



## 0 - Présentation de l'activité 3 Console DMX :

Traitement numérique du signal. Acquisition d'une grandeur analogique par un convertisseur ADC. Relever les signaux sur le bus SPI

Pré-requis :           ⇒ Avoir réaliser l'intégralité de l'activité 2  
Durée estimée :       ⇒ 4 heures  
Objectif :             ⇒ Implanter et programmer un **processeur** dans un **FPGA**.

Vocabulaire spécifique :

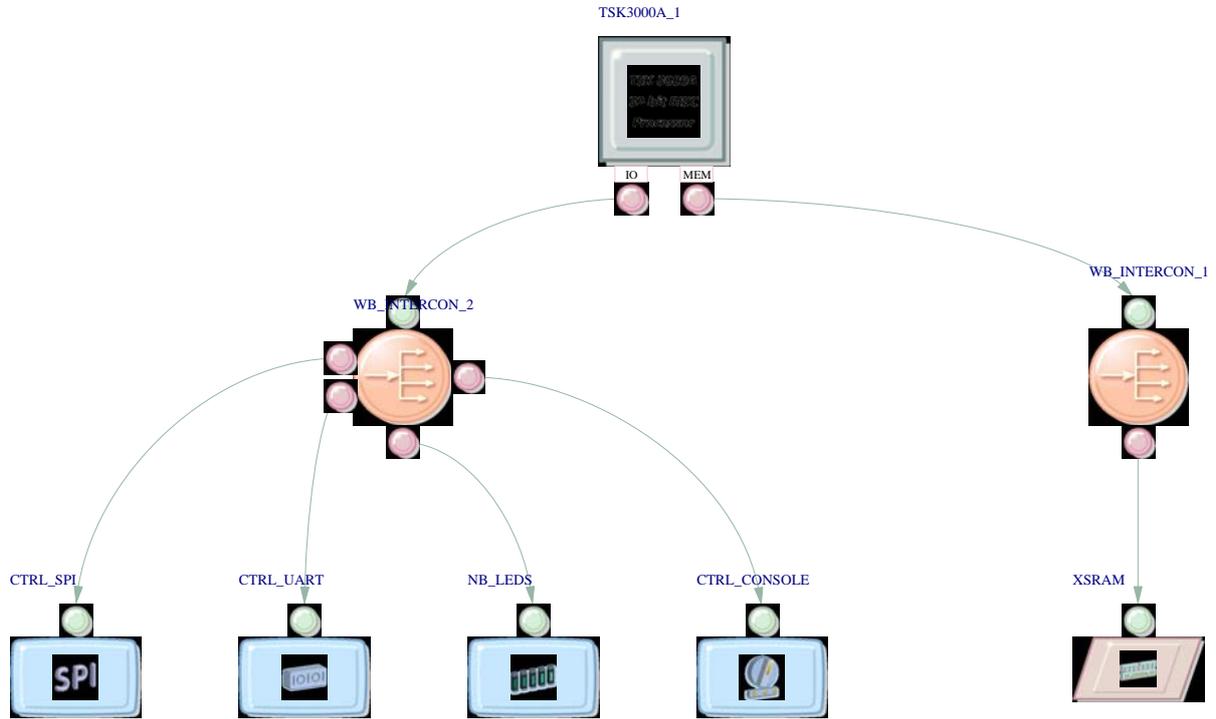


Sommaire de l'activité 3 du mini projet :

- 1 Créer un nouveau projet FPGA.
- 2 Editer le fichier OpenBus.
- 3 Créer le « TOP » schéma.
- 4 Définir les fichiers de contraintes.
- 5 Création du projet embarqué.
- 6 Construction du fichier « Software platform » Mise en place des API.
- 7 Ajout du fichier C principal
- 8 Compiler, Synthétiser, construire, Programmer le FPGA.
- 9 Mettre en œuvre les instruments de mesure virtuels.

