Exemple de code VHDL pour interfacer le LCD sur DE2

Dans cet exemple, nous allons voir comment écrire du VHDL pour utiliser le FPGA pour envoyer des données sur l'écran LCD de la plaquette. Certaines broches du FPGA sont connectes a une puce qui sert à contrôler le LCD. Il faut donc savoir comment communiquer avec cette puce pour que l'affichage se fasse correctement sur le LCD.

La puce en question est le HD44780. Une bonne habitude à prendre c'est d'aller voir la fiche technique. La fiche technique nous donne beaucoup d'information sur le fonctionnement. De plus, il faut aussi aller voir la fiche technique de la plaquette DE2 pour savoir quelles broches y sont connectées.

Une recherche sur la fiche technique du DE2 nous donne les fils qui sont connectés.

Signal Name	FPGA Pin No.	Description
LCD_DATA[0]	PIN_J1	LCD Data[0]
LCD_DATA[1]	PIN_J2	LCD Data[1]
LCD_DATA[2]	PIN_H1	LCD Data[2]
LCD_DATA[3]	PIN_H2	LCD Data[3]
LCD_DATA[4]	PIN_J4	LCD Data[4]
LCD_DATA[5]	PIN_J3	LCD Data[5]
LCD_DATA[6]	PIN_H4	LCD Data[6]
LCD_DATA[7]	PIN_H3	LCD Data[7]
LCD_RW	PIN_K4	LCD Read/Write Select, 0 = Write, 1 = Read
LCD_EN	PIN_K3	LCD Enable
LCD_RS	PIN_K1	LCD Command/Data Select, 0 = Command, 1 = Data
LCD_ON	PIN_L4	LCD Power ON/OFF
LCD_BLON	PIN_K2	LCD Back Light ON/OFF

Ca nous dit qu'il y a 8 bits pour les données et 5 fils pour le contrôle. L'une nous permet de choisir si on veut lire ou écrire, une autre active le LCD, une autre indique si c'est une donnée ou une commande et les 2 derniers nous disent si on veut allumer le LCD et sa lumière ou pas.

Ce qu'il faut donc comprendre c'est que je peux lire ou écrire à cette puce. En plus de pouvoir envoyer des données à afficher, je peux aussi lui donner des commandes (« tout effacer », par exemple).

Il reste quelques informations a aller chercher. Premièrement, il n'est probablement pas possible d'envoyer les données à une vitesse infinie. Il y a une vitesse maximale après laquelle la puce ne pourra plus réagir. De plus, il faut connaitre les commandes différentes et la séquence d'opération pour pouvoir écrire des lettres sur le LCD. Toute cette information se trouver a l'intérieur de la fiche technique de la puce.

En lisant le document, il est possible de découvrir plusieurs choses. Même si je résume l'information ici, je vous encourage a le faire vous-mêmes :

- Les commandes sont envoyées dans la puce au front descendant du signal ENABLE.
- Les commandes demandent des microsecondes et parfois plus que 1 milliseconde.
- Finalement, il y a un tableau qui nous résume les commandes en nous disant quelles valeurs mettre sur les fils différents.

Nous pouvons tirer des conclusions pour chaque point énuméré précédemment. Le premier point nous dit que, si nous voulions envoyer plusieurs commandes de suite, nous ne pouvons pas garder le ENABLE à '1'. Il faudra le monter à '1' et le baisser a '0' avant d'envoyer. Le deuxième point nous dit que notre horloge de 50MHz ne pourra pas être utilise directement. Il faudra soit la réduire ou soit utiliser une autre horloge. Le troisième point indique que nous n'avons pas besoin de deviner quelles valeurs envoyer dans les fils différents et qu'il existe un tableau qui résume le tout pour nous.

Pour concevoir le système, il faut commencer par créer une horloge qui est assez lente. Pour le but de cet exercice, nous n'avons pas besoin d'être précis. Nous savons que l'horloge de 50MHz a une période de 20ns. Si nous utilisions un compteur de 20 bits, nous aurions une fréquence qui est 10^6 fois plus lent (20ms). Le LCD aurait donc amplement de temps pour faire n'importe quelle opération puisque l'opération la plus lente ne prend que 1.52ms.

```
PROCESS (clk)
BEGIN
    IF clk'EVENT AND clk = '1' THEN
        s_slowclk_counter <= s_slowclk_counter + 1;
        IF s_slowclk_counter > 524287 THEN
            s_slowclk <= '0';
        ELSE
            s_slowclk <= '1';
        END IF;
    END IF;
END PROCESS;
```

Dans cet exemple, s_slowclk_counter est de 20bit. L'horloge s_slowclk est une horloge qui est à peu près de 20ms. Nous l'utilisons comme horloge principale pour les autres opérations.

