0 - Présentation de l'activité 4 - Console DMX :

Traitement numérique du signal. Faire évoluer le projet afin de mettre en œuvre l'affichage multiplexé.
Relever des signaux commandant le multiplexage.

Pré-requis : ⇒ Avoir réaliser l'intégralité de l'activité 3
Durée estimée : ⇒ 4 heures
Objectif : ⇒ Implanter et programmer un **processeu**r dans un **FPGA.**

Vocabulaire spécifique à l'activité :

Durant cette activité vous implanterez un processeur dans un **FPGA.**
Pour cela vous allez éditer un schéma **OpenBus** mettant en œuvre des blocs **IP**.
Le processeur ainsi décrit est un **SOC**.

Sommaire de l'activité 4 :

* Rappel : sous ALTIUM la feuille de schéma *.SchDoc est en haut du projet, c'est le « TOP LEVEL ».

Schéma OpenBus à dessiner:

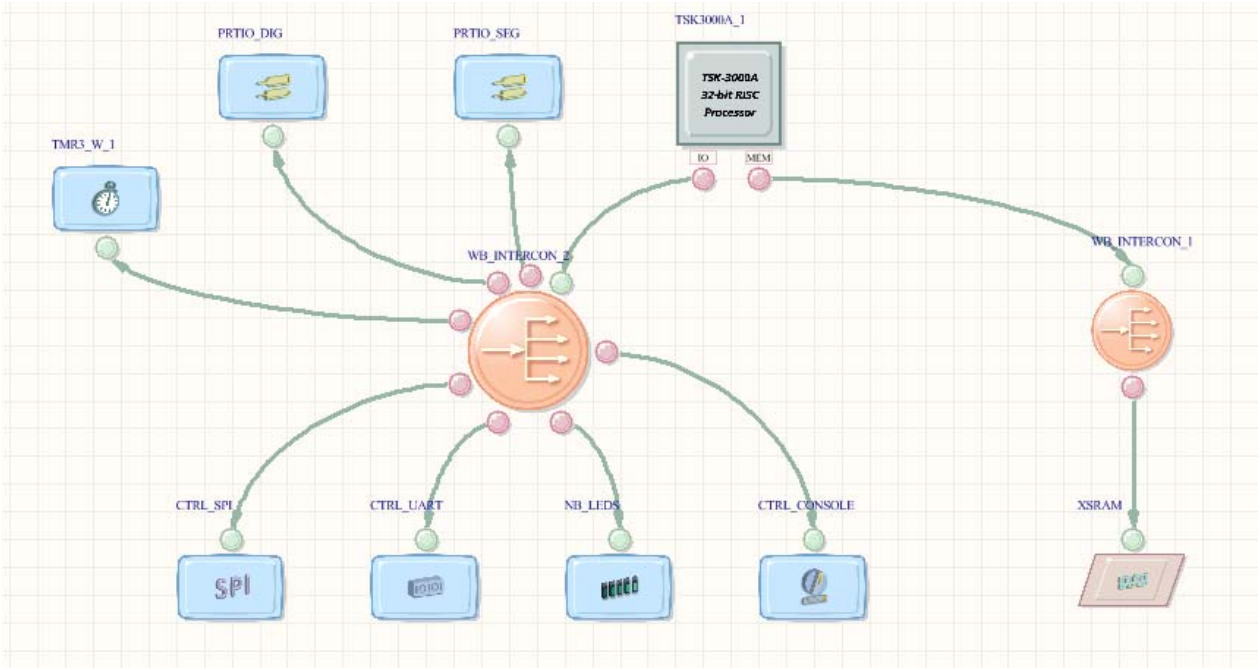
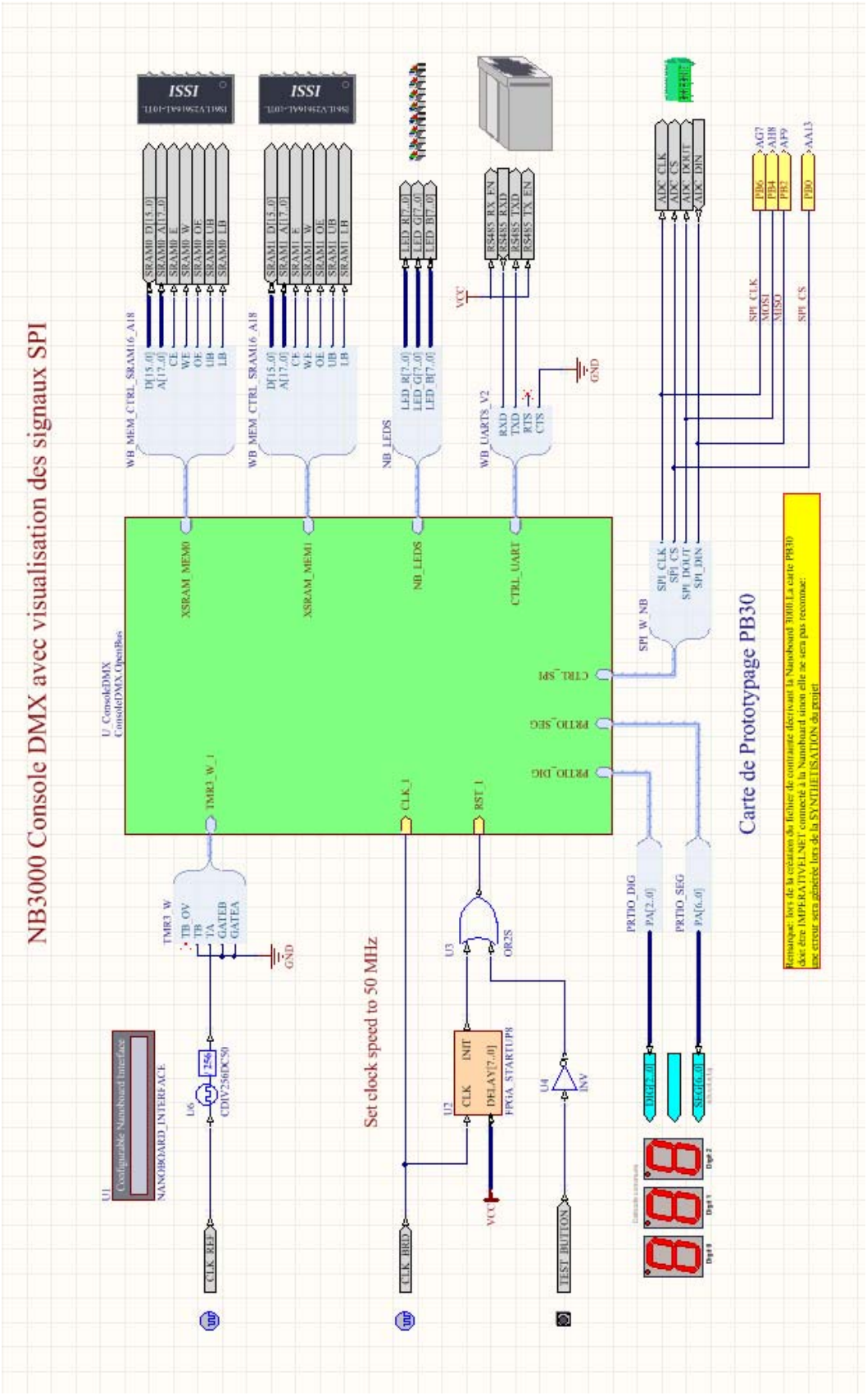


Schéma Top à modifier au cours du TP73 Console DMX:



1 Ouvrir un projet FPGA existant.

1.1 Repartir d'un projet existant :

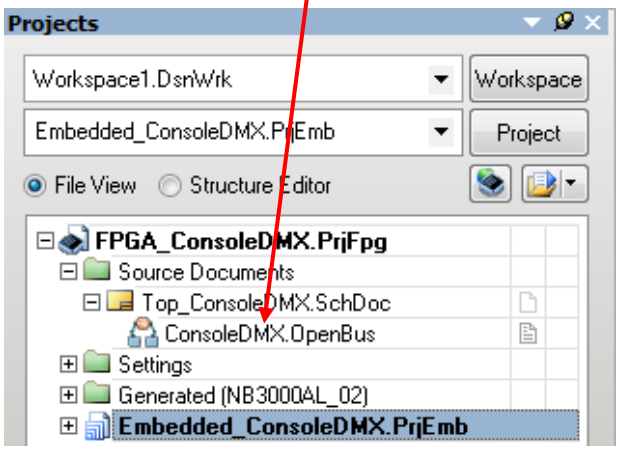
Ouvir un projet existant en utilisant la commande : **File >> Open >> Projet >> FPGA Projet**.

Dans l'onglet qui s'ouvre, aller dans le répertoire de travail « **TP72_ConsoleDMX** » puis choisir le projet nommé **« FPGA_ConsoleDMX.PrjFpg »**.

2 Editer et modifier le fichier OpenBus

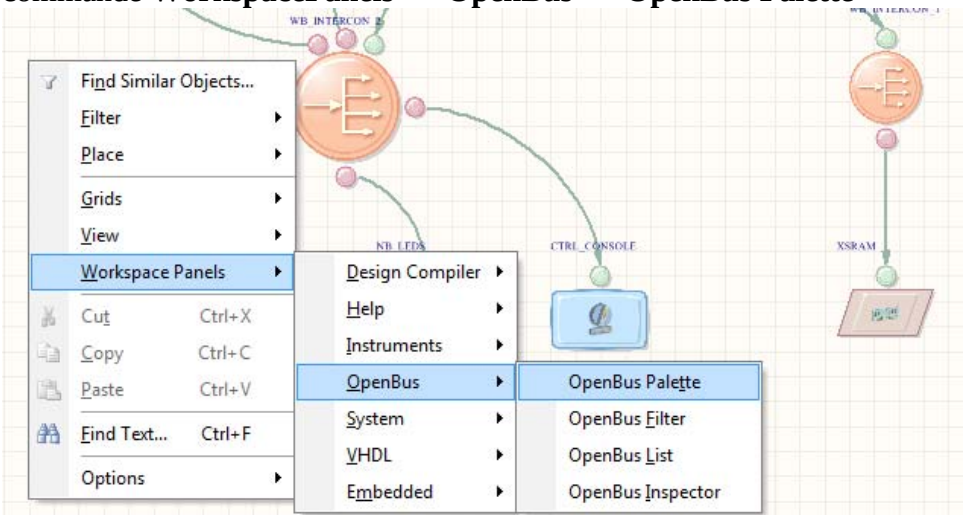
2.1 Ouvrir un fichier OpenBus existant

Double Clic bouton gauche sur le nom du fichier OpenBus *(ConsoleDMX.OpenBus)* dans l'onglet Projets.



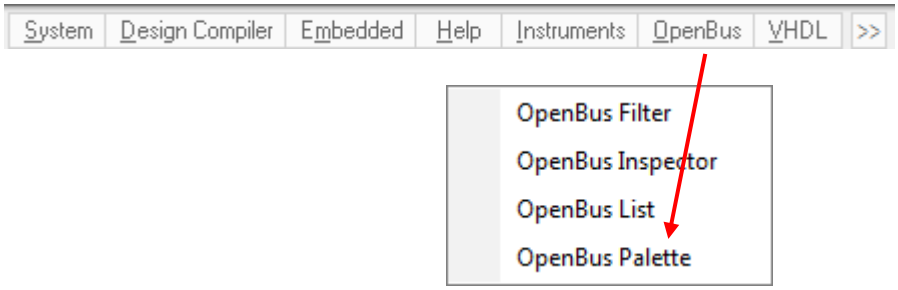
2.2 Placer les éléments OpenBus vsuivants afin de modifier la feuille de schéma OpenBus

1 ière méthode :
Clic bouton droit sur la feuille de travail OpenBus et choisir dans l'onglet qui apparaît la commande **WorkspacePanels >> OpenBus >> OpenBus Palette**



2 ième méthode :

