bus ;

2- Effectuer une opération logique ou arithmétique sur une donnée ou entre deux données dans l'unité centrale;

3- Effectuer des tests sur le résultat obtenu (dépassement, signe...) ;

4- Modifier certains registres ;

5- Aller écrire un résultat à une adresse de la mémoire prévue.

Toutes ces instructions doivent en principe être exécutées en un seul cycle d'horloge.

L'architecture contient donc un noyau DSP si elle comporte une unité centrale laquelle est composée d'une ou plusieurs unités de traitement, chacune comportant des unités de calcul, dont l'ALU (Arithmetic and Logic Unit*)* le MAC *(***Multiplier and ACcumulator***)*, le décaleur à barillet *(***Barrel Shifter)** et enfin des multiplexeurs d'aiguillage des données. En plus de l'unité centrale, le noyau DSP comporte un **séquenceur**, pour envoyer les adresses des instructions enregistrées dans la mémoire programme, et un ou plusieurs **générateurs d'adresses** (parfois appelés pointeurs) agissant sur les bus d'adresses.

2. Architecture de Von Neumann et de Harvard

À chaque cycle d'horloge, le processeur sait par le compteur de programme l'instruction qu'il doit faire exécuter. Nous avons vu qu'il va chercher chaque instruction en mémoire, l'exécute avec les données correspondantes et retourne les données résultantes en mémoire.

Dans l'architecture de la machine de *Von Neumann,* le programme et les données sont enregistrés sur la même mémoire. Chaque instruction contient la commande de l'opération à effectuer et l'adresse de la donnée à utiliser, il faut donc souvent plusieurs cycles d'horloge pour exécuter une instruction. La *Figure 3.1* indique une architecture simple de Von Neumann, constituée d'un bus de données et de programme et d'un bus d'adresses.

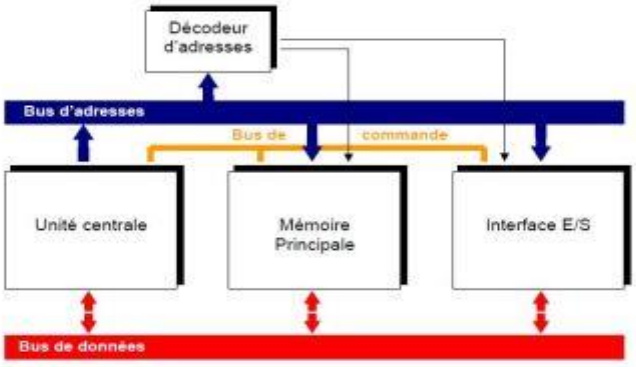


Figure 3.1 : Architecture de Von Neumann

On voit que les échanges s'effectuent de manière simple entre l'unité arithmétique et logique (ALU), c'est-à-dire l'unité centrale et la mémoire unique, par un bus transitant les codes de programme et les données. On a ainsi des données « collées » aux instructions. Les microprocesseurs et beaucoup de microcontrôleurs utilisent cette architecture car elle est très souple pour la programmation.

Dans l'architecture dite *de Harvard* (car mise au point dans cette université américaine en 1930), on sépare systématiquement la mémoire de programme de la mémoire des données : l'adressage de ces mémoires est indépendant. La *Figure 3.2* indique une architecture simple de Harvard, constituée d'un bus de données, d'un bus de programme et de deux bus d'adresse.