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

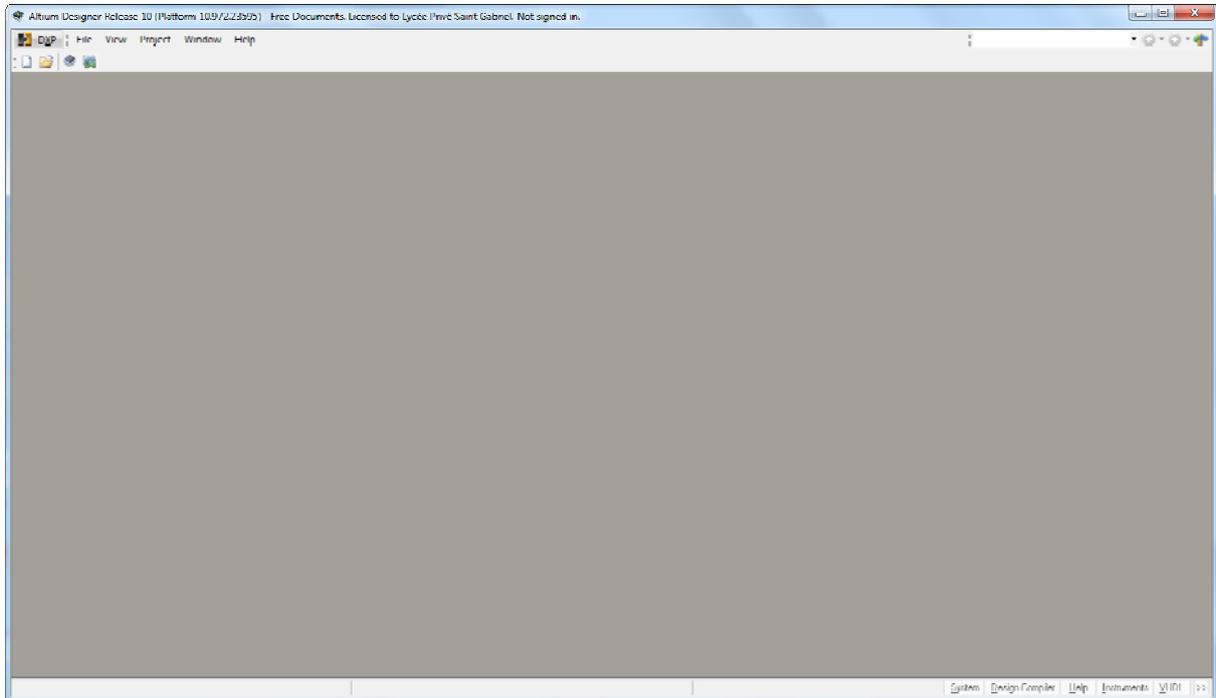
Schéma OpenBus à dessiner:





