NOM : **Feuille réponse n°1** DS M4209C Oct. 2015

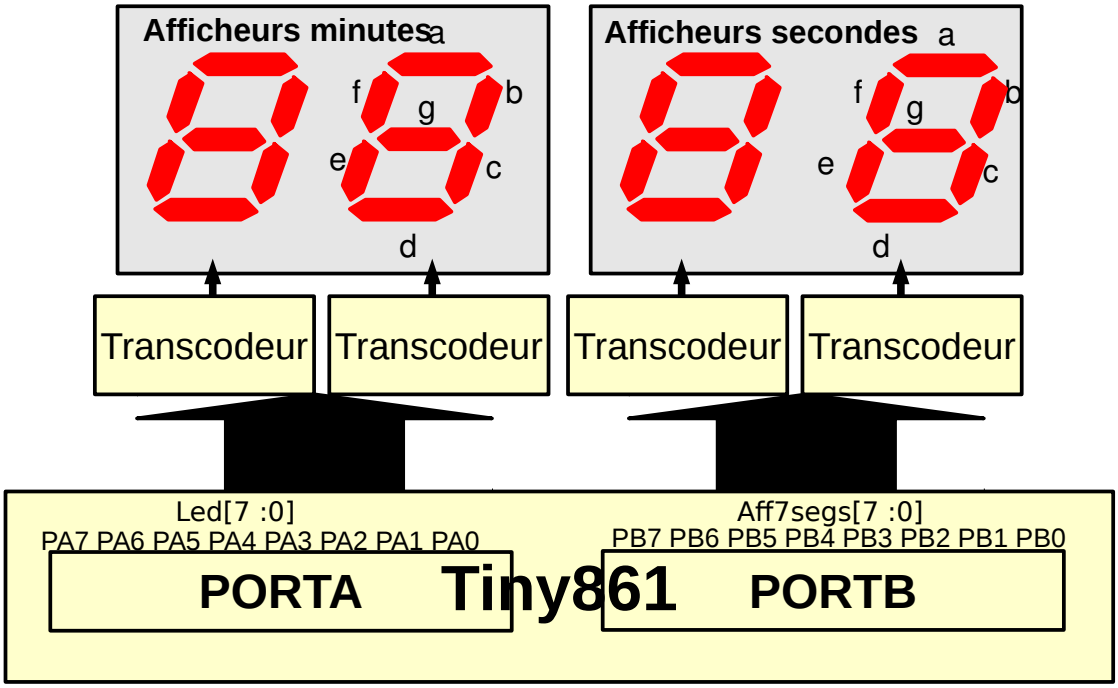
Prénom : **Correction ajoutée en rouge**
Groupe :

Exercice 1 (Etude partielle d'un Réveil avec processeur)

Dans cet exercice, on cherche à réaliser un réveil dans un FPGA en utilisant le processeur ATTiny861 des Tds et Tps.

I) Processeur avec transcodeurs

La partie matérielle est présentée partiellement sous forme schématique pour les minutes et les secondes. Les heures sont affichées avec le registre **DDRA** en suivant le même principe.



1°) Quelle est l'entité du transcodeur s'il s'agit du transcodeur habituel sept segments

Réponse :

```
1       LIBRARY IEEE ;
2       USE IEEE.STD_LOGIC_1164.ALL ;
3       ENTITY transcod is port (
           entrees : in std_logic_vector(3 downto 0) ;

           sorties : out std_logic_vector(6 downto 0)) ;

        END transcod ;
```

2°) Quelle instruction en C enverra 35 secondes sur l'affichage?

Réponse :

```
1       PORTB = 0x35 ;
```

3°) Quelle instruction en C enverra 28 minutes sur l'affichage?

Réponse :

```
1       PORTA = 0x28 ;
```

4°) Quelle instruction en C enverra 16 heures sur l'affichage?
Réponse :

```
DDRA = 0x16 ;
```

5°) Pour gérer correctement les trois affichages il faut utiliser une variable de 24 bits, ce qui n'existe pas en C. On utilisera donc une variable de 32 bits (uint32_t en C). On vous donne la partie de programme qui incrémente les heures minutes secondes :

```
uint32_t HmnS ;
//....
HmnS++;
  if ((HmnS & 0x0000000F) > 0x00000009)
      HmnS += 0x00000006;
  if ((HmnS & 0x000000F0) > 0x00000050)
      HmnS += 0x000000A0;
  if ((HmnS & 0x00000F00) > 0x00000900)
      HmnS += 0x00000600;
  if ((HmnS & 0x0000F000) > 0x00005000)
      HmnS += 0x0000A000;
  if ((HmnS & 0x000F0000) > 0x00090000)
      HmnS += 0x00060000;
  if ((HmnS & 0x00FF0000) > 0x00230000)
      HmnS = 0x00000000;
```

Que donne ce morceau de programme pour les deux valeurs HmnS=0x00235943 et HmnS=0x00235959. Quelle heure sera alors affichée ?
Réponses : HmnS=0x00235944 (aucun if exécuté)

HmnS=0x0023595A (1er if) donne 0x00235960 (2eme if) donne 0x00235A00 (3eme if) donne 0x00236000 (4eme if) donne 0x00240000 qui est le résultat final.

