|  |
| --- |
|  |
| **Laboratoire 6**  **Mesure du temps de réaction** |
| par :  Patrick Maillard  Guillaume Gonin  pour :  Etienne Messerli (Professeur)  Anthony Convers (Assistant) |
| HEIG-VD  Département TIC  20.12.2024 |

Table des matières

[Partie 1 3](#_Toc185450609)

[Code C 3](#_Toc185450610)

[GetString() 3](#_Toc185450611)

[Check\_all\_char 4](#_Toc185450612)

[Spécalité du code principal 4](#_Toc185450613)

[Test 5](#_Toc185450614)

[Partie 2 6](#_Toc185450615)

[Code C 6](#_Toc185450616)

[Petites modifications 6](#_Toc185450617)

[Programme principal 6](#_Toc185450618)

[Test 7](#_Toc185450619)

[Code VHDL 8](#_Toc185450620)

[Graphe MSS 9](#_Toc185450621)

[Choix d’implémentation 9](#_Toc185450622)

[Conclusion 10](#_Toc185450623)

[Annexes 10](#_Toc185450624)

# Travail préliminaire

## Uart

**UART0 :**

Adresse de base : **0xFFC02000**

GIC IRQ num : **194** (level)

**Donnée labo :**

On utilise le module UART0 est configuré dans le mode "No flow control".

L’horloge l4\_sp\_clk est à une fréquence de 100MHz.

Configurer le module UART0 avec les caractéristiques :

Baudrate : 9600

- Bit de données : 8

- Bit de parité désactivé

- Bit de stop : 1

- Activer les buffers FIFO en émission et réception

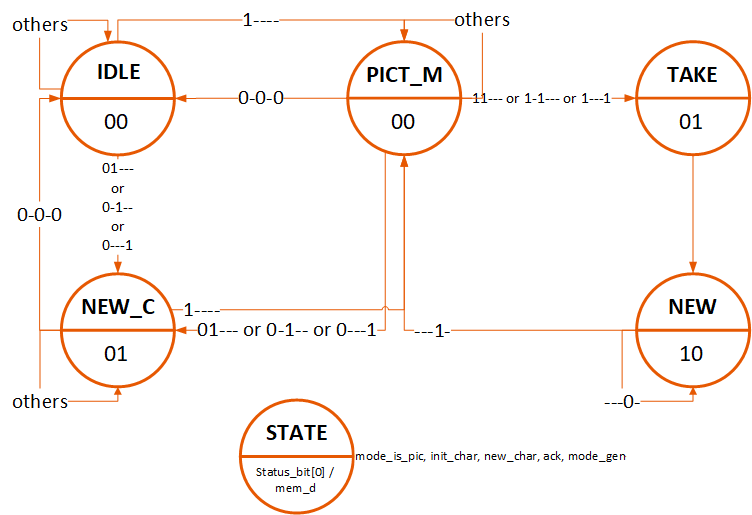
## Partie VHDL

Voici notre plan d’adressage :

|  |  |  |
| --- | --- | --- |
| Adresse | Rd | Wr |
| 0x00 | [31..0] Int Const ID | not used |
| 0x04 | [31..4] 0x0 & [3..0] buttons | not used |
| 0x08 | [31..10] 0x0 & [9..0] switchs | not used |
| 0x0C | [31..10] 0x0 & [9..0] leds | [31..10] reserved & [9..0] leds |
| 0x10 | [31..28] 0x0 & [27..0] hex3 à hex0 | [31..28] reserved & [27..0] hex3 à hex0 |
| 0x14 *– serial transmitter* | [31..2] 0x0 & [1..0] con\_80p\_status | [31..20] reserved & [19..16] code & [15..0] data |
| 0x18 *– internal counter* | [31..0] counter current value | [31..2] reserved & [1] enable counter & [0] reset counter |
| 0x1C *– IRQ managment* | [31..1] 0x0 & [0] IRQ\_flag | [31..1] 0x0 & [0] clear\_IRQ |

NB : si la valeur du compteur est 3, cela signifie 3\*20ns (donc 60ns et non 3ns)

### Graphe MSS



NB : le status\_bit[1] est toujours mit à un.

### Choix d’implémentation

Quand la mss active mem\_d elle met à jour les chars et le checksum. Quand ack, new\_char ou init\_char n’est pas écrit la valeur par défaut est 0. Ainsi si on écrit un 1 sur l’un de ces signaux alors il ne reste actif que pendant un seul coup d’horloge.

Pour avoir une nouvelle valeur il faut écrire un acknowledge ce qui signifie plusieurs choses. D’une part, pour le mode manuel, après avoir appuyé sur new\_c il faudra faire un ack pour mettre à jour la valeur. De manière similaire, dans le mode auto, il faudra faire un ack après chaque lecture si l’on veut pouvoir lire une nouvelle valeur à chaque fois. Finalament, après un init\_char il faudra aussi faire un ack pour avoir la valeur à jour (pas forcement utile en mode auto car on fait déjà un ack après chaque lecture).

Lors d’un reset, l’interface se trouve en mode non fiable dans la MSS mais pas en état de "lecture" (NEW\_C), cet état amènera au bon endroit en fonction du mode de fonctionnement demandé et des autres entrées de la MSS.

# Conclusion

La partie 1 de ce labo fut assez simple à implémenter autant du côté VHDL que du côté C, la partie la plus difficile fut vraiment la partie 2 ou nous avons rencontré beaucoup de problème, dont la raison était très souvent difficile (voir impossible) à trouver. Nous avons effectué de multiples essais et nous nous sommes heurtés à beaucoup d’erreurs mais, au bout du compte, nous avons trouvés des solutions à toutes ces complications bien que certaines de ces solutions sont difficilement explicables (mais tout de même fonctionnelles).

|  |  |
| --- | --- |
| **Date** | 18.12.2024 |
| **Signature de l’étudiant 1** | Guillaume Gonin |
| **Signature de l’étudiant 2** | Patrick Maillard |

# Annexes

**avl\_user\_interface.vhdl (v1.4) :**

**avl\_user\_interface.vhdl (v3.4):**