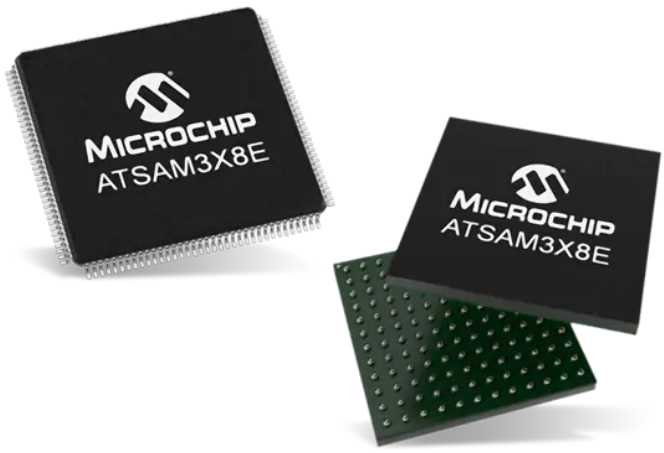
ES0EN317 Modélisation et langage système

ATMEL SAM3X

25 février 2021

Girones - Perez - Aucher - Lesieur - Lam - Boisseau



Introduction:

Le but de ce module est de recréer un processeur ATMEL SAM3X/A en TLM. Ce travail est réalisé par 4 groupes qui se répartissent 4 parties du processeur, à savoir : le timer, l'uart, le gpio et le PMC. Notre groupe se chargera de la partie UART. Nous réaliserons dans un premier temps une version simplifiée et en fonction du temps restant nous ajouterons toutes les interruptions et fonctionnalités.

Concept:

UART, pour Universal Asynchronous Receiver Transmitter, est un émetteur-récepteur asynchrone universel. Une trame UART est constituée des bits suivants :

-un bit de start toujours à 0 : servant à la synchronisation du récepteur

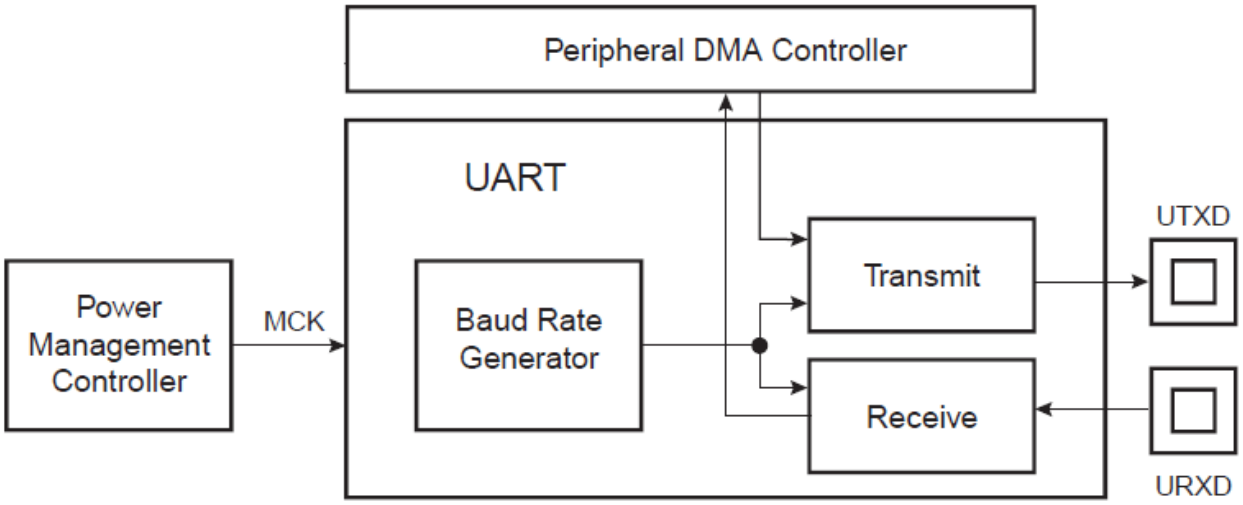
-Données : la taille des données est fixée à 8 bits dans notre cas. Les bits envoyés du LSB (bit de poids faible) au MSB (bit de poids fort).

-Parité : Paire, Impaire, forcée à 1 ou a 0 (optionnelle)

-Fin : Un bit de stop, toujours à 1.