## 1 Créer un nouveau projet FPGA

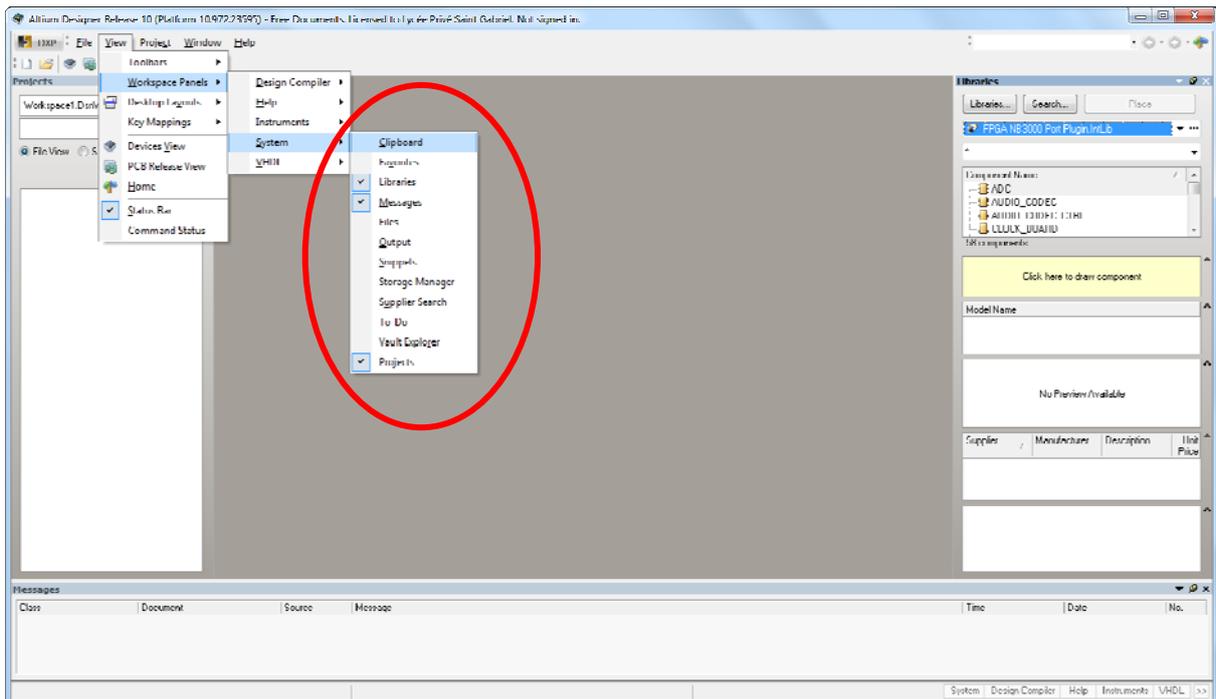
### 1.1 Repartir d'un environnement vide :



### 1.2 Ouvrir les fenêtres projet et messages :

Paramétrer l'environnement de travail d'Altium Designer en utilisant la commande :

**View >> Workspace Panels >> System**, puis cocher les options **Libraries**, **Messages** et **Projet**



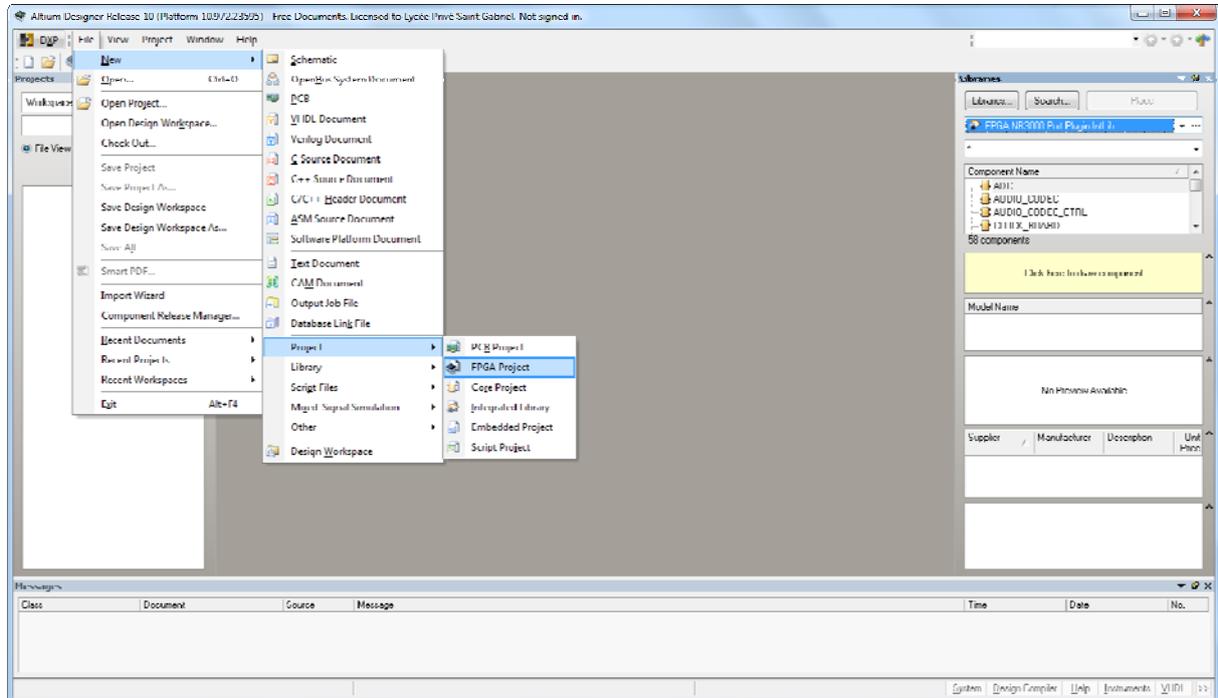
Activité 3 Mini projet DMX-FPGA : **Implanter un processeur dans le FPGA**  
Durée estimée 4h

