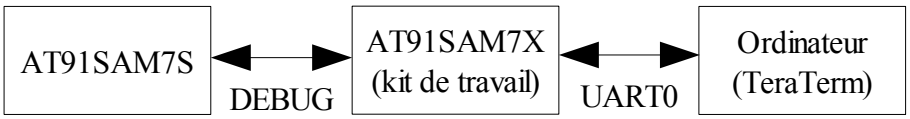
Microprocesseur

- Projet -
Gestion de flux de données

Le principe du projet de cette année est de programmer un système capable de gérer un flux entrant de données. L'affichage se fera par l'intermédiaire de la liaison UART et du terminal virtuel TeraTerm. Le flux de données entrant arrivera par la liaison DEBUG. Cette liaison utilise le même principe que la liaison UART série, mais en utilisant une architecture simplifiée. A notre niveau, elles sont équivalentes (mêmes noms de registres, mêmes masques...). Seuls quelques champs de registres ne sont pas implémentés, vérifiez que vous n'en avez pas besoin.

1) Communications

La carte est à connecter selon le schéma suivant :



Les paramètres de communication entre le kit AT91SAM7X et l'ordinateur sont libres. Pour la communication entre le deux kits, ils sont les suivants:

- 200 kbps
- 8 bits par mot
- 1 bit de stop
- pas de parité
- pas de contrôle de flux

Le kit AT91SAM7S répond aux commandes suivantes :

- Toutes les secondes, il envoie le texte "Allo ?↵"
- Lorsqu'on lui envoie le caractère '+', le texte envoyé périodiquement est modifié.
- Lorsqu'on lui envoie le caractère '**?**', il répond un texte plus conséquent, puis arrête de communiquer pendant 5 secondes
- Lorsqu'on lui envoie le caractère '**a**', il envoie une image en ASCii Art (caractères qui représentent une image). Ce transfert se fait en une seule fois. Lorsque l'envoi est effectué, le système arrête de communiquer pendant 20 secondes.
- Lorsqu'on lui envoie le caractère '**b**', il répond une autre image en ASCii Art, mais cette fois, les caractères sont envoyés dans le désordre, et petit à petit. Pour chaque caractère de l'image, le système envoie d'abord le caractère nul (valeur ASCii = 0). Ensuite, il envoie un numéro de colonne puis un numéro de ligne (codés sur un octet chacun). Finalement, le kit envoie un caractère à afficher aux coordonnées précédentes.
  Dans ce mode, l'envoi de chaque caractère se fera donc sous la forme :
  <0><colonne><ligne><caractère>
  Après cette commande, le kit arrêtera d'envoyer "Allo ?↵" pendant 10mn
- la commande ' ' (espace), force le kit à recommencer les appels "Allo ?↵"

2) Étapes de construction du projet

a) Réception de données / affichage

Dans un premier temps, vous devez afficher les données qui sont envoyées par le kit AT91SAMS sur le terminal de l'ordinateur. Pour cela, il "suffit" de renvoyer vers le terminal tous les caractères qui arrivent depuis le kit AT91SAM7S.

b) Communication bidirectionnelle

Dans un second temps, lisez les caractères frappés dans le terminal. Si un caractère correspond à une commande, renvoyez le au kit AT91SAM7S, sinon, affichez 'erreur'. Cette partie ne doit pas impacter le renvoi des messages pour affichage de la partie a) qui doit rester fonctionnel.

Si cette partie fonctionne, vous devriez être en mesure d'obtenir une réponse cohérente aux instructions '**?**', '**+**' et '**a**'.

c) Affichage progressif

Cette partie consiste à gérer l'affichage de la commande '**b**'. Une fois que vous avez transmis le caractère 'b', le format des données à transférer aura changé. Il faudra décoder le contenu des trames pour positionner correctement le curseur et générer l'affichage du caractère au bon endroit. Si rien n'est reçu pendant plus de deux seconde, c'est que l'animation est terminée, il faut repositionner le curseur en haut à gauche.

3) Précisions techniques

a) Conception d'une FIFO (au cas où...)

Pour réaliser une FIFO, il faut au moins un tableau qui contienne les caractères, un indice pour repérer la case d'écriture et un autre pour la case de lecture. Une écriture se fait en affectant la case d'écriture, puis en incrémentant son indice modulo le nombre d'éléments dans le tableau. Une lecture s'effectue avec les mêmes étapes : lecture de l'élément dans le tableau puis incrément de l'indice. Si les indices d'écriture et de lecture sont identiques, la FIFO est soit pleine, soit vide. À vous de choisir comment vous préférez gérer ces cas de figure (on peut par exemple interdire de remplir complètement la file pour clarifier la situation). Le plus pratique est de réaliser des fonctions de lecture et d'écriture qui allègent le code C et la compréhension du programme.

b)gestion du curseur

Pour positionner le curseur à un endroit déterminé, la séquence suivante doit être envoyée au terminal :

- un caractère 0x1B
- le caractère '['
- le numéro de ligne au format ASCII (un ou plusieurs caractères entre 0x30 et 0x39)
- le caractère ';'
- le numéro de colonne au format ASCII (un ou plusieurs caractères entre 0x30 et 0x39)
- le caractère 'H'

ATTENTION, la ligne (position y) est envoyée avant la colonne (position x) pour des raisons techniques historiques. La numérotation se fait à partir de (1,1)

La séquence abrégée 0x1B puis '[' et 'H', permet de revenir en haut à gauche du terminal.

4) Évaluation

Le rapport est à rendre pour le Mardi 29 avril 13h (précises) et n'excédera pas 15 pages utiles (6 pages devraient être largement suffisantes si la rédaction est précise et efficace). Vous pouvez joindre le code commenté en annexe, mais il est moins intéressant que l'algorithme, les structures choisies et le rôle qui leur est attribué. Il présentera dans un premier temps les choix techniques qui ont été faits, ainsi que les algorithmes utilisés, et les fonctions réalisées en interruptions s'il y en a. Pour chaque partie du programme, il faut justifier le choix de faire une implémentation en scrutation ou interruption. Si plusieurs sources d'interruptions sont actives en même temps, il faudra argumenter l'attribution de leur priorité. Si vous utilisez des machines d'état, il sera intéressant de les représenter.

La configuration de chacun des modules utilisés sera expliquée en détail (pourquoi telle valeur dans tel registre, ce que ça implique au niveau du système). Si des valeurs utilisées sont le résultat de formules, elles sont à expliciter.

Toute remarque effectuée par votre encadrant est prioritaire sur ce sujet.