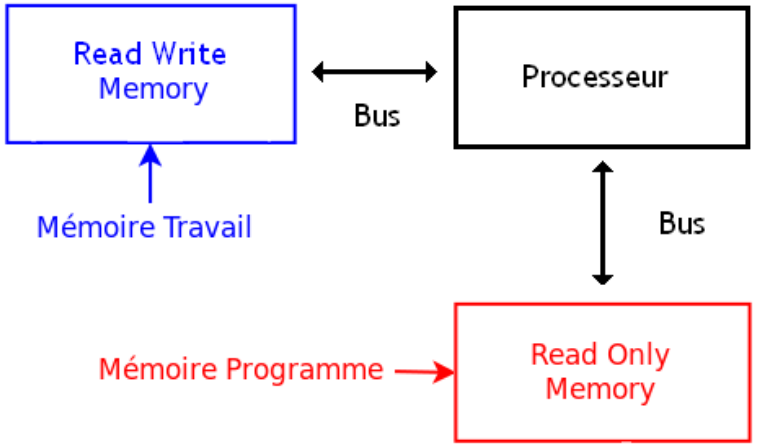


Figure 3.2 : Architecture de Harvard

On voit que les échanges s'effectuent de manière double entre l'unité centrale et les deux mémoires, ce qui permet une grande souplesse pour l'enregistrement et l'utilisation des données. D'ailleurs, la mémoire de programme est également utilisée en partie comme mémoire de données pour obtenir encore plus de possibilités de traitement avec des algorithmes complexes.

L'architecture généralement utilisée par les microprocesseurs est la structure Von Neuman (exemples : la famille Motorola 68XXX, la famille Intel 80X86). L'architecture Harvard est plutôt utilisée dans des microprocesseurs spécialisés pour des applications temps réels, comme les DSP. Il existe cependant quelques rares DSP à structure Von Neuman. La raison de ceci est liée au coût supérieur de la structure de type Harvard. En effet, elle requiert deux fois plus de bus de données, d'adresses, et donc de broches sur la puce. Or un des éléments augmentant le coût de productions des puces est précisément le nombre de broches à implanter.

Pour réduire le coût de la structure Harvard, certains DSP utilisent l'architecture dite « Structure de Harvard modifiée ». À l'extérieur, le DSP ne propose qu'un bus de données et un bus d'adresse, comme la structure Von Neuman. Toutefois, à l'intérieur, la puce DSP dispose de deux bus distincts de données et de deux bus distincts d'adresses. Le transfert des données entre les bus externes et internes est effectué par multiplexage temporel.

3. Utilisation de pipelines

Tout d'abord, expliquons comment fonctionnerait un processeur sans Pipeline. Une instruction qui arrive est découpée en plusieurs étapes, disons 5 pour respecter un modèle standard.

- IF (Instruction Fetch) charge l'instruction à exécuter dans le pipeline.
- ID (Instruction Decode) décode l'instruction et adresse les registres.

- EX (Execute) exécute l'instruction (par la ou les unités arithmétiques et logiques).
- MEM (Memory), dénote un transfert depuis un registre vers la mémoire dans le cas d'une instruction du type STORE (accès en écriture) et de la mémoire vers un registre dans le cas d'un LOAD (accès en lecture).
- WB (Write Back) stocke le résultat dans un registre. La source peut être la mémoire ou bien un registre.

Pour effectuer une étape (on parle aussi d'étage), il faut un cycle d'horloge. Pour traiter une instruction, il faut passer par ces 5 étapes et ensuite le processeur passe à la prochaine instruction et ainsi de suite...
Ci dessus, nous avons 3 instructions qui arrivent, donc 3 instructions * 5 (étapes) = 15 étapes au total donc 15 cycles d'horloges pour executer ces 3 instructions.

Pour améliorer les performances de l'unité de traitement, les DSP les plus récents utilisent la méthode du pipeline. Elle consiste à imposer un ordre et un rythme dans le déroulement des instructions de manière à optimiser en rapidité leur exécution. En un cycle processeur, les opérations élémentaires suivantes peuvent être exécutées en parallèle :

6. Aller chercher l'instruction en mémoire programme (*Fetch*) ;

Ici c'est la lecture de l'instruction (Instruction Fetch)

La prochaine instruction à exécuter est chargée à partir de la case mémoire pointée par le compteur de programme PC dans le registre d'instruction IR. Ensuite le compteur de programme est incrémenté pour pointer sur l'instruction suivante.