En regardant dans la fiche technique, il est possible de voir que nous devons faire quelques opérations avant que le LCD puisse être utilise :

1. Indiquer le format et la taille utilisée. Il faut dire qu'on veut opérer en format 8 bits (il est possible d'utiliser 4 bits aussi) et que nous voulons utiliser soit une ligne ou soit les deux lignes de l'afficheur.
2. Allumer l'afficheur
3. Indiquer le mode de fonctionnement. Il faut dire qu'on veut que l'afficheur qu'on veut qu'il se déplace vers la droite a chaque fois qu'il reçoit un nouveau caractère.
4. Ecrire le caractère qu'on veut envoyer.

Pour gérer ce genre de séquences, il est pratique de se créer une machine a états. Dans notre machine, chaque numéro énuméré précédemment correspond a 2 états : il y a un état pour faire l'action voulu et l'autre état pour ramener le ENABLE a zéro pour que la puce lise ces valeurs.

Le système que nous voulons concevoir aura seulement 2 fonctions : effacer l'afficheur pour recommencer du début et écriture de la lettre 'A'. Quand nous pesons sur un bouton, l'afficheur sera efface et le curseur reviendra en haut a gauche. Quand nous pesons sur le deuxième bouton, on écrira la lettre 'A' sur l'afficheur.

Notre machine à états aura les états suivants :

etat_rien : Etat du début quand on attend qu'un bouton soit pesé

```
        WHEN etat_rien =>
            IF s_button_pressed = '1' THEN
                s_etat_prochain <= etat_conf;
            ELSIF swstartagain = '1' THEN
                s_etat_prochain <= etat_efface;
            ELSE
                s_etat_prochain <= etat_rien;
            END IF;
            LCD_EN <= '0';
            LCD_RS <='0';
            LCD_RW <= '0';
            LCD_DATA0 <= '0';
            LCD_DATA1 <= '0';
            LCD_DATA2 <= '0';
```

```
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
```

etat_efface : Quand on pèse sur le bouton pour effacer, on passe a cet état. Toutes les valeurs pour effacer son inscrites ici et le ENABLE est a ‘1’. Il passera automatiquement à etat_efface2. On doit se rappeler que les valeurs sont données dans la fiche technique.

```
WHEN etat_efface =>
    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '1';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_efface2;
```

etat_efface2 : A etat_efface2, on ramène ENABLE a ‘0’ sans se préoccuper des autres valeurs. On passera automatiquement à etat_retour.

```
WHEN etat_efface2 =>
    LCD_EN <= '0';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_retour;
```

etat_retour : Dans etat_retour, on donne la commande pour retourner en haut a gauche de l’afficheur. Ici, on met toutes les valeurs requises a nos sorties et notre ENABLE a ‘1’. On passe automatiquement à etat_retour2.

```
WHEN etat_retour =>
    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '1';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
```

```
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_retour2;
```

etat_retour2 : Ici, ENABLE est remis à ‘0’ et on retourne automatiquement à etat_rien.

```
WHEN etat_retour2 =>
    LCD_EN <= '0';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_rien;
```

A partir de maintenant, nous n’allons plus parler des 2e états puisque nous savons que ca existe et que leur seul but est de mettre le ENABLE a ‘0’. LCD_EN, LCD_RS, LCD_RW sont a ‘0’ et les données peuvent avoir n’importe quelle valeur. Il faut seulement se rappeler d’indiquer le bon prochain état.