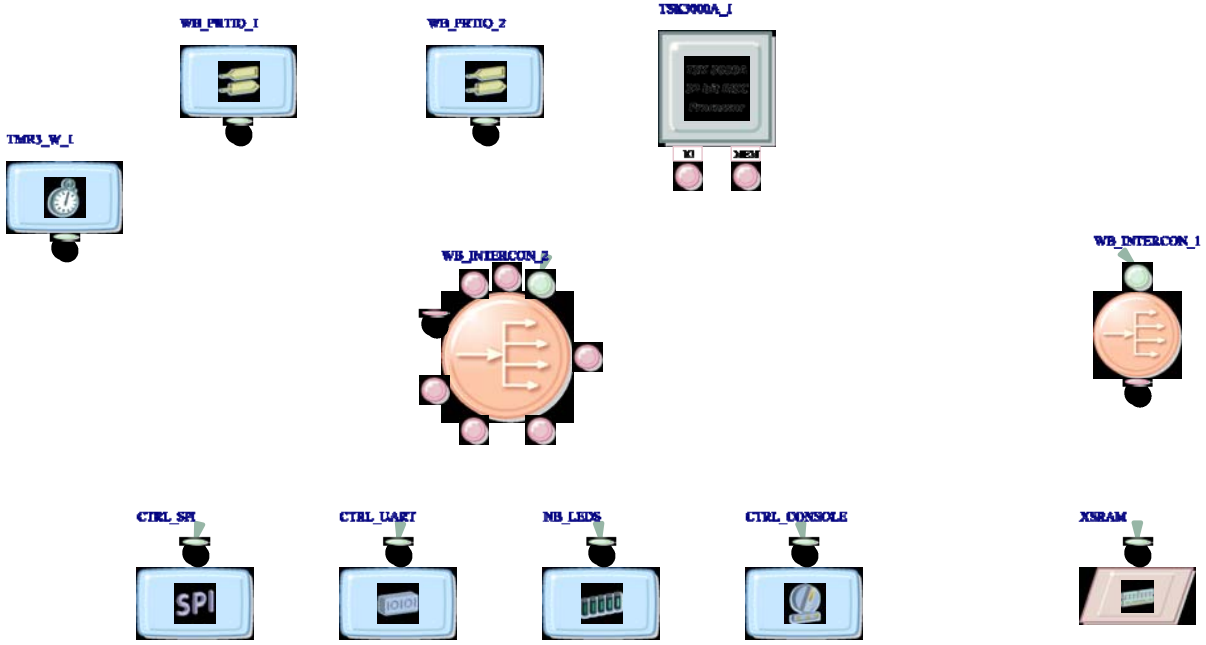
Clic bouton gauche en bas de la feuille de travail du projet, puis choisir l'icône Openbus Palette



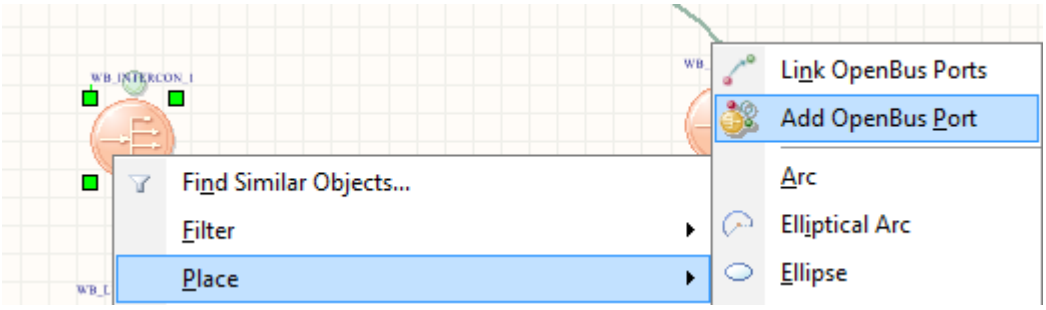
Le tableau ci-dessous identifie les composants Openbus à placer sur le bus périphérique

	Port IO	Permet de commander les digits unité, dizaine et centaine de l'afficheur à anode commune.
	Port IO	Permet de commander les segments (a, b, c, d, e, f et g) de l'afficheur à anode commune.
	Dual Timer Unit	Timer qui permet générer une interruption pour piloter l'affichage multiplexé.

=> Placer les composants sur la feuille de schéma OpenBus



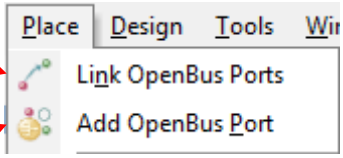
=> Ajouter un port OpenBus au composant WB_INTERCON_2 :



=> Dessiner les liens entre les éléments OpenBus :

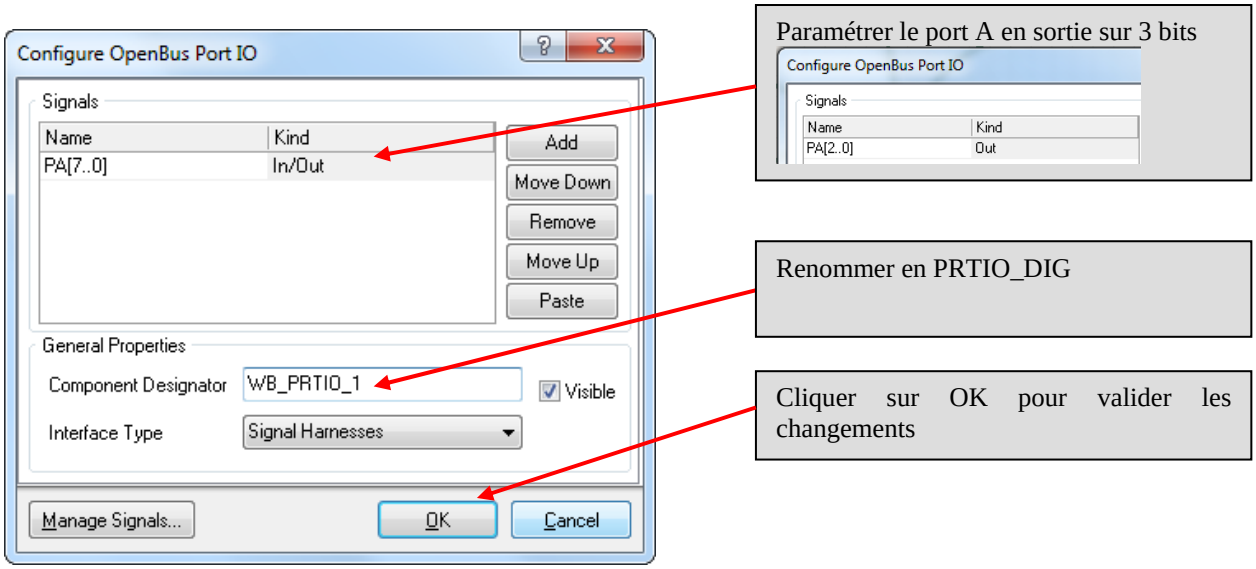
Pour dessiner les liens entre les structures OpenBus utiliser la fonction LINK

Pour rajouter les ports aux inter-connexions utiliser la fonction ADD Open Bus port



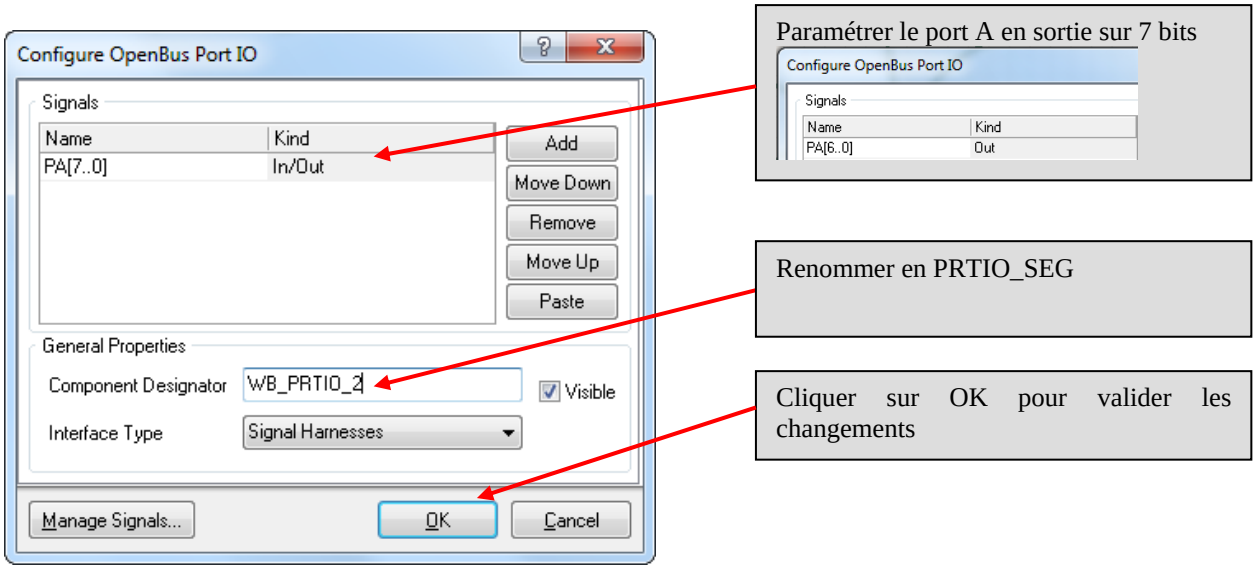
=> Paramétrer et renommer le composant WB_PRTIO_1

Clic bouton droit sur le composant WB_PRTIO_1, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus Port IO**



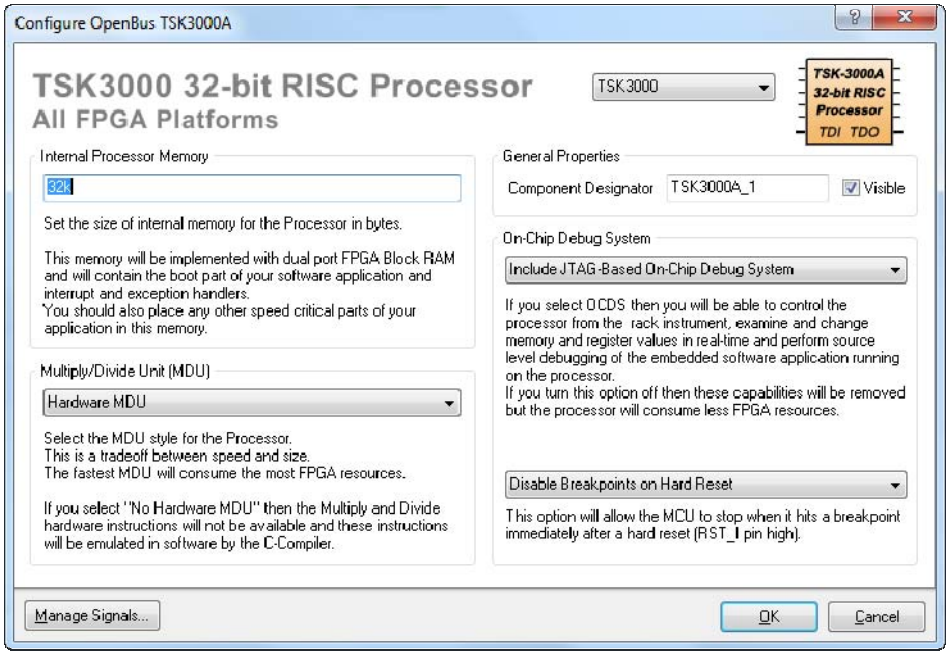
=> Paramétrer et renommer le composant WB_PRTIO_2

Clic bouton droit sur le composant WB_PRTIO_2, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus Port IO**



=> Paramétrer le processeur comme ci-dessous :

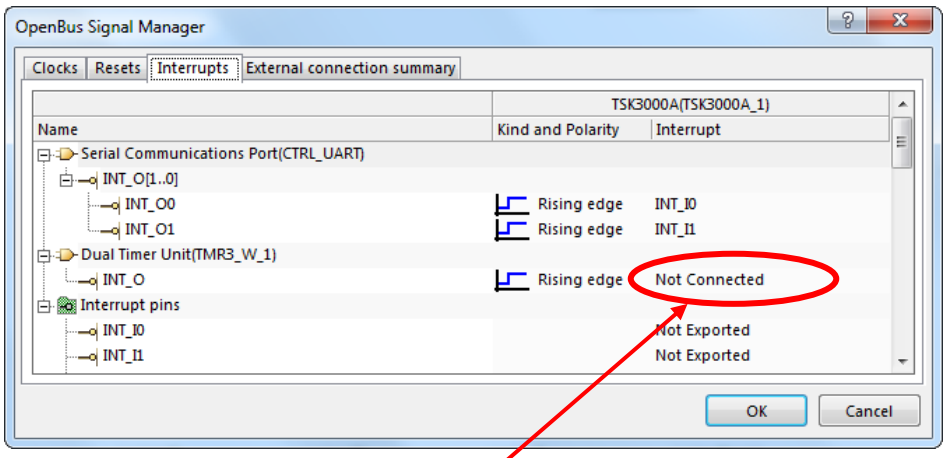
Clic bouton droit sur le composant port Processeur , puis dans l'onglet qui apparaît sélectionner **Configure TSK3000A_1**