7. Réaliser le décodage de l'instruction, et des adresses des opérandes (*Decode*) ;
8. Lire les opérandes en mémoire de données (*Read*) ;

Ces étapes consistent à préparer les arguments de l'instruction pour l'étape suivante où ils seront utilisés. Ces arguments sont placés dans deux registres A et B. Si l'instruction utilise le contenu de un ou deux registres, ceux-ci sont lus et leurs contenus sont rangés en A et B. Si l'instruction contient une valeur immédiate, celle-ci est étendue (signée ou non signée) à 16 bits et placée dans le registre B. Pour les instructions de branchement avec offset, le contenu de PC est rangé en A et l'offset étendu dans B. Pour les instructions de branchement avec un registre, le contenu de ce registre est rangé en A et B est rempli avec 0.

4. Exécuter l'opération et écrire le résultat (*Execute*).

Exécution de l'instruction (Instruction Execution)

Cette étape utilise l'unité arithmétique et logique pour combiner les arguments. L'opération précise réalisée par cette étape dépend du type de l'instruction.

Le principe de pipeline consiste à découper le travail en tâches élémentaires de même durée pour permettre leur réalisation en parallèle. Il faut prévoir des registres tampon entre chaque opération élémentaire.

Pour comprendre le mécanisme du pipeline, il est nécessaire au préalable de comprendre les phases d'exécution d'une instruction. Les phases d'exécution d'une instruction pour un processeur contenant un pipeline « classique » à 5 étages sont les suivantes :

- **LI** : (*Lecture de l'Instruction* (en anglais *FETCH instruction*) depuis le cache ;
- **DI** : *Décodage de l'Instruction* (*DECODe instruction*) et recherche des opérandes (Registre ou valeurs immédiate);
- **EX** : *Exécution de l'Instruction* (*EXECute instruction*) (si ADD, on fait la somme, si SUB, on fait la soustraction, etc.);
- **MEM** : *Accès mémoire* (*MEMory access*), écriture dans la mémoire si nécéssaire ou chargement depuis la mémoire ;
- **ER** : *Ecriture* (*Write instruction*) de la valeur calculée dans les registres.

Les instructions sont organisées en file d'attente dans la mémoire, et sont chargées les unes après les autres.

Grâce au pipeline, le traitement des instructions nécessite au maximum les cinq étapes précédentes. Dans la mesure où l'ordre de ces étapes est invariable (LI, DI, EX, MEM et ER), il est possible de créer dans le processeur un certain nombre de circuits spécialisés pour chacune de ces phases.

L'objectif du pipeline est d'être capable de réaliser chaque étape en parallèle avec les étapes amont et aval, c'est-à-dire de pouvoir lire une instruction (LI) lorsque la précédente est en cours de décodage (DI), que celle d'avant est en cours d'exécution (EX), que celle située encore précédemment accède à la mémoire (MEM) et enfin que la première de la série est déjà en cours d'écriture dans les registres (ER).

ti	t1	t2	t3	t4	t5

Tableau 3.1 : Principe de pipeline

Sur le schéma ci dessus, on peut voir qu'à :
à t1 on va pouvoir charger le premier étage, le chargement de la première instruction.
à t2 on décode la première instruction et on charge la seconde instruction.
à t3 on execute la première instruction, on décode la 2ème et on charge la troisième.
à t5 tous les étages sont occupés : la première instruction utilise l'étage de Write back, la deuxième l'étage MEM, la troisième l'étage EX etc...

Donc à t=10 on a fini d'exécuter nos 5 instructions.
Ce que l'on peut vite comprendre, c'est que cette technique permet d'accélerer le traitement des instructions. En 9 cycles d'horloge, 5 instructions sont traitées alors que sans Pipeline, il faudrait 5*5= 25 cycles d'horlorge.
On augmente ainsi clairement les performances d'un processeur en mettant en place le pipeline. En exploitant donc cette notion de pipeline, on peut donc se dire que l'on peut rajouter des pipelines pour pouvoir aller encore plus vite, c'est ce qui a été mis en place dans des processeurs dits superscalaires : on met en parallèle plusieurs pipelines pour multiplier les performances de façon linéaire. Le principe du pipeline est ainsi comparable avec une chaîne de production de voitures. La voiture passe

d'un poste de travail à un autre en suivant la chaîne de montage et sort complètement assemblée à la sortie du bâtiment. Pour bien comprendre le principe, il est nécessaire de regarder la chaîne dans son ensemble, et non pas véhicule par véhicule. Il faut ainsi 3 heures pour faire une voiture, mais pourtant une voiture est produite toute les minutes !