**Programmer le processeur**

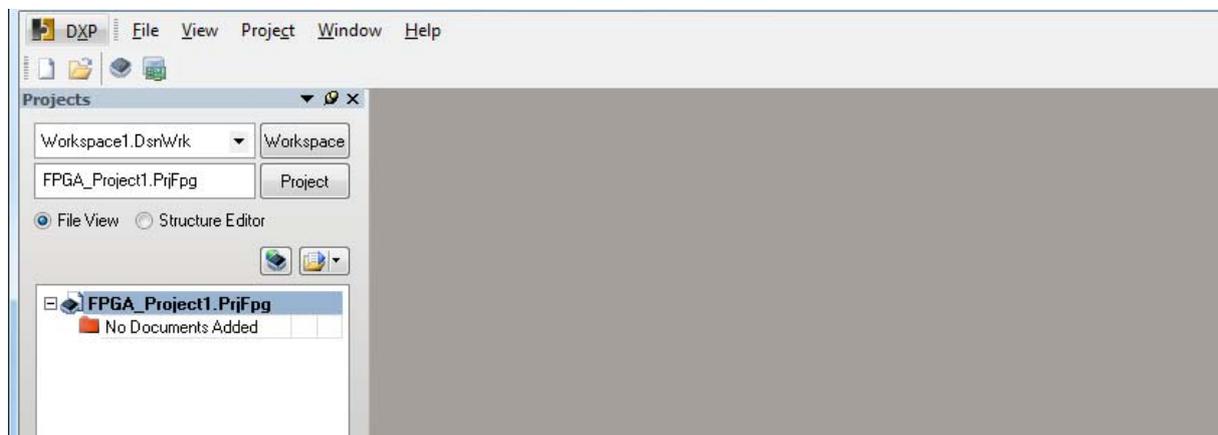


1.3 Créer et renommer le projet :

Créer un nouveau projet en utilisant la commande : **File >> New >> Projet >> FPGA Project.**



Un projet nommé (*FPGA\_Projet1.PrjFPGA*) apparaît dans l'onglet gestion de projet.