etat_conf : Dans etat_rien, quand nous pesons sur le bouton pour envoyer, la machine a états entre dans cet état. Dans cet état, nous indiquons que nous voulons être en 8 bits et que nous voulons utiliser les 2 lignes de l’afficheur. Nous passons automatiquement à etat_allume.

```
WHEN etat_conf =>
    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '1';
    LCD_DATA4 <= '1';
    LCD_DATA5 <= '1';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_conf2;
```

etat_allume : Dans etat_allume, nous allumons le LCD. On passe automatiquement à etat_mode.

```
WHEN etat_allume =>
    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '1';
```

```
        LCD_DATA2 <= '1';
        LCD_DATA3 <= '1';
        LCD_DATA4 <= '0';
        LCD_DATA5 <= '0';
        LCD_DATA6 <= '0';
        LCD_DATA7 <= '0';
        s_etat_prochain <= etat_allume2;
```

etat_mode : Dans etat_mode, nous indiquons que nous voulons déplacer le curseur vers la droite après chaque écriture. Nous passons immédiatement à etat_ecrit.

```
    WHEN etat_mode =>
        LCD_EN <= '1';
        LCD_RS <='0';
        LCD_RW <= '0';
        LCD_DATA0 <= '0';
        LCD_DATA1 <= '1';
        LCD_DATA2 <= '1';
        LCD_DATA3 <= '0';
        LCD_DATA4 <= '0';
        LCD_DATA5 <= '0';
        LCD_DATA6 <= '0';
        LCD_DATA7 <= '0';
        s_etat_prochain <= etat_mode2;
```

etat_ecrit : Dans état écrit, nous envoyons le caractère voulu (dans ce cas-ci, ‘A’). Apres cet état, nous passons automatiquement à etat_rien. Il est a noter que etat_ecrit n’a pas de etat_ecrit2 puisque etat_rien ramène notre ENABLE a ‘0’.

```
    WHEN etat_ecrit =>
        LCD_EN <= '1';
        LCD_RS <='1';
        LCD_RW <= '0';
        LCD_DATA0 <= '1';
        LCD_DATA1 <= '0';
        LCD_DATA2 <= '0';
        LCD_DATA3 <= '0';
        LCD_DATA4 <= '0';
        LCD_DATA5 <= '0';
        LCD_DATA6 <= '1';
        LCD_DATA7 <= '0';
        s_etat_prochain <= etat_rien;
```

L’énumération des états se fait de la façon suivante.

```
TYPE etats IS (etat_rien, etat_efface, etat_efface2, etat_retour,
etat_retour2, etat_conf, etat_conf2, etat_allume, etat_allume2, etat_mode,
etat_mode2, etat_ecrit);
```

Avec un code VHDL comme celui la, le système devrait déjà fonctionner. En pesant la touche pour effacer, l’afficheur devrait s’effacer et le curseur devrait retourner en haut a gauche. En pesant la touche pour envoyer, la

première ligne de l'afficheur devrait se remplir automatiquement de 'A'. La raison est que, en pesant sur la touche, notre doigt génère un '1' qui dure plusieurs millisecondes. Il est donc probable que le système détecte le bouton, envoie 'A', retourne a l'état de départ et recommence parce que notre doigt est encore sur le bouton. Si notre doigt restait assez longtemps, la boucle pourrait se faire plusieurs fois, et ainsi, ca pourrait remplir la ligne de 'A'. Si nous étions satisfait avec ce comportement, le code pourrait être termine.

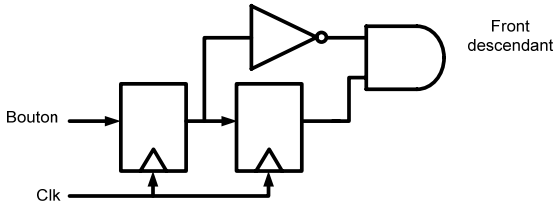
Par contre, si nous voulions que l'appui d'une touche ne génère qu'un seul 'A', il faudrait ajouter du code de plus. Une première idée qui vient a l'esprit c'est d'utiliser ce signal comme une entrée d'horloge dans une flip flop. Bien que ce « truc » puisse fonctionner pour des petits systèmes, ca cause beaucoup de complications quand les systèmes deviennent plus gros. On parle notamment de la distribution de l'horloge qui risque d'être moins bonne, une difficulté dans l'analyse de fréquence d'horloge maximale et un risque de métastabilité. Bref, ce n'est pas une bonne pratique d'utiliser ce signal dans l'entrée d'horloge.

La bonne façon de faire serait d'avoir un circuit qui génère un '1' qui dure exactement 1 cycle d'horloge en réponse a l'appui d'un bouton. Ceci pourrait se faire soit par une détection de front montant ou une détection de front descendant.