Cliquer sur OK pour valider les changements

Clic bouton gauche sur le **Manage Signal**, puis dans l'onglet qui apparaît sélectionner **Interrupts**

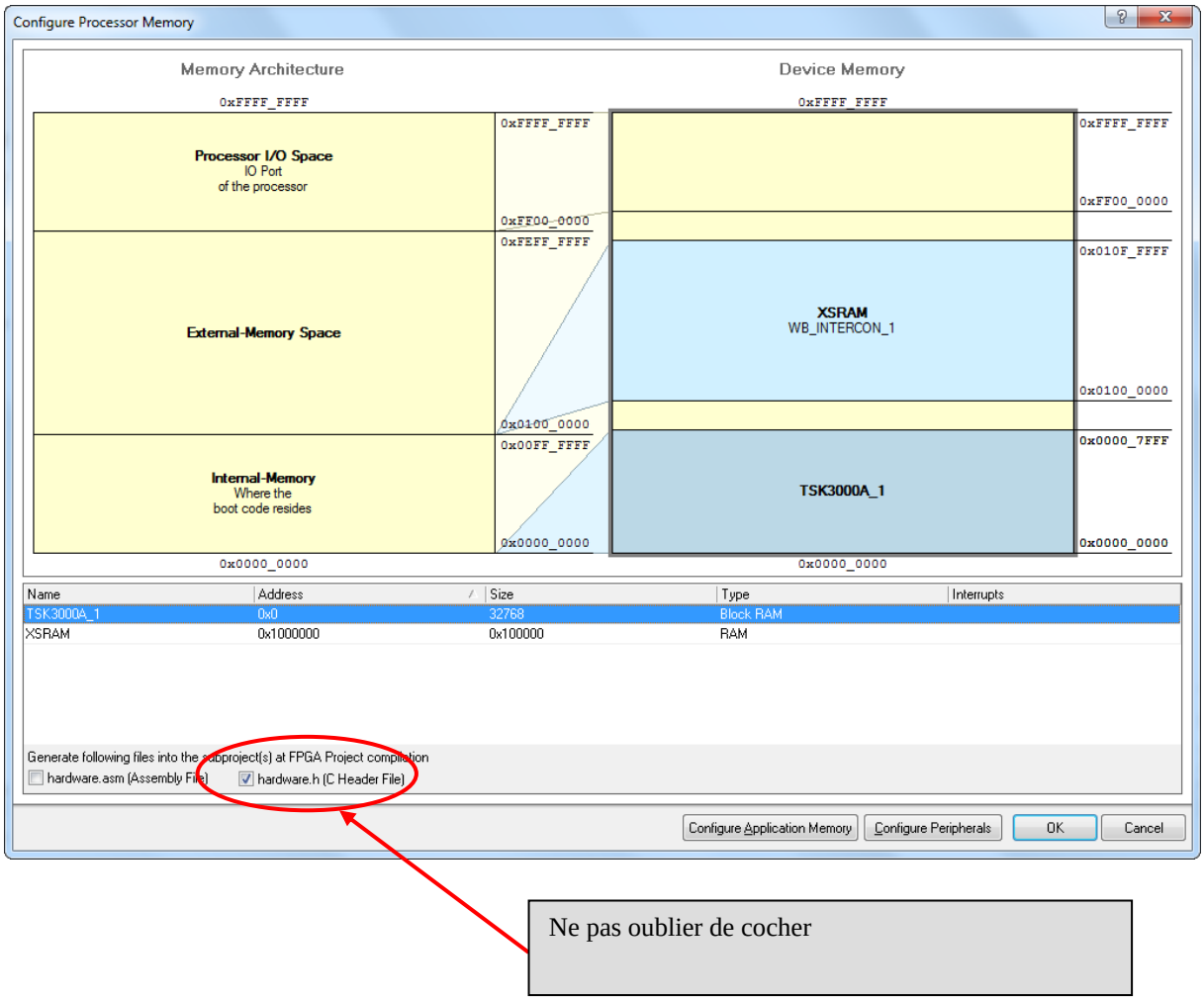
=> Visualiser le plan des interruptions du processeur comme ci-dessous :



Cliquer sur Not Connected pour choisir INT_I2

=> Visualiser le plan mémoire du processeur comme ci-dessous :

Clic bouton droit sur le port Processeur TSK3000, puis dans l'onglet qui apparaît sélectionner **Configure Processor Memory**

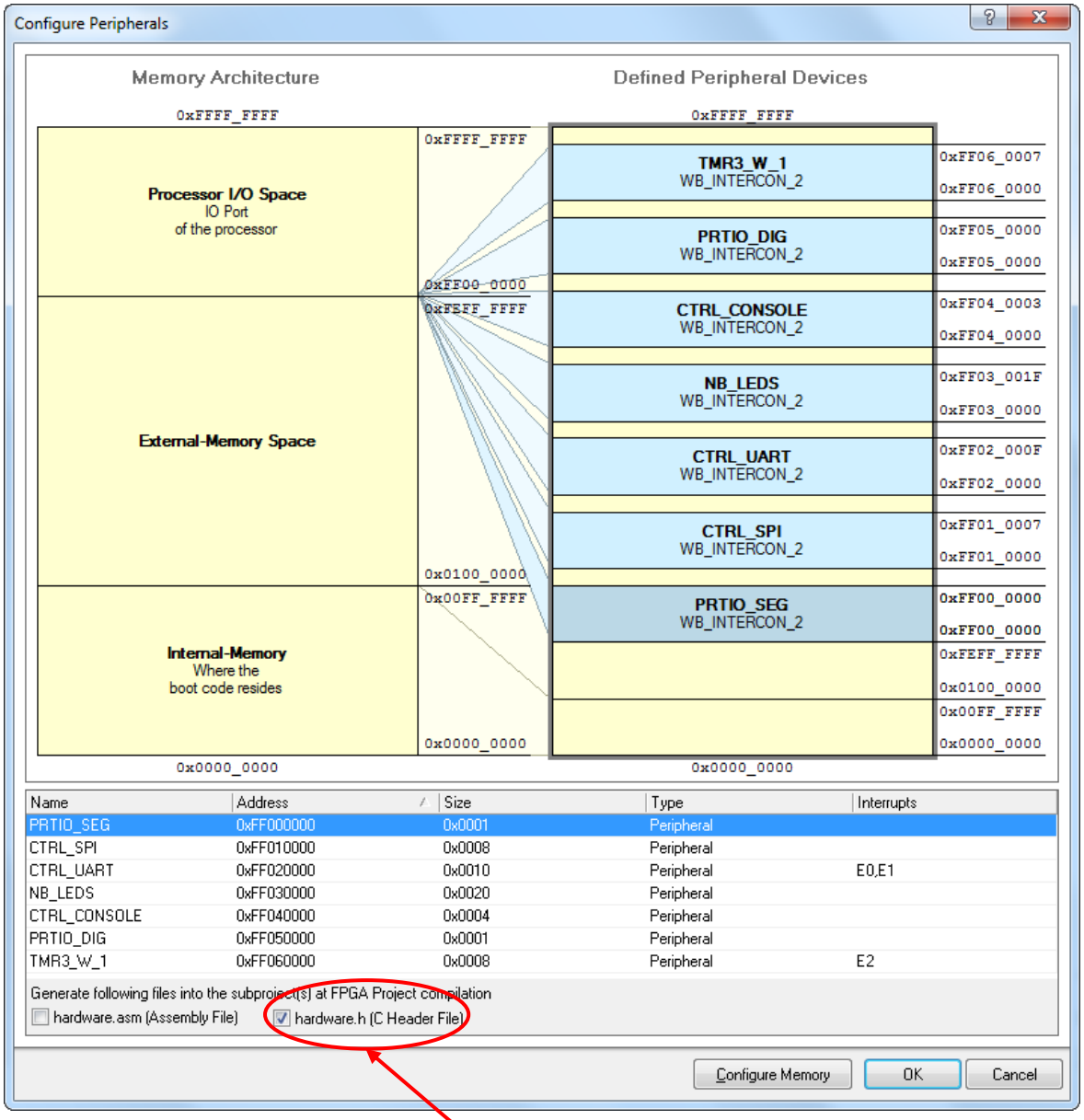


Ne pas oublier de cocher

Cliquer sur OK pour valider les changements

=> Visualiser le plan mémoire des périphériques comme ci-dessous :

Clic bouton droit sur le port Processeur TSK3000, puis dans l'onglet qui apparaît sélectionner **Configure Processor Peripherals**



Cliquer sur OK pour valider les changements

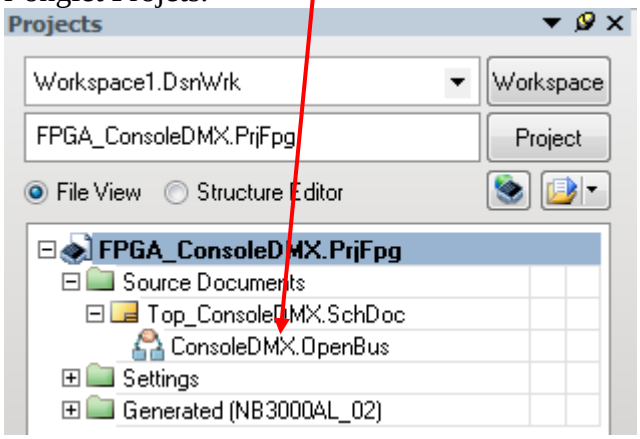
2.3 Sauvegarder le fichier OpenBus

Clic bouton droit sur le nom du fichier OpenBus *(ConsoleDMX.OpenBus*) dans l'onglet Projets et choisir la commande sauvegarder le document **Save As** avec le nom *ConsoleDMX.OpenBus* dans le répertoire de travail « **TP72_ConsoleDMX** ».

3 Editer et modifier le « TOP » schéma

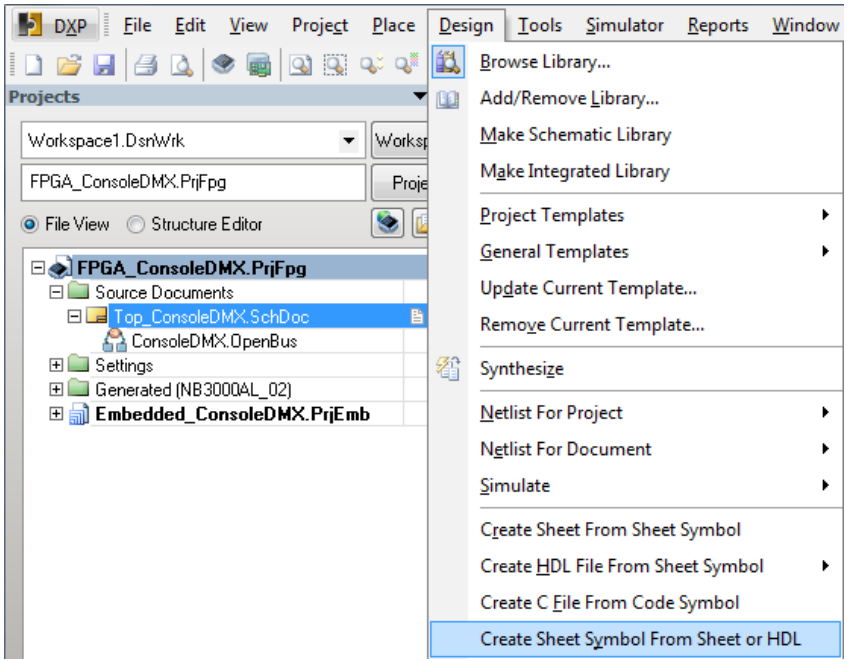
3.1 Ouverture du fichier schéma en tête du projet FPGA :

=> Ouvrir un schéma existant :
Double Clic bouton gauche sur le nom du fichier schéma *(Top_ConsoleDMX.SchDoc*) dans l'onglet Projets.