Il faut noter toutefois qu'il existe différents types de pipelines, de 2 à 40 étages, mais le principe reste le même.

- # L’architecture du DSP Tms320c6713

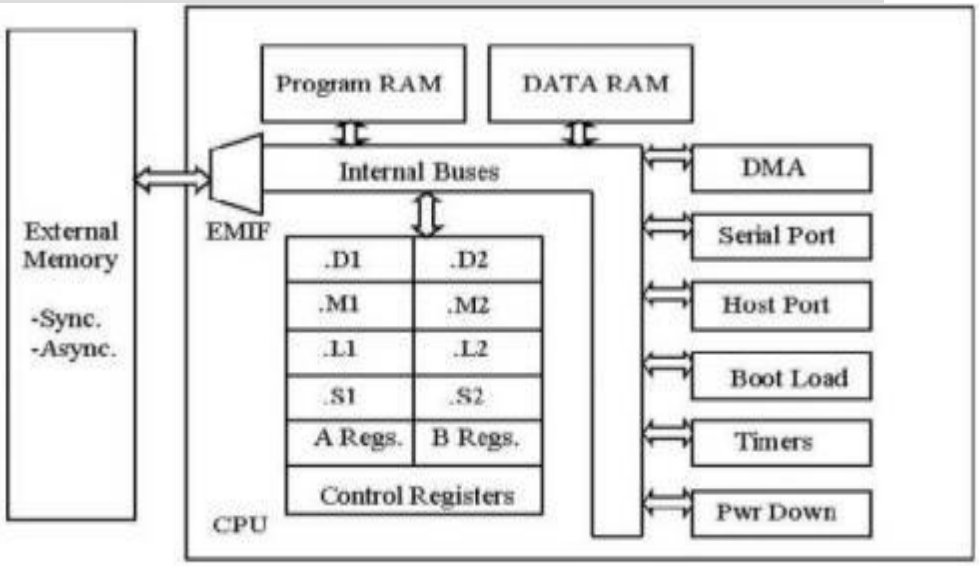


Figure 2.2 - Bloc diagramme simplifié de la famille TMS320C67xx.

-

4.1Généralités

L’ensemble C6713 DSK (Development System Kit) est un outil de développement qui permet à des utilisateurs de mettre au point et de tester des applications utilisant le processeur de traitement de signal Texas Instruments (TI) TMS320C6713. Autour de ce DSP sont connectés une grande variété de périphériques permettant une large gamme d’applications en traitement numérique du signal. La carteTMS320C6713 : DSP de la gamme TI travaillant à 225 MHz. Il est connecté à ses périphériques par un bus de 32 bits. Un « codec » stéréo AIC23 dédié à l’interfaçage pour les applications « audio ». Il comprend les convertisseurs CAN pour la capture de signaux (LINE IN, MIC IN) et les convertisseurs CNA pour l’exportation de signaux (LINE OUT, HP OUT). Le dialogue entre le processeur et le codec est réalisé par un multiplexeur McBSP (Multi-channel Buffered Serial Port), c'est-à-dire un multiplexeur à port série. En réalité, intégrés avec le processeur, il y a deux multiplexeurs : McBSP0 qui est prévu pour le contrôle du

codec (programmation de fonctions et des tâches) et McBSP1 qui est chargé de l'échange des données collectées ou à transmettre.

4.2 L'architecture du Tms320c6713

Comme le montre la figure ci-dessous la carte TMS320C6713 est constituée de trois parties principales :

- L'unité centrale de traitement CPU.

- Les périphériques.

- La mémoire

4.2.1-Unité centrale de traitement (CPU) :

Le CPU est constitue d'une unité de contrôle de programme, de deux unites fonctionnelles, de deux blocs de 16 registres de 32 bits, de contrôleurs d'interruptions et d'autres éléments.