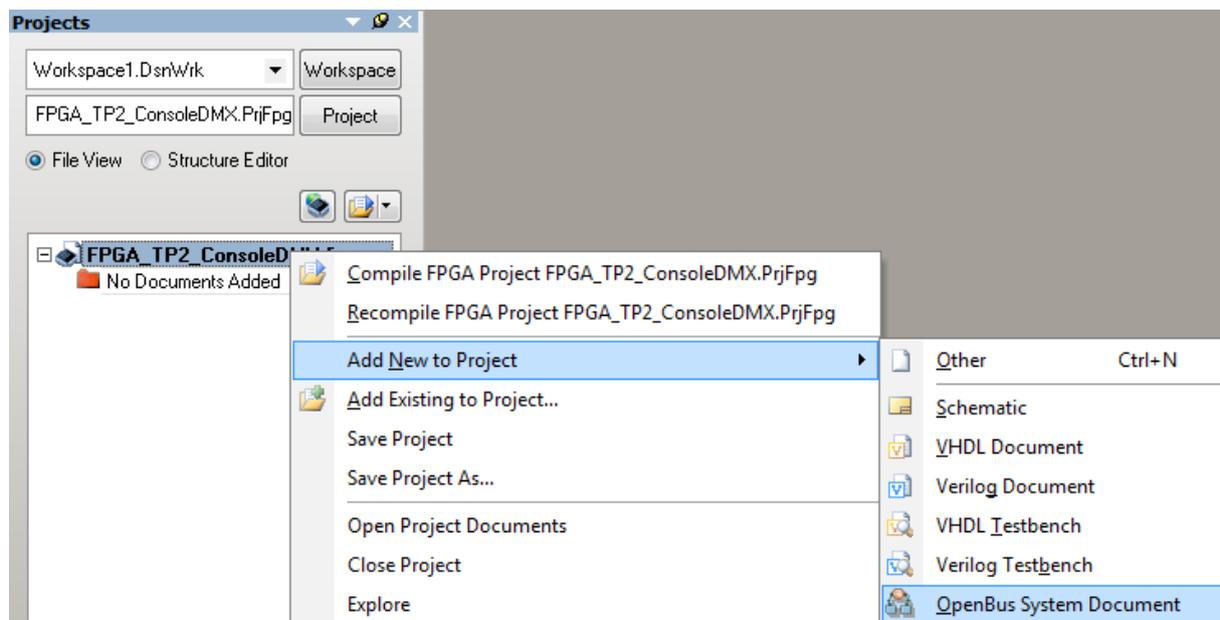
**Clic bouton droit** sur le nom du nouveau projet (*FPGA\_Projet1.PrjFpg*) dans l'onglet Projets et choisir la commande **Save Project as** « **FPGA\_TP72\_ConsoleDMX.PrjFpg** » pour sauvegarder le projet dans le répertoire de travail « **\TP72\_ConsoleDMX** »

Remarque : Les caractères espace ( ) et/ou tiret (-) ne doivent pas être utilisés dans les noms du projet ou des documents. Le caractère underscore ( \_ ) peut être utilisé pour améliorer la lisibilité.

## 2 Editer le fichier OpenBus

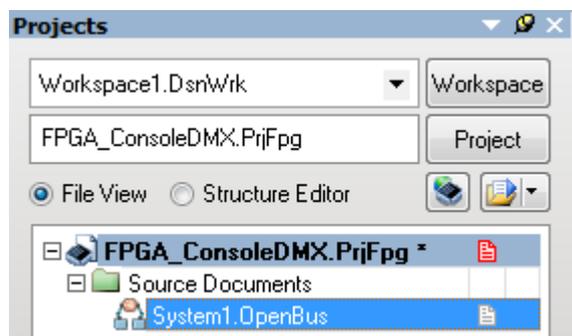
### 2.1 Ajouter au projet un fichier OpenBus et sauvegarder ce fichier

Ajouter un nouveau fichier OpenBus par un *Clic bouton droit* sur le nom du projet FPGA dans l'onglet Projets et choisir la commande **Add New to Project>> OpenBus System Document**

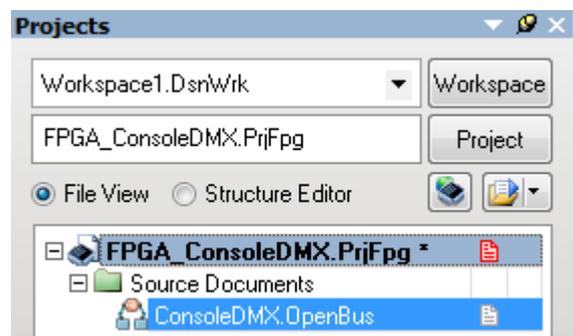


*Clic bouton droit* sur le nom du nouveau fichier OpenBus (*System1.OpenBus*) dans l'onglet Projets et choisir la commande sauvegarder le document **Save As** avec le nom *ConsoleDMX.OpenBus* dans le même dossier parents du projet.