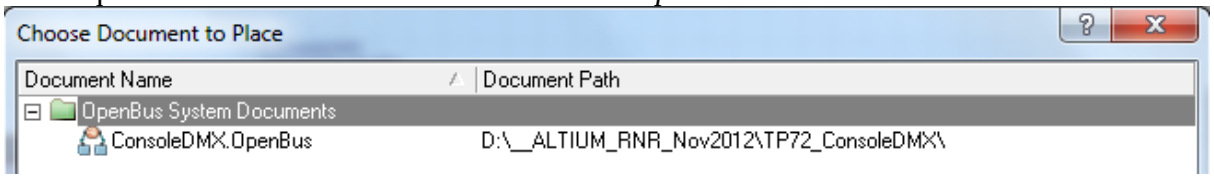


3.2 Placer dans le schéma le symbole créé à partir du fichier OpenBus :

Dans la barre de menu choisir la commande **Design >> Create Sheet Symbol From Sheet or HDL**



=> Cliquer sur le nom du fichier « *ConsoleDMX.OpenBus* »:

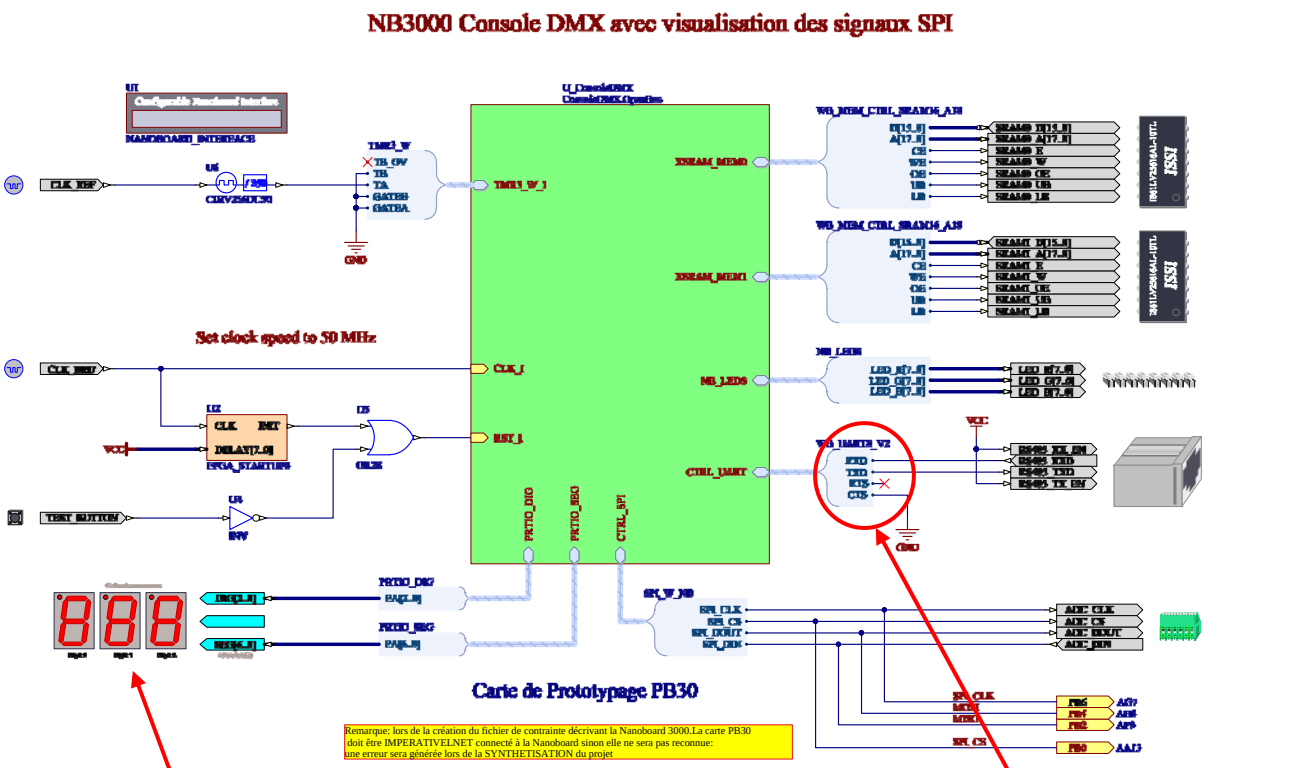


=> Organiser le corps du symbole comme ci-dessous en déplaçant les ports bleu et jaune :

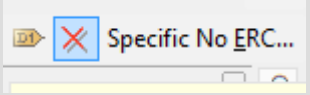
Vous obtenez le schéma bloc symbolisant le schéma OPEN BUS à implanter dans le FPGA :

Entrée du timer TMR3 :
Le timer TMR3 est câblé pour fonctionner en Mode Compteur, (division par 128).
Le débordement du timer TMR3 va permettre de déclencher une interruption.

Schéma TOP complet à obtenir :



Déclarer les signaux non utilisés :



Pour accéder au symbole de l'afficheur multiplexé « AFF_7SEG_BUS » qui se trouve dans la librairie « EXT_DE0_NANO.SchLib », il faut copier cette librairie dans le répertoire \Bibliothèques\Documents\Altium\AD10\Library\DE._NANO.

3.3 Placer et paramétrer les composants sur le schéma :

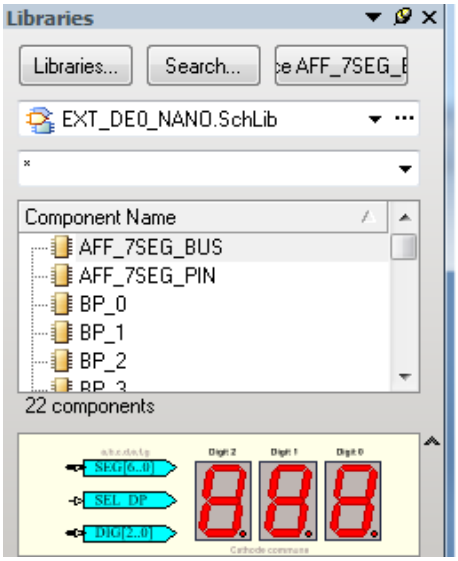
⇒ Placer dans le schéma les éléments suivants :

Description	Nom de la fonction	bibliothèque
Horloge de référence de 50 MHz	CLOCK_REFERENCE	FPGA NB3000 Port-Plugin.IntLib
Diviseur générique par 256	CDIV256DC50	FPGA Generic.IntLib
Afficheur Multiplexé 3 digits à 7 segments	AFF_7SEG_BUS	EXT_DE0_NANO.SchLib

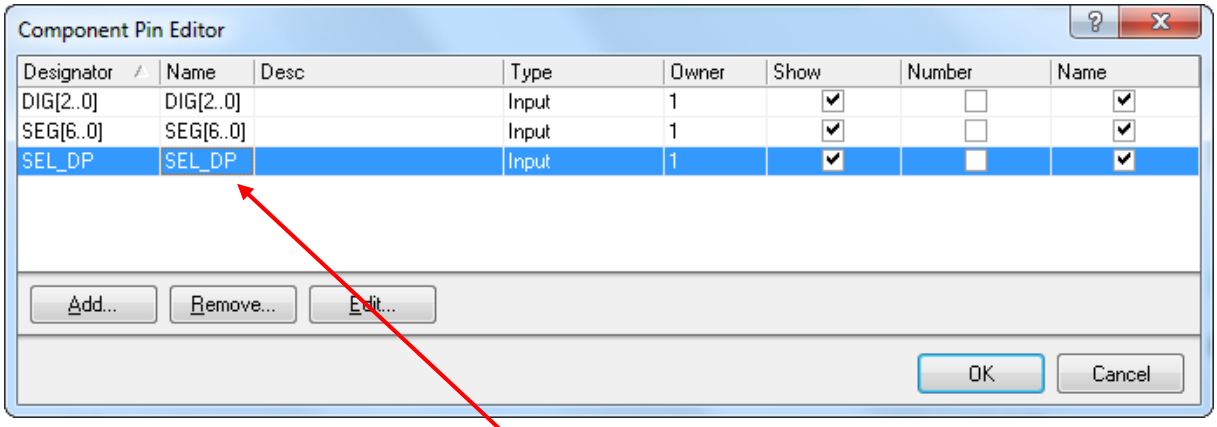
3.3.1 Placer l'afficheur multiplexé sur le schéma

Pour placer un nouveau composant :

⇒ cliquez sur **Librairies** sur le bord droit de l'écran
⇒ Sélectionnez la bibliothèque du composant
⇒ Sélectionnez le composant
⇒ Placer le composant



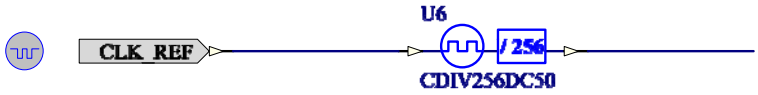
Double clic sur le composant AFF_7SEG_BUS puis clic sur l'onglet ***Edit Pins***



Supprimer l'entrée « SEL_DP » puis cliquer sur OK

3.3.2 Placer le diviseur comme sur le schéma comme ci-dessous :

Le diviseur va permettre de définir la fréquence de rafraîchissement de l'affichage multiplexé.

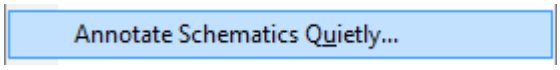


⇒ Placer le composant diviseur sur le schéma

⇒ Relier les fils entre eux

3.4 Numérotation des composants :

Utiliser la fonction automatique : ⇒ **Menu** : TOOLS
⇒ **Commande** : Annotate Schematics Quietly…



4 Définir les fichiers des contraintes.

Les fichiers des contraintes décrivent notamment la connexion broche à broche des fonctions implémentées dans le FPGA. Comme nous travaillons toujours avec la Nanoboard 3000AL2 il est plus rapide de reprendre toujours le même fichier de contraintes fourni par ALTIUM.

Autres rôles des fichiers des contraintes:

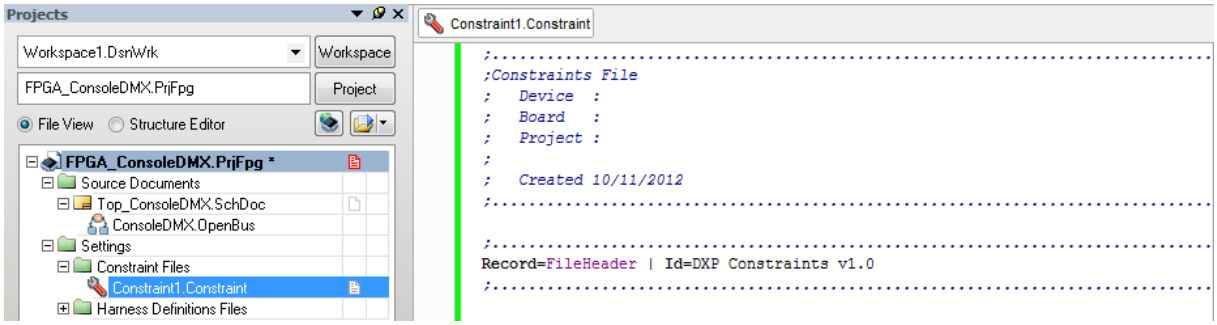
⇒ paramétrer les broches spécifiques tels que l'horloge.

⇒ si nous décrivons un projet à une autre carte que la Nanoboard il faudra alors créer les fichiers contraintes propres à cette carte.

4.1 Le fichier de contrainte sur les horloges :

⇒ Modifier le fichier de contrainte du projet :

Double clic sur le nom du fichier contrainte *(Contraint1.Contraint)*



⇒ Pour compléter ce fichier des contraintes nous allons copier les lignes suivantes dans le fichier de contrainte.

```
;.....................................................................................
Record=FileHeader | Id=DXP Constraints v1.0
;.....................................................................................

Record=Constraint | TargetKind=Port | TargetId=CLK_REF              | FPGA_CLOCK=TRUE
Record=Constraint | TargetKind=Port | TargetId=CLK_REF              | FPGA_CLOCK_FREQUENCY=20 Mhz
Record=Constraint | TargetKind=Port | TargetId=CLK_BRD              | FPGA_CLOCK=TRUE
Record=Constraint | TargetKind=Port | TargetId=CLK_BRD              | FPGA_CLOCK_FREQUENCY=50 Mhz
Record=Constraint | TargetKind=Port | TargetId=JTAG_NEXUS_TCK       | FPGA_CLOCK=TRUE
Record=Constraint | TargetKind=Port | TargetId=JTAG_NEXUS_TCK       | FPGA_CLOCK_FREQUENCY=1 Mhz
;.....................................................................................

;-------------------------------------------------------------------------------------
; Peripheral Board - General I/O   Afficheur Multplexé 3 Digits
;-------------------------------------------------------------------------------------
Record=Constraint | TargetKind=Port| TargetId=SEG[6..0]          | FPGA_PINNUM= AB3,AD1,AE2,AC2,AF3,AH3,AB5
Record=Constraint | TargetKind=Port| TargetId=DIG[2..0]          | FPGA_PINNUM=W10,W8,AC3
Record=Constraint | TargetKind=Port| TargetId=PB0                | FPGA_PINNUM=AA13
Record=Constraint | TargetKind=Port| TargetId=PB2                | FPGA_PINNUM=AF9
Record=Constraint | TargetKind=Port| TargetId=PB4                | FPGA_PINNUM=AH8
Record=Constraint | TargetKind=Port| TargetId=PB6                | FPGA_PINNUM=AG7
;-------------------------------------------------------------------------------------
```

Clic bouton droit sur le nom du fichier contrainte *(Contraint1.Contraint)* dans l'onglet Projets et choisir la commande sauvegarder le document **Save** dans le répertoire de travail « **\TP72_ConsoleDMX** ».