A- Unité de contrôle de programme :

Elle est constituée des éléments suivants,

- Unité **"fetch"** programme : Elle a pour rôle récupérer les programmes. Cette opération se déroule en quatre phases :

1- **Phase PG:** l'adresse du code est générée.

2- **Phase PS :** l'adresse est envoyée à la mémoire.

3- **Phase PW:** l'attente de lecture du code de la mémoire.

4- **Phases PR** : la lecture du code.

-Unité "dispatche" de l'instruction: le code récupéré de la mémoire est affecté à 1'unité fonctionnelle associée.

-Unité de décodage de l'instruction: elle a pour rôle de décoder l'instruction.

B- Unités fonctionnelles :

Le CPU contient huit unités: fonctionnelles divisées en deux parties 1 et 2. Leurs fonctions sont les suivantes:

-Unités .Ml et .M2 : ces unités sont dédiées à la multiplication.

- Unités .L1 et .L2 : ces unités sont dédiées à l'arithmétique et la logique.

- Unités .Dl et .D2 : ces unités sont dédiées au chargement, la sauvegarde et calcul d'adresse.

- Unités .S1 et .S2: ces unités sont dédiées pour le décalage de bit, l'arithmétique, la logique et le branchement

C-Registre :

Le CPU contient 32 registres de 32 bits divises en deux blocs égaux : registre fichier A (A0-A15) et registre fichier B (B0-B15), leurs fonctions sont réparties comme suit :

-**Les registresA1-A2 et B0-B1-B2**: ils sont utilisés comme registres conditionnels.

-**Les registresA4-A7 et B4-B7**: ils sont utilisés pour adressage circulaire.

- Les registres AO-A9, BO-B2 et B4-B9 : ils sont utilisés comme registres temporaires.

- Les registres A10-A15 et B10-B15 : ils sont utilisés pour la sauvegarde et la restitution de données d'un sous-programme.

A ces 32 registres s'ajoutent les registres de contrôles et d'interruptions.

4.2.2- Les périphériques du TMS320C6713 :

Le TMS320C6713 a plusieurs périphériques qui sont :

- **Le contrôleur DMA** : Il permet sans l'aide du CPU de transférer des données entre les espaces mémoire (interne, externe et des périphériques). Il a quatre canaux programmables et un autre canal auxiliaire.

- **Le contrôleur EDMA** : Il permet le transfert des données entre les espaces mémoire comme le DMA. Il a 16 canaux programmables.

- **L'interface port hôte HPI**. Il donne au processeur hôte un contrôle total pour un accès direct de l'espace mémoire du CPU et à la cartographie de la mémoire des périphériques du DSP.

- **Deux McBSP qui sont des ports séries multi-canaux protégés**. Ils permettent la communication avec les périphériques externes. Ils ont la même structure. Ils supportent une communication full-duplex.

- **L'interface de mémoire externe EMIF** : Il permet l'interface avec plusieurs éléments (mémoires) externes.

- **Les compteurs** : Le DSP possède deux compteurs qui peuvent être synchronisés par une source interne ou externe et ils sont utilisés comme générateurs de pulsations, compteurs d'événements externes, interrupteurs du CPU après l'exécution de tâches et déclencheur du

DMA/EDMA.

- Les interruptions : l'ensemble des périphériques contient jusqu'à 32 sources d'interruptions.

4.2.3- La structure de la mémoire :

Le TMS320C6713 basé sur l'architecture de Havard modifiée utilise une mémoire externe et une mémoire interne (Figure 4.5).

- La mémoire externe occupe les espaces CEO, CE1, CE2, CE3.

- La mémoire interne a une taille de 260 KB qui est décomposée en deux niveaux :

*Le niveau (L1) est constitué de deux mémoires caches de 4 KB chacune, (L1P) qui est utilisée pour les programmes et (L1D) qui est utilisée pour les données.