Le niveau logique de repos est le 1.

Le bloc UART au sein du ATMEL SAM3 peut se simplifier au diagramme suivant:

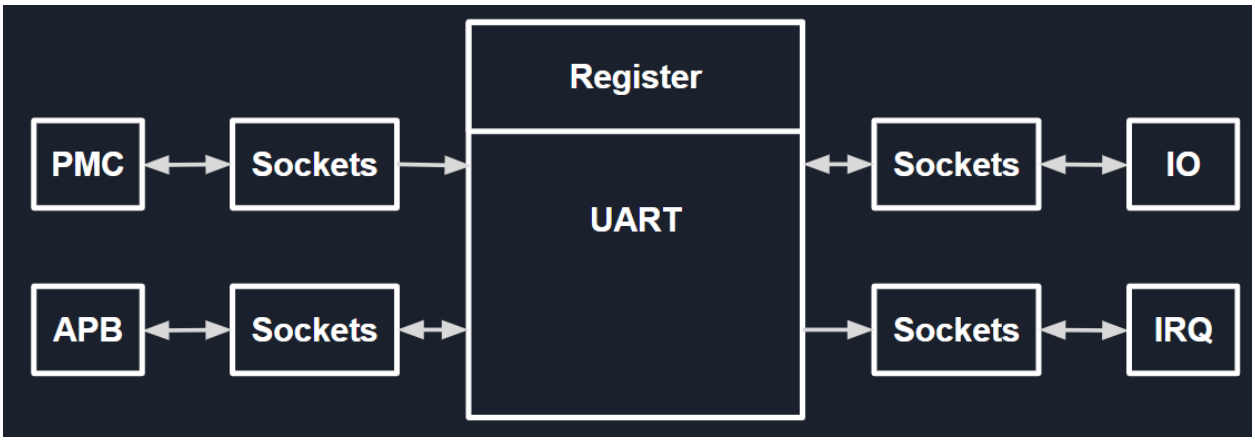


Réalisation:

Nous allons répartir le travail en 3 points clés ;

1-Architecture UART

Voici ci-dessous une représentation de l'architecture du code. On y trouve au centre notre interprétation du bloc UART décrit dans la datasheet et codé dans une classe Uart. Ce fichier se base sur une classe contenant tous les registres. Tout cela entouré de sockets pour la communication avec les périphériques permettant le bon fonctionnement du bloc.



2-Sockets

Nous avons ensuite implémenté 4 sockets: 2 doubles sens pour les IO et le bus interne APB, et 2 unidirectionnels avec le PMC en entrée et les IRQs en sorties.

3-Testbench

Pour le testbench nous avons codé un bloc qui vient se greffer sur les 4 sockets du projets. Dans un premier temps, il génère des stimulis sur les sockets du PCM, APB et des IOs. Dans un deuxième temps, il scrute les sorties sur les sockets des IO, IRQ et APB. Enfin, on analyse les délais et valeurs de sorties pour pouvoir valider le bon fonctionnement.

Difficultés:

1-Niveaux de C++

Comme la plupart des groupes, n'ayant pas de formation à proprement parler de C++ nous avons eu beaucoup de lacunes pour nous immiscer dans le projet. En effet, il est compliqué de concevoir une architecture et des fonctions complexes en n'ayant que quelques bases. Nous avons eu de nombreux blocages sur des erreurs de code nécessitant des corrections de l'intervenant.

2-Sockets

Les Sockets étant quelque chose de nouveau pour nous tous, nous avons eu beaucoup de mal à s'imprégner de la structure et du principe même du fonctionnement des sockets en TLM. Nous sommes alors repartis d'un exemple de sockets donné en cours ce qui a ensuite permis de coder plus simplement le bloc.

3-Mauvaise architecture

Au démarrage du projet nous avons voulu recréer le bloc UART en reprenant l'architecture de Bloc UART. Ne maîtrisant pas parfaitement le C++, il se trouve que reprendre cette architecture compliquait grandement le codage du projet.

Conclusion:

Ce module nous a permis d'implémenter en C++ un bloc UART d'un processeur ATMEL SAM3X/A en TLM. Nous avons pu comprendre l'enjeux du système C lors des processus de créations d'architecture de composant complexe. Durant les séances nous avons dû apprendre à travailler rapidement en équipe étant donné le temps imparti restreint.