Après :

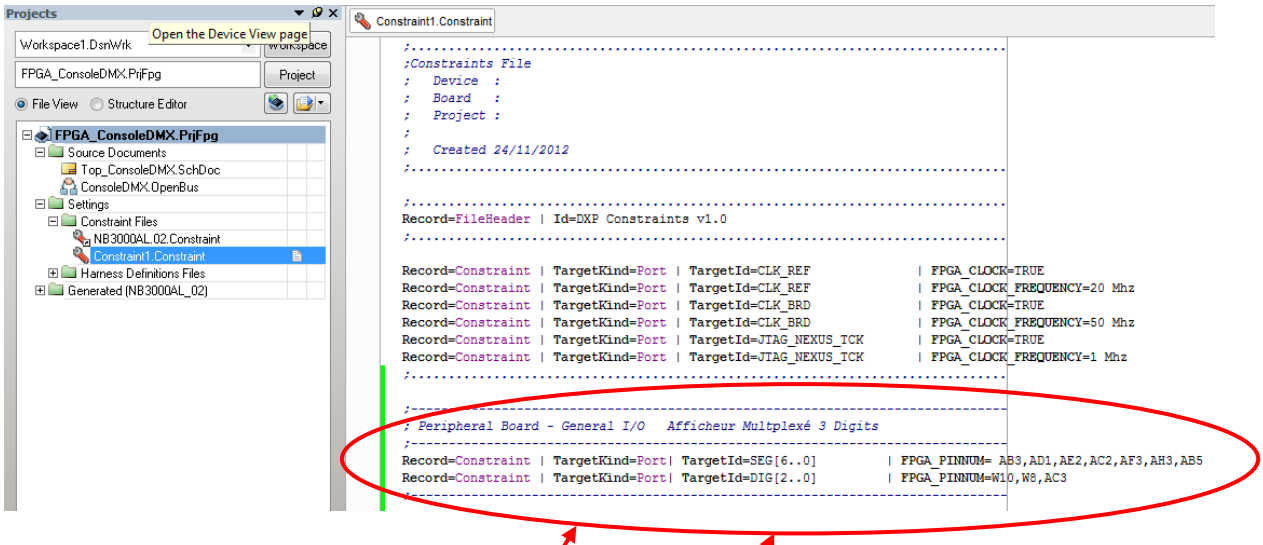


Tableau de correspondance entre l'afficheur multiplexé câblé sur la carte de prototype PB30 et le brochage du FPGA :

Nom netlist sur schéma		TargetId	FPGA_PINNUM	Carte PB30
SEG[6]	Segment a	SEG[6]	AB5	IO17
SEG[5]	Segment b	SEG[5]	AH3	IO19
SEG[4]	Segment c	SEG[4]	AF3	IO21
SEG[3]	Segment d	SEG[3]	AC2	IO23
SEG[2]	Segment e	SEG[2]	AE2	IO25
SEG[1]	Segment f	SEG[1]	AD1	IO27
SEG[0]	Segment g	SEG[0]	AB3	IO29
DIG[0]	Digit unite	DIG[0]	AC3	IO31
DIG[1]	Digit dizaine	DIG[1]	W8	IO33
DIG[2]	Digit centaine	DIG[2]	W10	IO35
SPI_CLK		PB6	AG7	IO7
SPI_DOUT		PB4	AH8	IO5
SPI_DIN		PB2	AF9	IO3
SPI_CS		PB0	AA13	IO1

5 Ouverture du projet embarqué.

Programmation du code C : SOFTWARE DESIGN FLOW

5.1 Ouvrir un projet embarqué existant

Le projet a été ouvert lors de l'ouverture du projet **« FPGA_ConsoleDMX.PrjFpg »**.

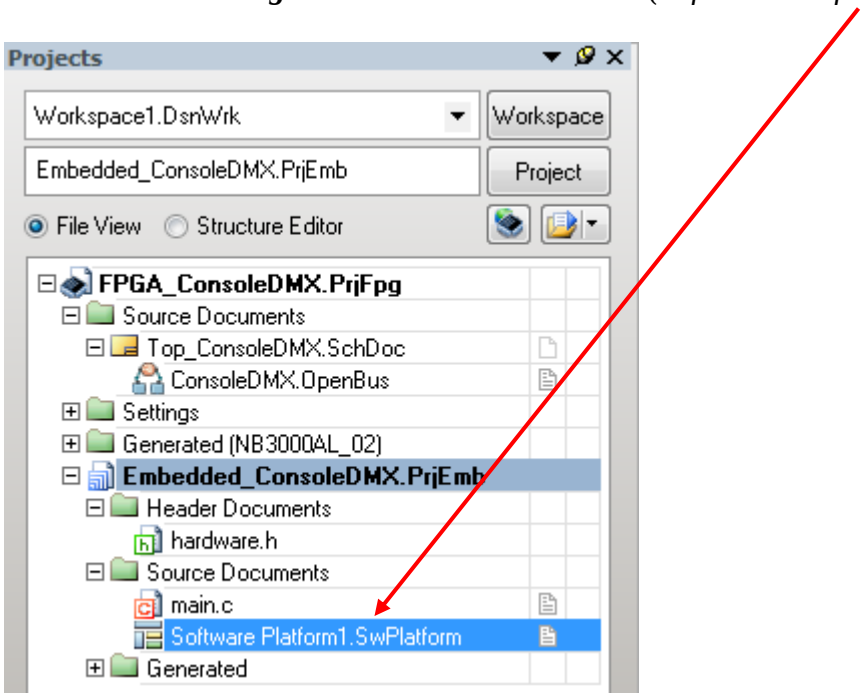
6 Modification du fichier « Software Platform » Mise en place des API

API : Application **P**rogramming **I**nterface
Une **interface de programmation** est une interface fournie par un programme informatique. Elle permet l'interaction entre le programme et les couches matérielles de bas niveaux.

Un exemple : Le paramétrage d'une liaison série : débit binaire (9600 bauds), nombre de bits par octet (8), type de parité (aucune), nombre de bits de stops (1).

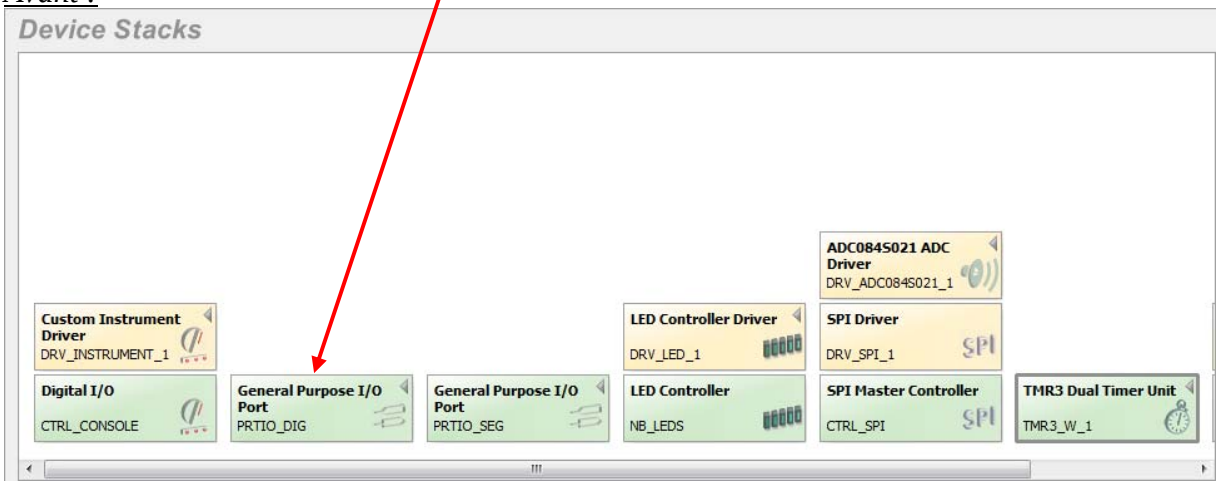
Pour plus d'information lire : Introduction ti the Software platform

Double Clic bouton gauche sur le nom du fichier *(Software Platform1.SwPlatform)*

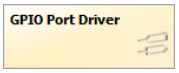


Pour obtenir la couche supérieure du port Digital I/O
⇒ ***Clic bouton droit*** sur l'icône **General Purpose I/O Port**

Avant :

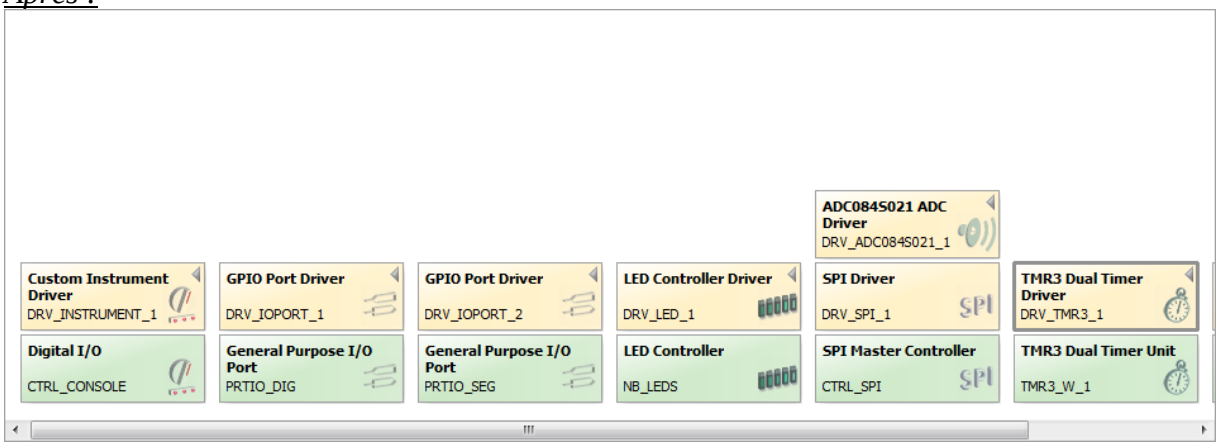


Puis dans l'onglet qui s'ouvre ,

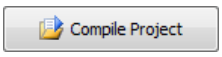
cliquer sur **Grow Stack Up** puis cliquer sur l'icône 

Répéter la commande sur les icônes **General Purpose I/O Port et TMR3 Dual Timer Unit**

Après :



Ces interfaces donnent accès aux différentes fonctions prédéfinies (API) :
- commande des digits de l'afficheur multiplexé (DRV_IOPORT_1)
- commande des segments de l'afficheur multiplexé (DRV_IOPORT_2)
- gestion du timer TMR3 (DRV_TIMER3_1)

=> Cliquer sur 

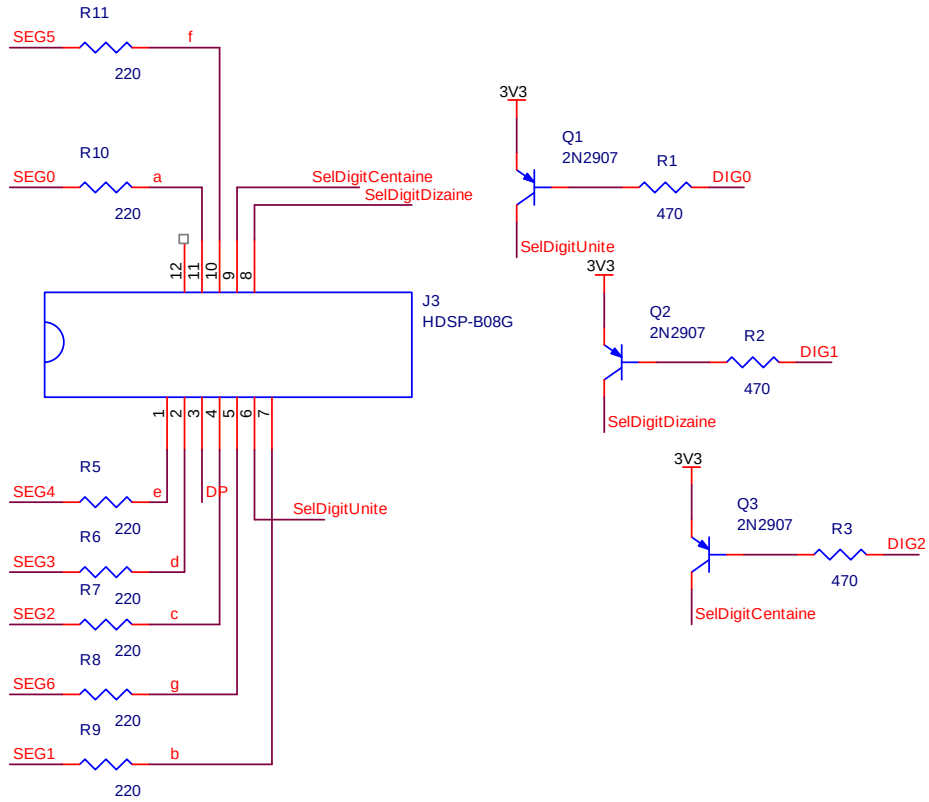
pour générer les fichiers « swplatform.h » et « swplatform.c »

7 Ajout du fichier C principal

7.1 Créer le tableau de codage BCD vers afficheur à 7 Segments

L'afficheur multiplexé est composé par un afficheur à anodes communes composé de 4 digits (HDSP-B08G).