Avant :

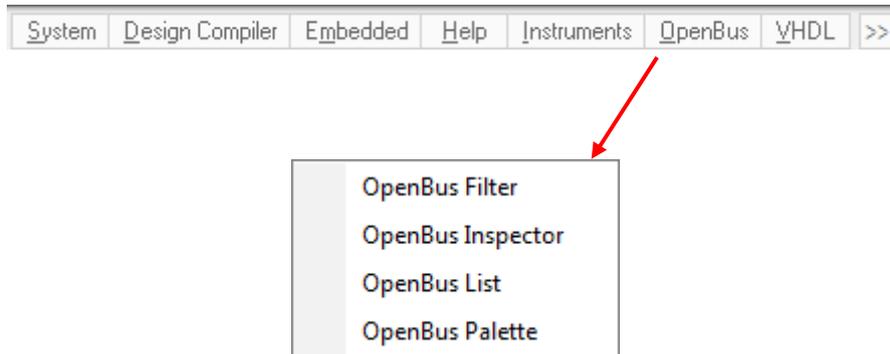


Après :

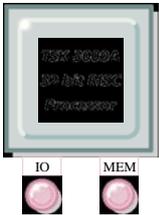
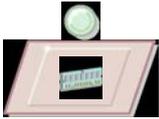


## 2.2 Placer les éléments OpenBus suivants afin de dessiner le schéma

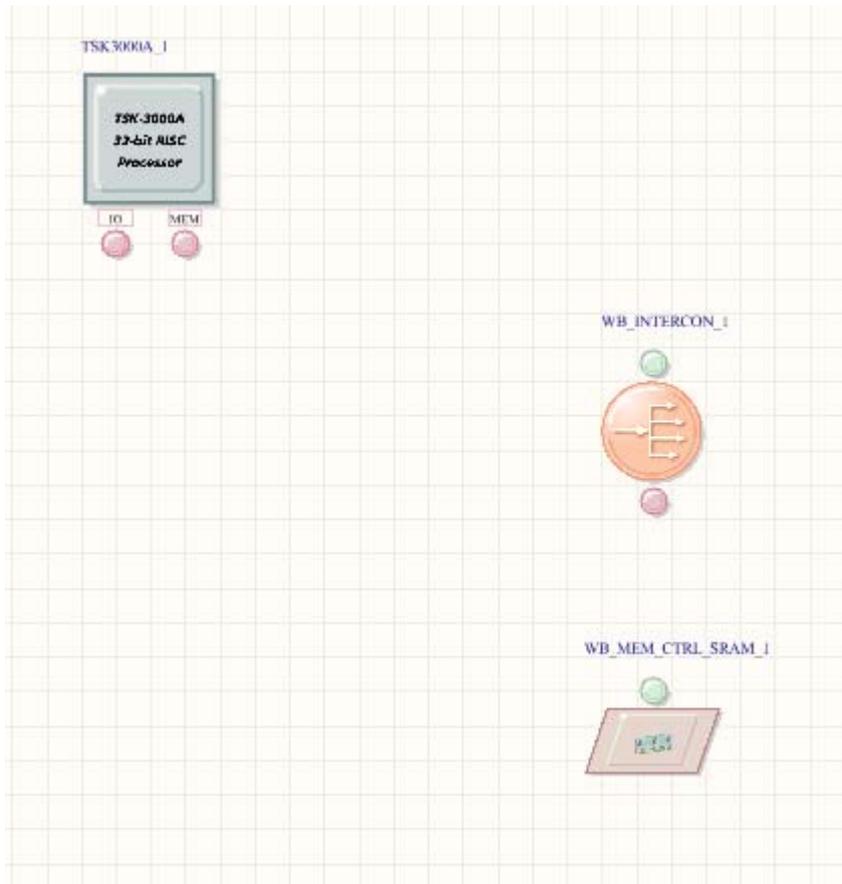
**Clic bouton droit** sur la feuille de travail OpenBus et choisir dans l'onglet qui apparaît la commande **WorkspacePanels >> OpenBus >> OpenBus Palette**



Le tableau ci-dessous identifie les composants OPEN\_BUS à placer sur le bus mémoire