6°) Écrire les instructions C capable de prendre la variable « HmnS » et de l'afficher.

Réponse :

PORTB = HmnS ; ou PORTB = HmnS & 0x000000FF;

PORTA = (HmnS & 0x0000FF00)>>8;

DDRA = (HmnS & 0x00FF0000)>>16;

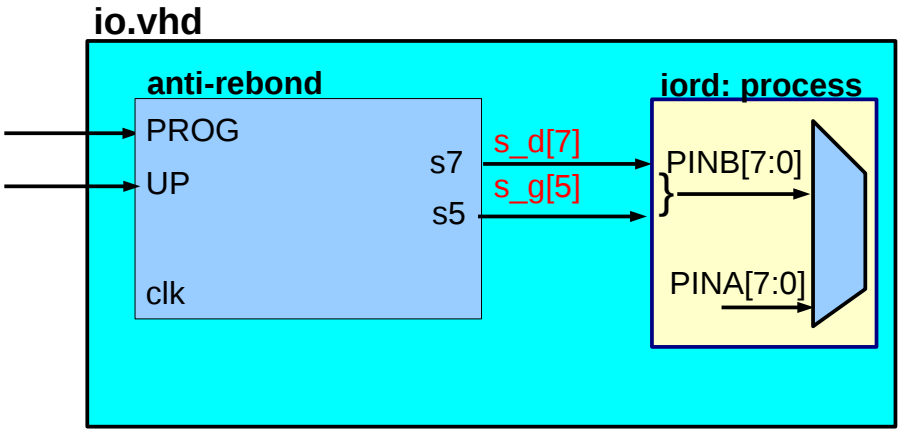
II) Étude des entrées

Un réveil nécessite des entrées : PROG pour entrer dans le mode de programmation de l'heure réveil, UP pour incrémenter cette même heure, DOWN pour décrémenter cette heure, et REV pour enclencher le mode déclenchement de sonnerie.

NOM : **Feuille réponse n°3** DS M4209C Oct. 2015

- PROG arrive en b_7 de PINB
- UP arrive en b_5 de PINB

Tout ceci passe à travers un circuit anti-rebond que l'on ne détaillera pas ici.



1°) Comment écrire en C un test vrai seulement si PROG est à 1 ?

Réponse :

if ((PINB & (1<<7))==(1<<7)) // decalage de 7 car b7 ou 0x80 en hexa

2°) Comment écrire en C une boucle qui attend tant que UP soit à 0 ?

Réponse :

while ((PINB & (1<<5))==0) ; // decalage de 5 car b5 ou 0x20 en hexa

III) Amélioration

La gestion de l'heure courante et le réglage de l'heure de réveil est un problème assez complexe. On décide donc de déporter l'heure courante complètement dans le matériel. On utilise donc les compteurs que vous avez réalisez en TP. Mais le mode programmation et le mode réveil sont laissés à l'initiative du processeur. S'il veut connaître l'heure courante il lui faut la lire.

1°) Combien de registres 8 bits nécessite la lecture de l'heure courante ?

Réponse : l'heure est sur trois octets nécessite donc trois PORTs

2°) Choisissez les registres correspondants (du tableau p29 du poly) et donnez alors le programme C pour les lire et mettre le résultat de la lecture dans la variable « HmnS ».

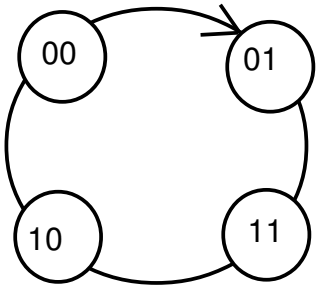
Réponses : PINA (mn), PINB (secondes) et DDRB (heures) feront l'affaire

```
HmnS = DDRB ;
HmnS = HmnS << 8 ;
HmnS |= PINA ;
HmnS = HmnS << 8 ;
HmnS |= PINB ;
```

Exercice 2 (VHDL classique)

Donner un programme VHDL qui implante le diagramme d'évolution ci-dessous en utilisant un ''case when'', c'est à dire sans les équations de récurrences :

NOM : **Feuille réponse n°4** DS M4209C Oct. 2015



On vous demande de gérer une initialisation asynchrone.