Schéma :



7.1.1 Commande des DIGITS

Quel niveau logique doit-on appliquer sur l'entrée DIG[x] pour sélectionner un afficheur ?
NL0 qui permet de rendre le transistor passant

Compléter le tableau ci-dessous :

		DIG[2]	DIG[1]	DIG[0]	DIG[2..0]
Sélection digit Unité		**1**	**1**	**0**	**0x06**
Sélection digit Dizaine		**1**	**0**	**1**	**0x05**
Sélection digit Centaine		**0**	**1**	**1**	**0x03**

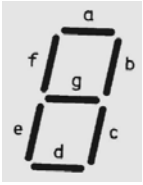
Exemple commande du digit Unité :

```
ioport_set_value(drv_ioport_dig, 0, 0x06);
```

7.1.2 Commande des segments à anode communes

Quel niveau logique doit-on appliquer sur l'entrée d'un SEG[x] pour allumer un segment ?

Compléter le tableau ci-dessous :



Nombre BCD		Segments							
		a	b	c	d	e	f	g	SEG[6..0]
0		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0x01**
1		**1**	**0**	**0**	**1**	**1**	**1**	**1**	**0x4F**
2		**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0x12**
3		**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0x06**
4		**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0x4C**
5		**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0x24**
6		**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0x60**
7		**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0x0F**
8		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0x88**
9		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0x04**

Déclaration du tableau BCD_7Seg_Table en langage C :

```
// Tableau conversion BCD -> 7 Segments
unsigned char BCD_7Seg_Table[16]={0x01,0x4F,0x12,0x06,0x4C,0x24,0x60,0x0F,0x00,0x04,
              0XFF,0xFF,0xFF,0xFF,0XFF,0xFF};
```

Exemple commande du digit Unité :

```
ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Unite]);
```

7.2 Algorithme de l'interruption extérieure

Les variables utilisées dans l'algorithme suivant sont :

Etat_Actuel : variable représente l'état de la machine d'état (Etat_Digit_Unite, Etat_Digit_Dizaine, Etat_Digit_Centaine).
Val_Aff : variable représente le nombre à afficher (0 à 255).

Centaine : mémorise la valeur du digit centaine.
Reste : mémorise un calcul intermédiaire.
Dizaine : mémorise la valeur du digit dizaine.
Unité : mémorise la valeur du digit unité.

Début interruption extérieure 2

```
Centaine ← Val_Aff / 100 ;          // Extrait le digit Centaine à partir de Val_Aff
Reste ← Val_Aff % 100 ;     // Extrait les digits Dizaine et Unité à partir de Val_Aff
Dizaine ← Reste / 10 ;                    // Extrait le digit Dizaine à partir de Reste
Unité ← Reset % 10 ;                // Extrait le digit Unité à partir de Reste

Suivant (Etat_Actuel) Faire
    Cas (Etat_Actuel == Etat_Digit_Unite)
        Sélectionner Digit Unité;
        Afficher Caractère Unité;
        Etat_Actuel ← Etat_Digit_Dizaine;

    Cas (Etat_Actuel == Etat_Digit_Dizaine)
        Sélectionner Digit Dizaine ;
        Afficher Caractère Dizaine;
        Etat_Actuel ← Etat_Digit_Centaine;

    Cas (Etat_Actuel == Etat_Digit_Centaine)
        Sélectionner Digit Centaine ;
        Afficher Caractère Centaine;
        Etat_Actuel ← Etat_Digit_Unite;

    Sinon Etat_Actuel ← Etat_Digit_Unite ;

FinSuivant
```

Fin

7.3 Coder l'algorithme du programme d'interruption en langage C

A partir de l'algorithme, coder le programme d'interruption en langage C

```
/*******************************************/
/* Déclaration de la fonction d'interruption */
/*******************************************/

void __interrupt (INTNUMBER_EXT2) it_exterieure (void)
{
    Centaine=(Val_Aff/100);
    Reste=  Val_Aff%100;
    Dizaine=Reste/10;
    Unite= Reste%10;
    switch ( Etat_Actuel )
    {
        case Etat_Digit_Unite:
        {
            ioport_set_value( drv_ioport_dig, 0, UNITE);      // Sélection Digit Unité
            ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Unite]);
            Etat_Actuel=Etat_Digit_Dizaine;
        }
        break;
        case Etat_Digit_Dizaine:
        {
            ioport_set_value( drv_ioport_dig, 0, DIZAINE);      // Sélection Digit Dizaine
            ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Dizaine]);
            Etat_Actuel=Etat_Digit_Centaine;
        }
        break;
        case Etat_Digit_Centaine:
        {
            ioport_set_value( drv_ioport_dig, 0, CENTAINE);      // Sélection Digit Centaine
            ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Centaine]);
            Etat_Actuel=Etat_Digit_Unite;
        }
        break;
        default:  Etat_Actuel=Etat_Digit_Unite;
        break;
    }
}
```

7.4 Copier le fichier « main.c »

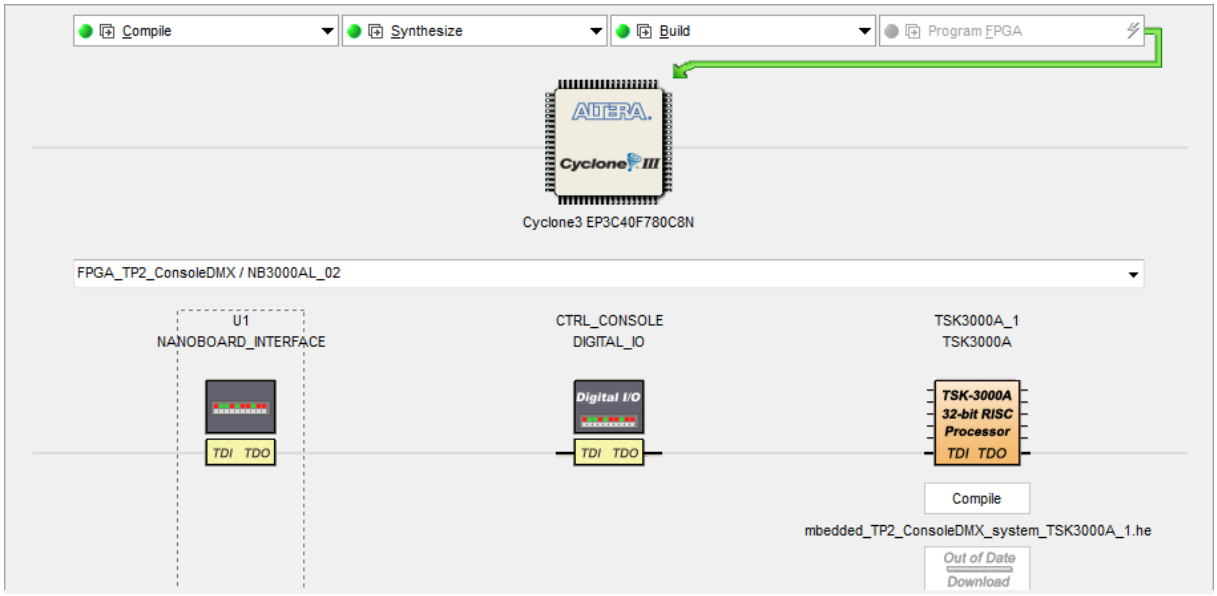
Copier le fichier ci-dessous dans le fichier main.c puis enregistrer et compiler le projet.

```
/*************************************/
/* Déclaration des fichiers d'entête */
/*************************************/
#include <swplatform.h>
#include <stdio.h>

/*************************************/
/* Déclaration des équivalences      */
/*************************************/
#define UNITE 0x06
#define DIZAINE 0x05
#define CENTAINE 0x03

/*************************************/
/* Déclaration des variables         */
/*************************************/
unsigned char DMX_Table[512];          // Déclaration d'un tableau de 512 caractères
// Tableau conversion BCD -> 7 Segments
unsigned char BCD_7Seg_Table[16]={0x01,0x4F,0x12,0x06,0x4C,0x24,0x60,0x0F,0x00,0x04,
              0XFF,0xFF,0xFF,0xFF,0XFF,0xFF};

uint8_t Val_Aff=0;
//uint8_t Etat=0;
uint8_t Centaine,Dizaine,Unite, Reste;

 /* etats */
typedef enum
{
   Etat_Digit_Unite,
   Etat_Digit_Dizaine,
   Etat_Digit_Centaine,
}
etat_type;

etat_type Etat_Actuel = Etat_Digit_Unite ;


/*******************************************/
/* Déclaration de la fonction d'interruption */
/*******************************************/

void handler_timer_a (tmr3_t *drv_tmr3_1, void *data)
{
    Centaine=(Val_Aff/100);
    Reste=  Val_Aff%100;
    Dizaine=Reste/10;
    Unite= Reste%10;
    switch ( Etat_Actuel )
    {
        case Etat_Digit_Unite:
        {
            ioport_set_value( drv_ioport_1, 0, UNITE);      // Sélection Digit Unité
            ioport_set_value( drv_ioport_2, 0, BCD_7Seg_Table[Unite]);
            Etat_Actuel=Etat_Digit_Dizaine;
        }
        break;
        case Etat_Digit_Dizaine:
        {
            ioport_set_value( drv_ioport_1, 0, DIZAINE);      // Sélection Digit Dizaine
            ioport_set_value( drv_ioport_2, 0, BCD_7Seg_Table[Dizaine]);
            Etat_Actuel=Etat_Digit_Centaine;
        }
        break;
        case Etat_Digit_Centaine:
        {
            ioport_set_value( drv_ioport_1, 0, CENTAINE);      // Sélection Digit Centaine
            ioport_set_value( drv_ioport_2, 0, BCD_7Seg_Table[Centaine]);
            Etat_Actuel=Etat_Digit_Unite;
        }
        break;
        default:  Etat_Actuel=Etat_Digit_Unite;
        break;
    }
}

/*******************************************/
/* Déclaration de la fonction initialisation */
/*******************************************/

int Initialize(void)
{
```