Voici le code pour un détecteur de front descendant :

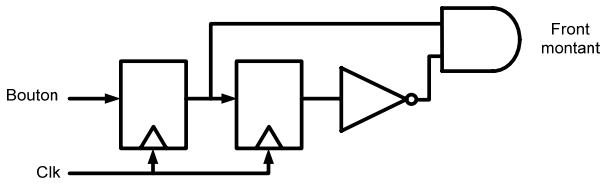
```
PROCESS (s_slowclk)
BEGIN
    IF s_slowclk'EVENT AND s_slowclk = '1' THEN
        s_button1 <= button;
        s_button2 <= s_button1;
    END IF;
END PROCESS;

s_button_pressed <= s_button2 AND NOT s_button1;
```

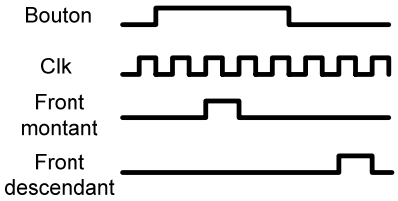


Si nous décidions d'utiliser le front montant l'expression de s_button_pressed deviendrait :

```
s_button_pressed <= s_button1 AND NOT s_button2;
```



Les diagrammes temporels pour le détecteur de front montant et celui de front descendant sont illustres a la figure suivante :



L’implémentation choisie repousse la détection d’un cycle d’horloge. Ceci n’est pas grave puisque nous voulons simplement savoir si le bouton a été pesé. De plus, a ces hautes vitesses, une personne ne pourra probablement pas détecter qu’il y a un délai d’un cycle d’horloge.

Le code final ressemblerait à ceci:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY lcd IS
    PORT (
          clk : IN STD_LOGIC;
          button : IN STD_LOGIC;
          swstartagain : IN STD_LOGIC;
          LCD_DATA0 : OUT STD_LOGIC;
          LCD_DATA1 : OUT STD_LOGIC;
          LCD_DATA2 : OUT STD_LOGIC;
          LCD_DATA3 : OUT STD_LOGIC;
          LCD_DATA4 : OUT STD_LOGIC;
          LCD_DATA5 : OUT STD_LOGIC;
          LCD_DATA6 : OUT STD_LOGIC;
          LCD_DATA7 : OUT STD_LOGIC;
          LCD_RW    : OUT STD_LOGIC;
          LCD_EN    : OUT STD_LOGIC;
          LCD_RS    : OUT STD_LOGIC;
          LCD_ON    : OUT STD_LOGIC;
          LCD_BLON  : OUT STD_LOGIC
         );
END lcd;

ARCHITECTURE rtl OF lcd IS

SIGNAL s_button1: STD_LOGIC;
SIGNAL s_button2: STD_LOGIC;
SIGNAL s_button_pressed: STD_LOGIC;
SIGNAL s_slowclk : STD_LOGIC;
SIGNAL s_slowclk_counter : STD_LOGIC_VECTOR(19 DOWNTO 0);
TYPE etats IS (etat_rien, etat_efface, etat_efface2, etat_retour,
etat_retour2, etat_conf, etat_conf2, etat_allume, etat_allume2, etat_mode,
etat_mode2, etat_ecrit);
SIGNAL s_etat_present  : etats;
SIGNAL s_etat_prochain : etats;

BEGIN

LCD_BLON <= '0';
LCD_ON <= '1';

PROCESS (clk)
BEGIN
    IF clk'EVENT AND clk = '1' THEN
        s_slowclk_counter <= s_slowclk_counter + 1;
        IF s_slowclk_counter > 524287 THEN
            s_slowclk <= '0';
```

```
        ELSE
            s_slowclk <= '1';
        END IF;
    END IF;
END PROCESS;

PROCESS (s_slowclk)
BEGIN
    IF s_slowclk'EVENT AND s_slowclk = '1' THEN
        s_button1 <= button;
        s_button2 <= s_button1;
    END IF;
END PROCESS;

s_button_pressed <= s_button2 AND NOT s_button1;

PROCESS (s_slowclk)
BEGIN
    IF s_slowclk'EVENT AND s_slowclk = '1' THEN
        s_etat_present <= s_etat_prochain;
    END IF;
END PROCESS;