* Le Niveau (L2) est composé de 256 KB de mémoire partagée entre mémoire des données et mémoire de programmes.

Présentation de la carte SDK TMS320C6713 DSK (C6713 DSK)

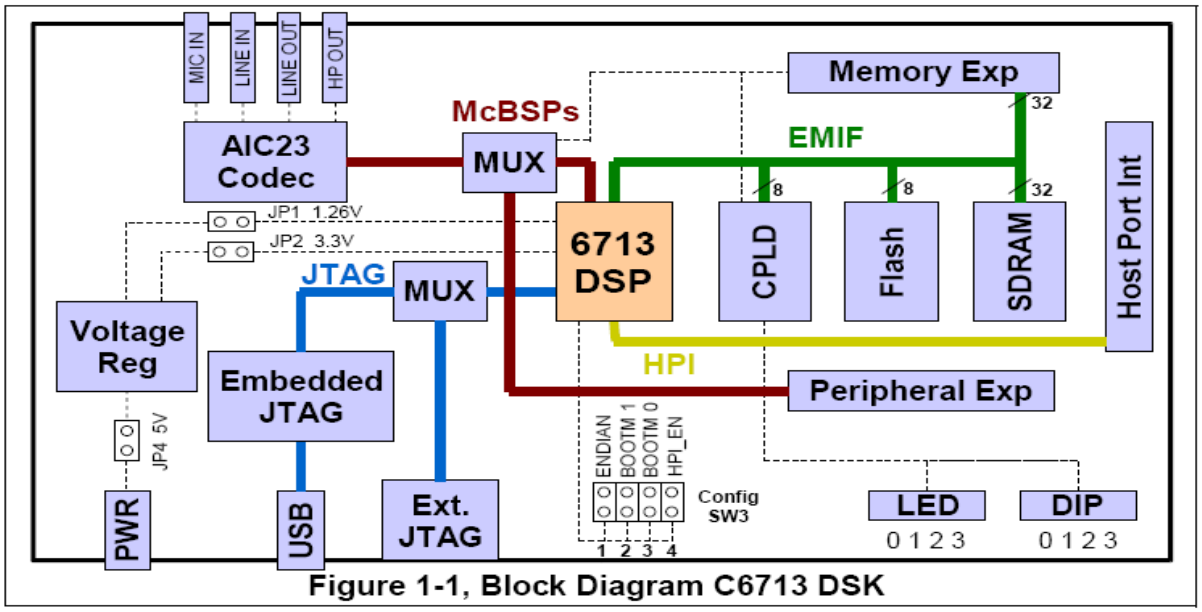


Figure 1-1, Block Diagram C6713 DSK

La carte d'évaluation DSK est puissante, relativement peu coûteuse, avec les outils de support matériels et logiciels nécessaires pour le traitement du signal en temps réel. Il s'agit d'un système DSP complet. Cette carte comprend le processeur de traitement numérique du signal, le TMS320C6713 à virgule flottante et un codec stéréo TLV320AIC23 (AIC23) de 32 bits pour l'entrée et la sortie (Figure 2.4).

Le codec AIC23 « utilisant une technologie sigma-delta » fonctionne comme un CAN pour les entrées analogiques et comme un CNA pour les sorties numériques du DSP. Il se connecte à une horloge système de 12 MHz. Le taux d'échantillonnage variable de 8 à 96 kHz et peut être réglé facilement.

La carte comporte un emplacement libre pour l'ajout d'un périphérique ou d'une carte additionnelle ("daughter card"), un emplacement pour ajouter de la mémoire, quatre LED ("Light-Emitting Diodes") et quatre DIP switchs ("Dual In-line Pin") programmables.