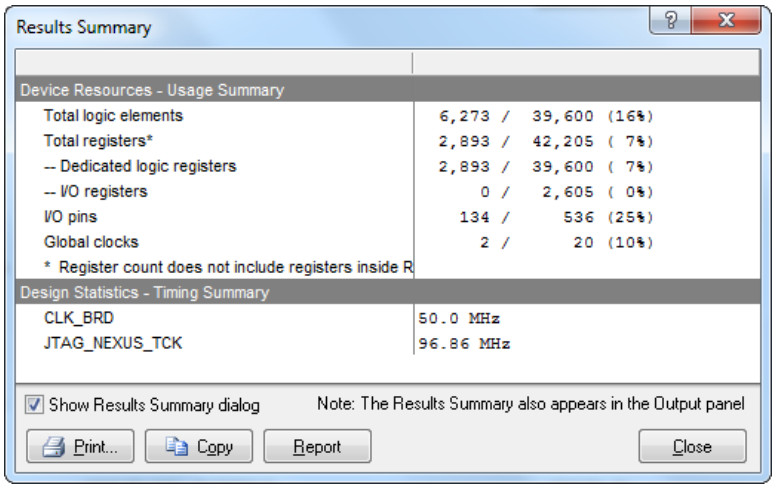
```
    // Ouverture du périphérique virtuel instrument INTRUMENT
    drv_instrument_1 = instrument_open(DRV_INSTRUMENT_1);
    // Ouverture du périphérique DRV_LED
    drv_led_1 = led_open(DRV_LED_1);
    led_turn_all_off(drv_led_1);                        // Extinction des leds

    // Ouverture du périphérique port série UART
    drv_uart8_1  = uart8_open(DRV_UART8_1);
     // 250kbauds, pas de parité, 8 bits de données et 2 bits de stops entre l'envoi de 2 caractères
    uart8_set_parameters(drv_uart8_1,250000, UART8_NO_PARITY,8,2);

    // Ouverture du périphérique port convertisseur ADC084S021
    drv_adc084s021_1= adc084s021_open(DRV_ADC084S021_1);

    // Ouverture des ports IO
    drv_ioport_1 = ioport_open(DRV_IOPORT_1);
    drv_ioport_2 = ioport_open(DRV_IOPORT_2);

    interrupts_enable( );                     // Autorisations des interruptions
    // Ouverture du périphérique port Timer 3
    drv_tmr3_1 = tmr3_open(DRV_TMR3_1);

    tmr3_timer_a_set_handler(drv_tmr3_1, handler_timer_a, NULL);
    // Timer 3 : Gate a non Activitée, Mode Compteur, Compteur qui permet de diviser par 128
    tmr3_timer_a_8bit_autoreload_mode(drv_tmr3_1, 0, 1, 128);
    tmr3_timer_a_start(drv_tmr3_1);               // Lancement du Timer3

    return 1;
}

/*****************************************/
/* Déclaration de la fonction initialisation */
/*****************************************/
int main(void)
{
    Initialize();                             // Appel de la fonction d'initialisation

    while (1)
    {
         // Lecture ADC     int adc084s021(adc_t *adc, unsigned channel)

        for (uint8_t i = 0; i < 4; i++) // Lecture des 4 consignes Rouge , Vert, Bleu et Dimmer  de
l'instrument virtuel
        {
            DMX_Table[i]= (char) adc084s021_read(drv_adc084s021_1,i+1);  // Stockage résultat conversion
dans un tableau
            delay_us(10);                                    // Temporisation de 10 us entre 2 conversions
            led_set_intensity(drv_led_1, i, DMX_Table[i]);       // Envoi commande des leds RVB
            instrument_set_value(drv_instrument_1,i,DMX_Table[i]);  // Envoi résultat conversion vers
intrument virtuel
        }
        Val_Aff = DMX_Table[0];

        uart8_putbreak(drv_uart8_1, 22);         // Envoi du break d'une durée de 22 Tbits
        uart8_putchar(drv_uart8_1, 0);           // Envoi du caractère '0'
        for (int i = 0; i < 512; i++)                        // Envoi des 4 consignes du buffer DMX sur la
liaison DMX
        {
            while (!uart8_transmit_buf_free(drv_uart8_1));       // Test que le buffer d'émmission est vide
            uart8_putchar(drv_uart8_1, DMX_Table[i]);            // Envoi du caractère vers l'UART
        }
    }
}
/*** FIN DU PROGRAMME PRINCIPAL ***/
```

8 Compiler, Synthétiser, construire, Programmer le FPGA.

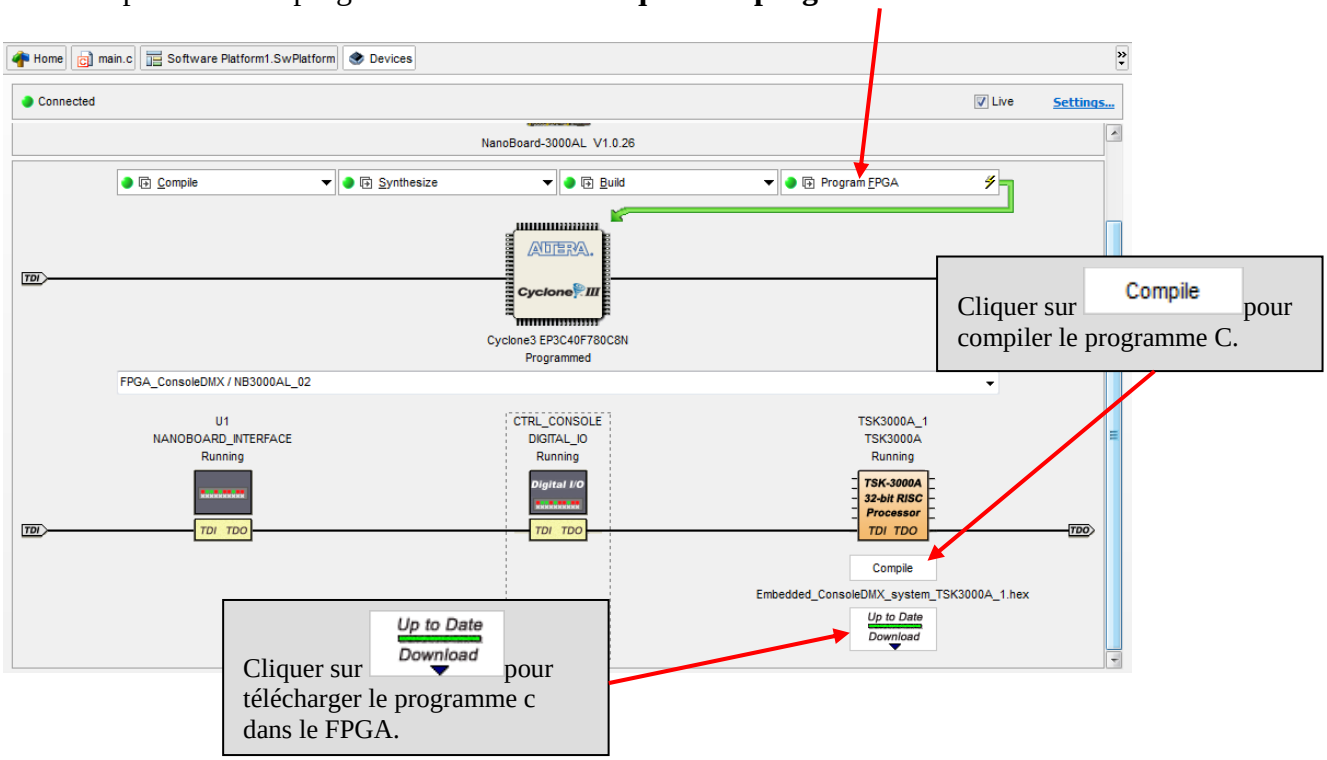
⇒ Cliquer sur Compile, cliquer sur Synthetize, cliquer sur Build.
⇒ Si une erreur apparaît vous devez la corriger en modifiant le fichier source identifié à partir du message d'erreur :
Sources d'erreurs possibles : ⇒ le fichier OpenBus
⇒ le schéma TOP
⇒ les fichiers de contraintes



Après avoir franchi ces trois étapes la fenêtre de résultats s'affiche



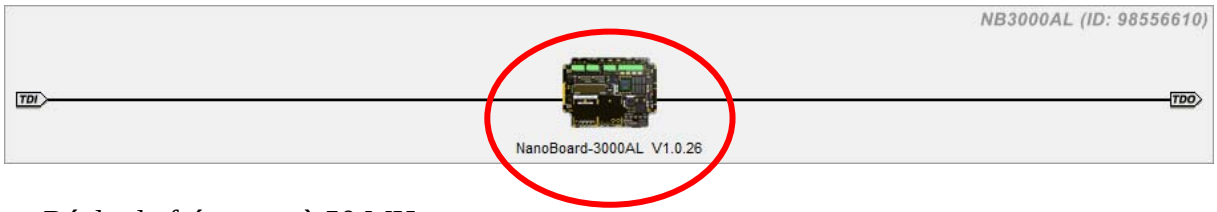
=> Vous pouvez alors programmer le FPGA : **cliquer sur program FPGA** !



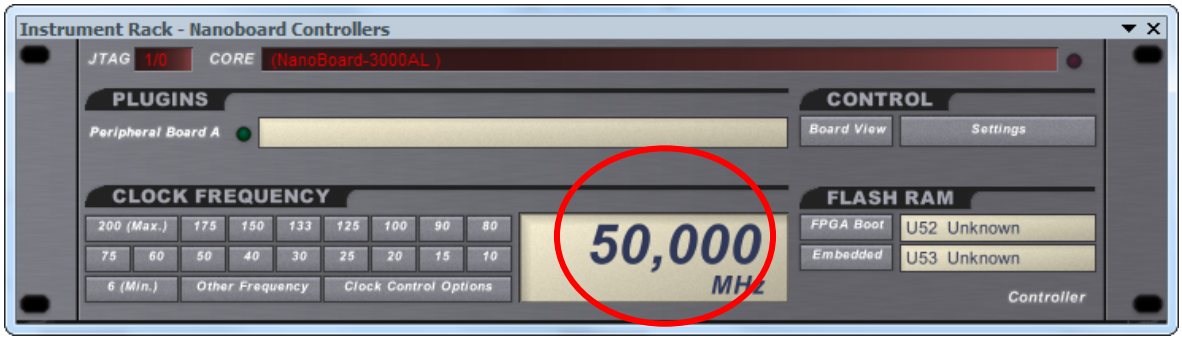
9 Mettre en œuvre les instruments de mesures virtuels et réels

9.1 Réglage de la fréquence entrante par U3 : le générateur d'horloge

⇒ Cliquer sur NanoBoard-3000AL:



⇒ Régler la fréquence à 50 MHz.

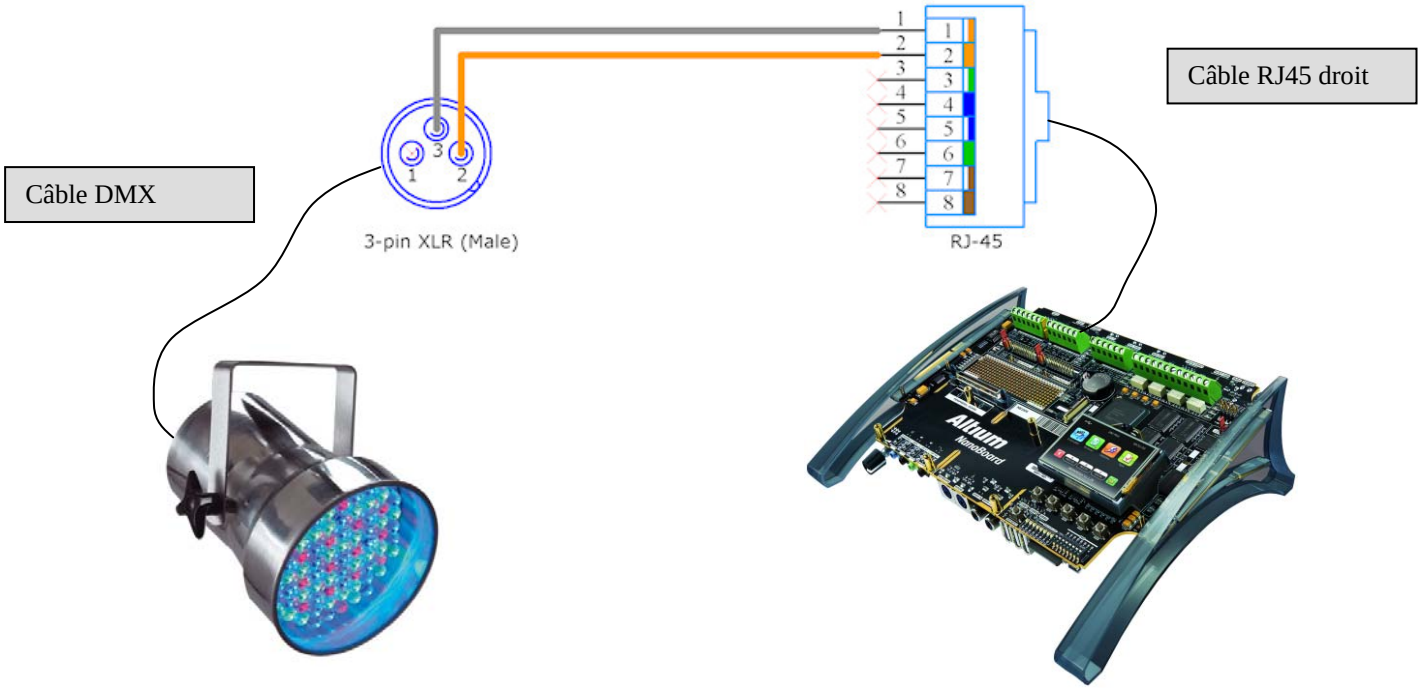


Il est important de paramétrer correctement la fréquence de l'horloge car celle-ci détermine la durée Tbit sur la liaison série RS 485.