PROCESS (s_etat_present, s_button_pressed, swstartagain)
BEGIN
    CASE s_etat_present IS
        WHEN etat_rien =>
            IF s_button_pressed = '1' THEN
                s_etat_prochain <= etat_conf;
            ELSIF swstartagain = '1' THEN
                s_etat_prochain <= etat_efface;
            ELSE
                s_etat_prochain <= etat_rien;
            END IF;
            LCD_EN <= '0';
            LCD_RS <='0';
            LCD_RW <= '0';
            LCD_DATA0 <= '0';
            LCD_DATA1 <= '0';
            LCD_DATA2 <= '0';
            LCD_DATA3 <= '0';
            LCD_DATA4 <= '0';
            LCD_DATA5 <= '0';
            LCD_DATA6 <= '0';
            LCD_DATA7 <= '0';

        WHEN etat_efface =>
            LCD_EN <= '1';
            LCD_RS <='0';
            LCD_RW <= '0';
            LCD_DATA0 <= '1';
            LCD_DATA1 <= '0';
            LCD_DATA2 <= '0';
            LCD_DATA3 <= '0';
            LCD_DATA4 <= '0';
            LCD_DATA5 <= '0';
            LCD_DATA6 <= '0';
```

```
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_efface2;

WHEN etat_efface2 =>
    LCD_EN <= '0';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '1';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_retour;

WHEN etat_retour =>
    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '1';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_retour2;

WHEN etat_retour2 =>
    LCD_EN <= '0';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '1';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';
    s_etat_prochain <= etat_rien;



WHEN etat_conf =>

    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '1';
```

```
    LCD_DATA4 <= '1';
    LCD_DATA5 <= '1';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';


    s_etat_prochain <= etat_conf2;

WHEN etat_conf2 =>

    LCD_EN <= '0';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '0';
    LCD_DATA2 <= '0';
    LCD_DATA3 <= '0';
    LCD_DATA4 <= '1';
    LCD_DATA5 <= '1';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';


    s_etat_prochain <= etat_allume;
WHEN etat_allume =>

    LCD_EN <= '1';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '1';
    LCD_DATA2 <= '1';
    LCD_DATA3 <= '1';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';

    s_etat_prochain <= etat_allume2;

WHEN etat_allume2 =>

    LCD_EN <= '0';
    LCD_RS <='0';
    LCD_RW <= '0';
    LCD_DATA0 <= '0';
    LCD_DATA1 <= '1';
    LCD_DATA2 <= '1';
    LCD_DATA3 <= '1';
    LCD_DATA4 <= '0';
    LCD_DATA5 <= '0';
    LCD_DATA6 <= '0';
    LCD_DATA7 <= '0';

    s_etat_prochain <= etat_mode;
WHEN etat_mode =>
```

```
            LCD_EN <= '1';
            LCD_RS <='0';
            LCD_RW <= '0';
            LCD_DATA0 <= '0';
            LCD_DATA1 <= '1';
            LCD_DATA2 <= '1';
            LCD_DATA3 <= '0';
            LCD_DATA4 <= '0';
            LCD_DATA5 <= '0';
            LCD_DATA6 <= '0';
            LCD_DATA7 <= '0';


            s_etat_prochain <= etat_mode2;

        WHEN etat_mode2 =>

            LCD_EN <= '0';
            LCD_RS <='0';
            LCD_RW <= '0';
            LCD_DATA0 <= '0';
            LCD_DATA1 <= '1';
            LCD_DATA2 <= '1';
            LCD_DATA3 <= '0';
            LCD_DATA4 <= '0';
            LCD_DATA5 <= '0';
            LCD_DATA6 <= '0';
            LCD_DATA7 <= '0';


            s_etat_prochain <= etat_ecrit;

        WHEN etat_ecrit =>

            LCD_EN <= '1';
            LCD_RS <='1';
            LCD_RW <= '0';
            LCD_DATA0 <= '1';
            LCD_DATA1 <= '0';
            LCD_DATA2 <= '0';
            LCD_DATA3 <= '0';
            LCD_DATA4 <= '0';
            LCD_DATA5 <= '0';
            LCD_DATA6 <= '1';
            LCD_DATA7 <= '0';


            s_etat_prochain <= etat_rien;
    END CASE;

END PROCESS;


END;
```