Réponse :

```
1         library ieee; -- les librairies ieee
2         use ieee.std_logic_1164.all;
3         ENTITY cmpt IS PORT(
4          clk : in std_logic ;
5          q : out std_logic_vector(1 downto 0)) ;
6         END cmpt ;
7
8         ARCHITECTURE acmpt of cmpt IS
9         SIGNAL s_cmpt : std_logic_vector(1 downto 0) ;
10        BEGIN
11         PROCESS(clk) BEGIN
12          IF clk'event and clk='1' then
13             CASE s_cmpt IS  --style case when
14                   WHEN "00" => s_cmpt <="01";
15                   WHEN "01" => s_cmpt <="11";
16                   WHEN "11" => s_cmpt <="10";
17                   WHEN OTHERS => s_cmpt <="00" ;
18                END CASE;
19          END IF ;
20         END PROCESS ;
21         q <= s_cmpt ;
22        END agray ;
```

Exercice 3

On désire réaliser un circuit capable de remplir une mémoire à partir d'un fichier HEX envoyé sur une liaison série. Nous allons étudier partiellement ce circuit. HEX est un format texte, on veut donc le transformer en hexadécimal. Par exemple, "FE" sera transformé en 0xFE où "FE" représente une donnée sur 16 bits (2 codes ASCII consécutifs 0x46 pour 'F' et 0x45 pour 'E'). Le circuit qui fait cette transformation est un circuit combinatoire appelé **comb**.

I) Étude du transcodage (comb)

NOM : **Feuille réponse n°5** DS M4209C Oct. 2015

Le circuit de transcodage est un circuit combinatoire dont on donne le code VHDL ci-dessous :

```
1         library ieee;
2         use ieee.std_logic_1164.all;
3         entity comb is port (
4           data : in std_logic_vector(7 downto 0);
5           value_bit,colon_bit,LF_bit,CR_bit : out std_logic;
6           value4 : out std_logic_vector(3 downto 0)
7           );
8         end comb;
9
10        architecture acomb of comb is
11        signal valueMoins30,valueMoins37:std_logic_vector(7 downto 0);
12        signal sortie : std_logic_vector(7 downto 0);
13        begin
14          valueMoins30 <= data - x"30";
15          valueMoins37 <= data - x"37";
16          sortie <= "1000" & valueMoins30(3 downto 0) when
17                                (data <= x"39" and data >=x"30") else
18                "1000" & valueMoins37(3 downto 0) when
19                             (data <= x"46" and data >=x"41") else
20                "0100" & x"F" when data = x"3A" else --colon = ":"
21                "0010" & x"F" when data = x"0D" else --LF Line Feed
22                "0001" & x"F" when data = x"0A" else --CR Carriage
          Return
23                "0000" & x"F"; -- Error
24          value4 <= sortie(3 downto 0);
25          value_bit <= sortie(7);
26          colon_bit <= sortie(6);
27          LF_bit <= sortie(5);
28          CR_bit <= sortie(4);
29        end acomb;
```

1°) Calculer les valeurs des signaux valueMoins30 et valueMoins37 si on a le caractère 'B' en entrée (de code ASCII 0x42)

Réponses : valueMoins30 = 0x42 - 0x30 = 0x12

valueMoins37 = 0x42 – 0x37 = 0x0B = 11

2°) Calculer la valeur du signal "sortie" de ce composant si on a le caractère 'B' en entrée

Réponse : `"1000" & 0xB soit "10001011" (lignes 18-19)`

3°) Calculer toutes les sorties de ce composant si on a le caractère 'B' en entrée

Réponses : value4 = $(1011)_2$ value_bit = 1, colon_bit = 0, LF_bit=0, CR_bit=0.

NOM : **Feuille réponse n°6** DS M4209C Oct. 2015

II) Le composant combinatoire précédent est connecté à un composant FIFO qui reçoit les données provenant de la RS232. On donne l'entité complète de ce FIFO :

```
1       entity bbfifo_16x9 is
2           Port (       data_in : in std_logic_vector(8 downto 0);
3                       data_out : out std_logic_vector(8 downto 0);
4                          reset : in std_logic;
5                          write : in std_logic;
6                           read : in std_logic;
7                           full : out std_logic;
8                       half_full : out std_logic;
9                    data_present : out std_logic;
10                            clk : in std_logic);
11      end bbfifo_16x9;
```

1°) Parmi ces entrées et sorties laquelle connecteriez vous au composant combinatoire **comb** de I) ? Quel problème cela pose-t-il ?

Réponses : data_out mais il est trop grand d'un bit (de parité) que l'on peut négliger.

2°) Le code source du FIFO nous montre l'extrait suivant :

```
1               count_lut: LUT4
2                generic map (INIT => X"6606")
3                port map( I0 => pointer(0),
4                          I1 => read,
5                          I2 => pointer_zero,
6                          I3 => write,
7                           O => half_count(0));
```

On vous demande de faire la table de vérité correspondante et d'écrire son équation correspondante.

Réponses (au verso) :

I3	I2	I1	I0	0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1

NOM : **Feuille réponse n°7** DS M4209C Oct. 2015

1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0