9.2 Branchement de la NB3000 avec le projecteur à Leds

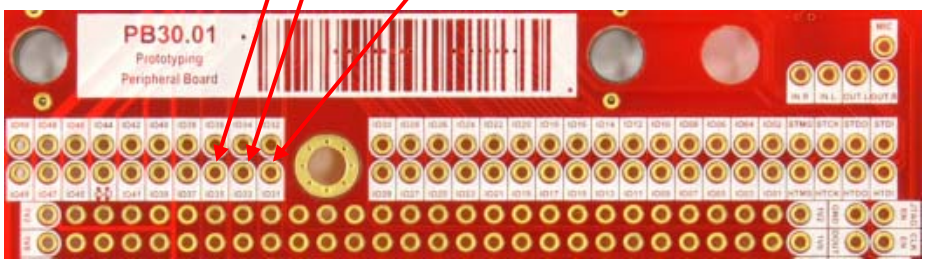
Brancher l'adaptateur entre la sortie RS485 (prise RJ45) et le câble DMX. Puis relier le câble DMX au projecteur à leds adressé sur le canal n°1.

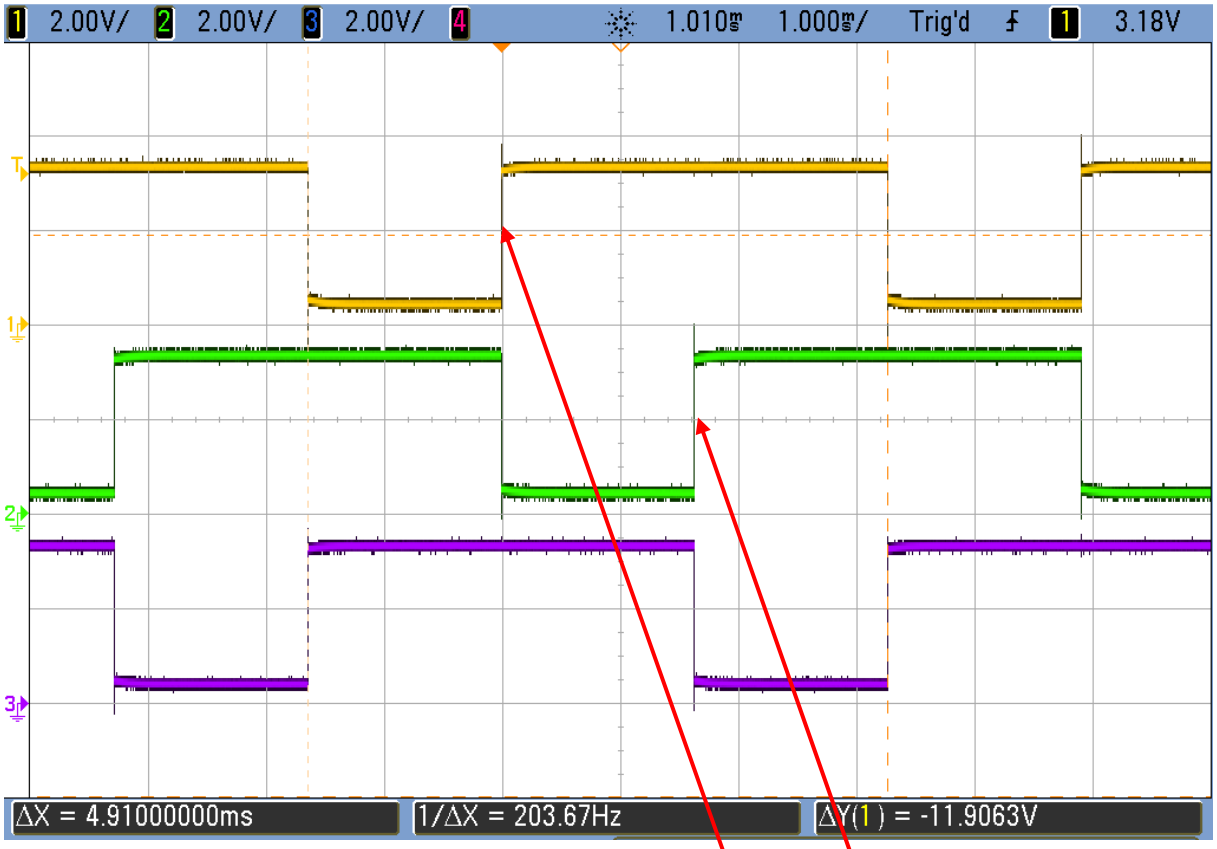
Pin 1 : DMMX OUT GROUND
Pin 2 : DMX OUT DATA -
Pin 3 : DMX OUT DATA +



9.3 Relevés des chronogrammes sur le connecteur de la PB30

Voie 1 : DIG[0] borne IO31 de PB30
Voie 2 : DIG[1] borne IO33 de PB30
Voie 3 : DIG[2] borne IO35 de PB30





Appel interruption

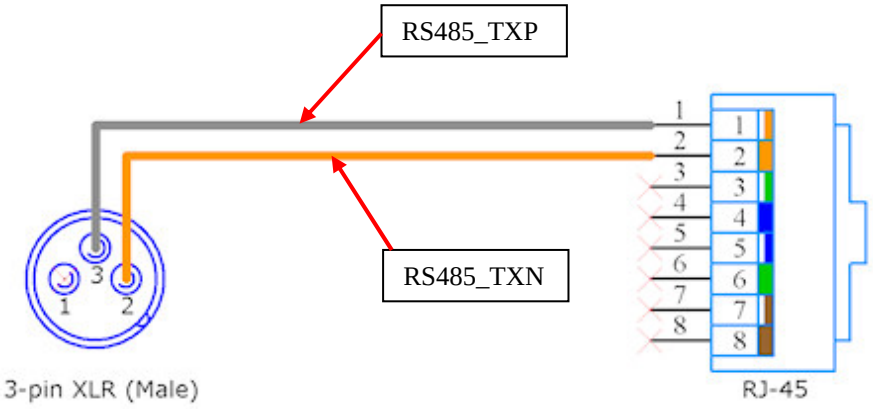
Mesurer la fréquence de rafraîchissement ?
203.67 Hz

Calculer la fréquence du signal entrant sur la broche ? Puis la comparer avec la fréquence de rafraîchissement ?
Freq Int_I2 = (CLK_REF / 256) /128 = 610.35 Hz avec CLK_REF = 20 MHz.
Cette fréquence est 3 fois plus grande, ce qui normal car l'interruption est appelée à chaque front montant

Comparer cette fréquence de rafraîchissement avec la persistance rétinienne ?
La fréquence de rafraîchissement est très supérieure à la persistance rétinienne qui est de 50Hz.

Annexe 1:

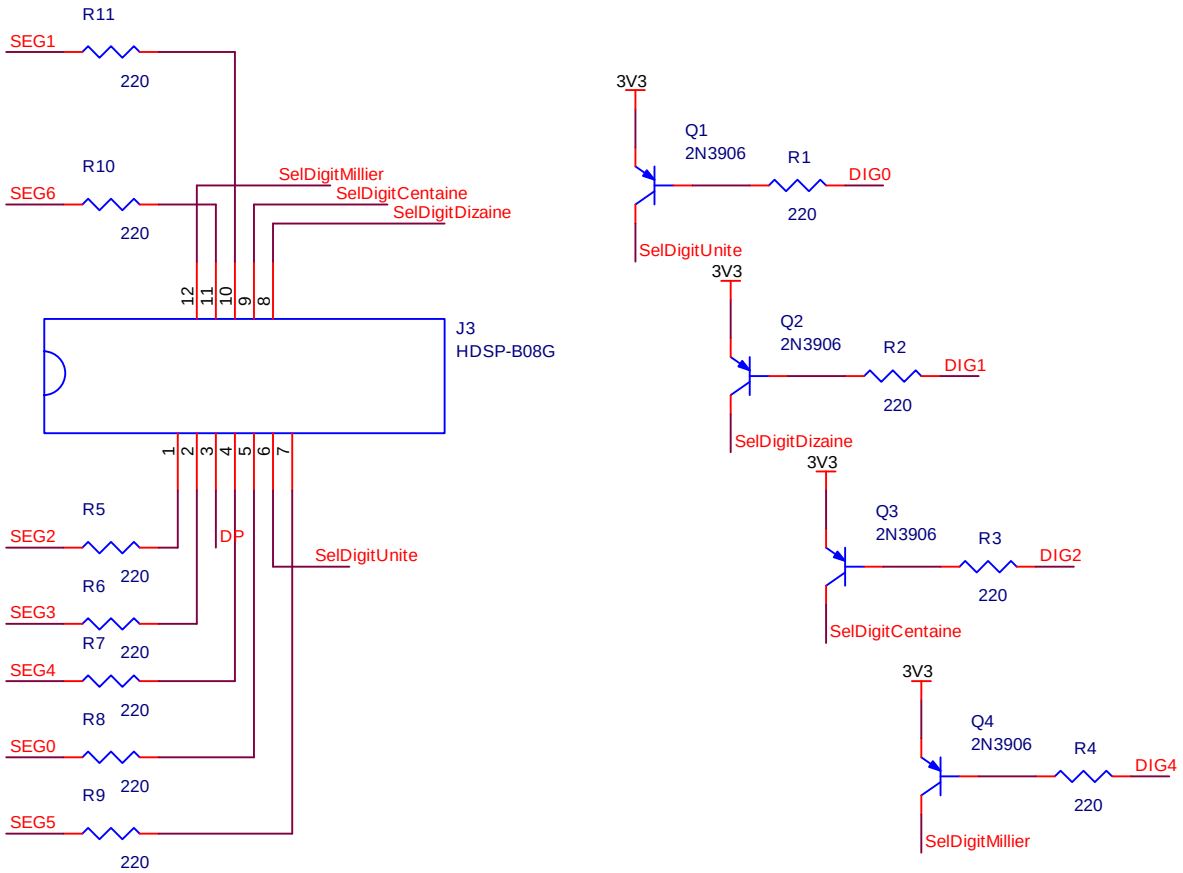
Câble pour adapter la prise RS-485 aux interfaces a 3 fiches DMX-512.



Pin 1 : DMMX OUT GROUND
Pin 2 : DMX OUT DATA -
Pin 3 : DMX OUT DATA +

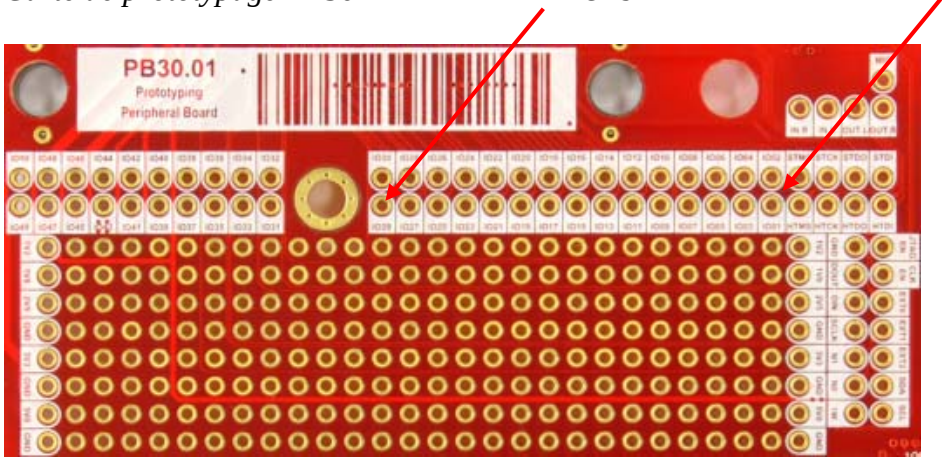
Annexe 2:

Schéma de l'afficheur multiplexé



Annexe 3:

Carte de prototypage PB30 *IO29* *IO1*



Fichier de contrainte pour la carte de prototype PB30 :

```
;--------------------------------------------------------------------------
; Peripheral Board - General I/O
;--------------------------------------------------------------------------
;Record=Constraint | TargetKind=Port | TargetId=PBA[49..0] |
FPGA_PINNUM=V6,U3,T8,U7,AA5,V2,U8,W2,AB7,AA6,AB4,AD5,V3,W3,Y3,W10,V8,W8,AD3,AC3,AB2,AB3,AC1,AD1,Y7,AE2,AE
3,AC2,AE4,AF3,AG3,AH3,AF5,AB5,AD7,AE1,AH4,AG6,AF6,AF2,AF7,AB12,AF8,AG7,AH7,AH8,AG10,AF9,AH10,AA13
;--------------------------------------------------------------------------
```

Annexe 4 :

Photo de l'afficheur multiplexé câblé sur la carte de prototypage PB30