La carte DSK comprend 16 MB de mémoire synchrone dynamique à accès aléatoire (DRAM) et 256 kB de mémoire flash. Le DSK fonctionne à 225 MHz, il intègre un régulateur de tension qui fournit 1,26 V pour le noyau C6713 et 3,3 V pour la mémoire et les périphériques. Le DSK dispose également de quatre prises audio jacks de 3,5 mm, deux pour les entrées : microphone (mono) et "line in" (stéréo), et deux pour les sorties : "speaker" (stéréo) et "line out" (stéréo). En fait les deux entrées (respectivement les deux sorties) renvoient les signaux au même port physique, c'est-à-dire au même signal d'entrée (respectivement de sortie). La seule différence entre microphone et "line in" (respectivement "speaker" et "line out") réside

dans les impédances des ports. Autrement dit, on a quatre prises audio, mais une seule entrée et une seule sortie, chacune disponible avec deux impédances différentes TMS320C6713 DSK (C6713 DSK)

La carte d'évaluation DSK est puissante, relativement peu coûteuse, avec les outils de support matériels et logiciels nécessaires pour le traitement du signal en temps réel. Il s'agit d'un système DSP complet. Cette carte comprend le processeur de traitement numérique du signal, le TMS320C6713 à virgule flottante et un codec stéréo TLV320AIC23 (AIC23) de 32 bits pour l'entrée et la sortie (Figure 2.4). Le codec AIC23 « utilisant une technologie sigma-delta » fonctionne comme un CAN pour les entrées analogiques et comme un CNA pour les sorties numériques du DSP. Il se connecte à une horloge système de 12 MHz. Le taux d'échantillonnage variable de 8 à 96 kHz et peut être réglé facilement. La carte comporte un emplacement libre pour l'ajout d'un périphérique ou d'une carte additionnelle ("daughter card"), un emplacement pour ajouter de la mémoire, quatre LED ("Light-Emitting Diodes") et quatre DIP switchs ("Dual In-line Pin") programmables. La carte DSK comprend 16 MB de mémoire synchrone dynamique à accès aléatoire (DRAM) et 256 kB de mémoire flash. Le DSK fonctionne à 225 MHz, il intègre un régulateur de tension qui fournit 1,26 V pour le noyau C6713 et 3,3 V pour la mémoire et les périphériques.

Le DSK dispose également de quatre prises audio jacks de 3,5 mm, deux pour les entrées : microphone (mono) et "line in" (stéréo), et deux pour les sorties : "speaker" (stéréo) et "line out" (stéréo). En fait les deux entrées (respectivement les deux sorties) renvoient les signaux au même port physique, c'est-à-dire au même signal d'entrée (respectivement de sortie). La seule différence entre microphone et "line in" (respectivement "speaker" et "line out") réside dans les impédances des ports.

Caractéristiques du DSK C6713

➢ DSP TMS320C6713 fonctionnant à 225 MHz.

➢ Codec stereo de 16 bits (AIC23)

➢ Quatre (4) connecteurs permettent des entrées et des sorties analogiques: • MIC IN : entrée de microphone, • LINE IN: entrée analogique "ligne" • LINE OUT: sortie analogique "ligne" • HEADPHONE: sortie écouteurs/casque (multiplexée avec l'entrée ligne).

➢ 16 Mbytes de SDRAM ➢ 512 Kbytes de mémoire flash (256 Kbytes pour une configuration par défault)

➢ 4 LED et 4 interrupteurs (DIP switches)

➢ Configuration de la carte par programmation du CPLD ➢ Deux connecteurs de 80 broches permettent des extensions de périphériques et mémoire externes.

➢ Un émulateur JTAG intégré jumelé avec la connectivité USB

References

[1] M. Correvon(2010). Introduction aux DSP orientés & applications industrielle - ***Notes du cours*** [Présentation PDF]. Repéré dans le site www.iai.heig-vd.ch.

[2]dsk6713_TechRef de TEXAS INSTRUMANT (2006), Datasheet du C6713 DSK.

[3] Rulph Chassaing(2005). Digital Signal Processing and Applications with the C6713 and C6416. USA, New Jersey: John Wiley & Sons, Inc.

[4] Sylvain MONTAGNY(2008). Microprocesseurs microcontrôleurs - *Notes du cours* [Présentation PDF]. Repéré dans le site *www.sfa.univ-savoie.fr*.

[5]TMS320 datasheet pdf datenblatt - Texas Instruments