Symbole	Nom du composant	Description
<p>TSK3000A_1</p> 	TSK3000A	Processeur RISC 32 bits
<p>WB_INTERCON_1</p> 	Interconnect	Permet de connecter la mémoire SRAM au bus du processeur
<p>WB_MEM_CTRL_SRAM_1</p> 	Contrôleur mémoire SRAM	Permet de connecter la mémoire statique asynchrone (SRAM) organisée en 2 x 16 bits de la Nano Board au bus du mémoire du processeur

=> Placer les composants sur la feuille de schéma OpenBus

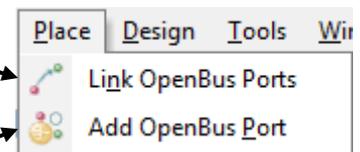


Remarque : la touche d'espace permet de faire tourner les composants

=> Dessiner les liens entre les éléments OpenBus:

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

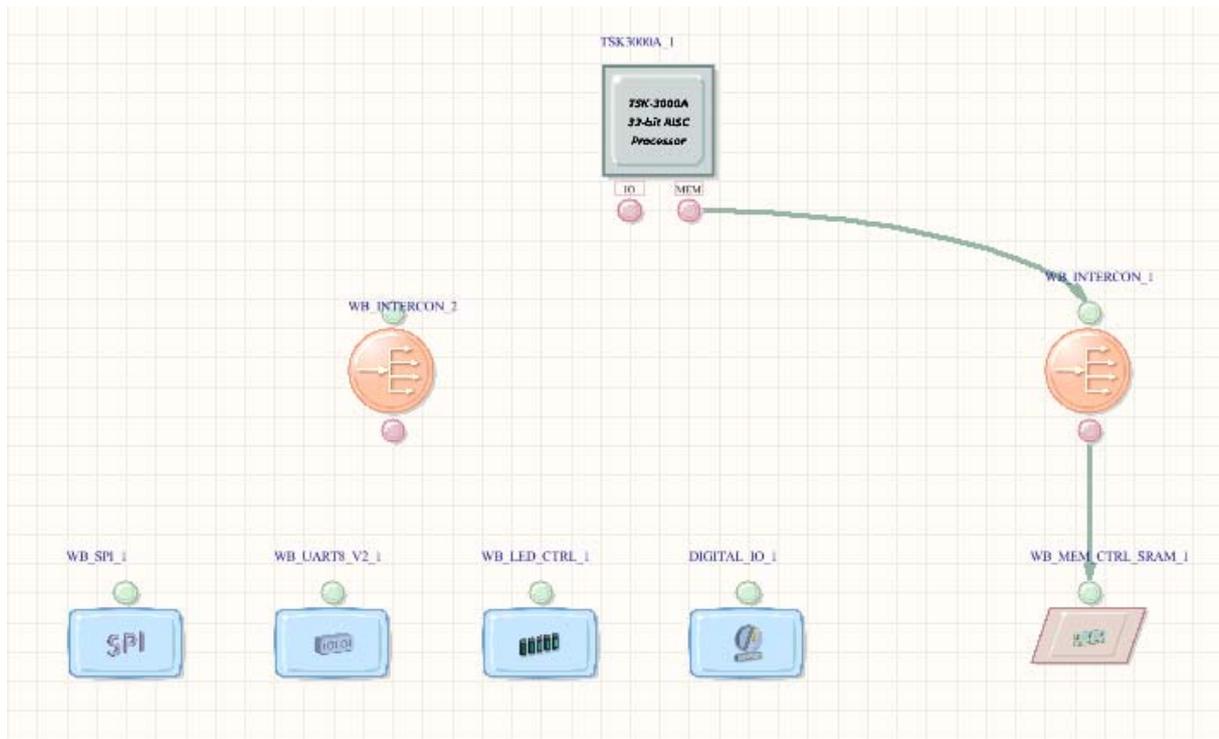
Pour rajouter les ports aux inter-connexions utiliser la fonction ADD OpenBus port



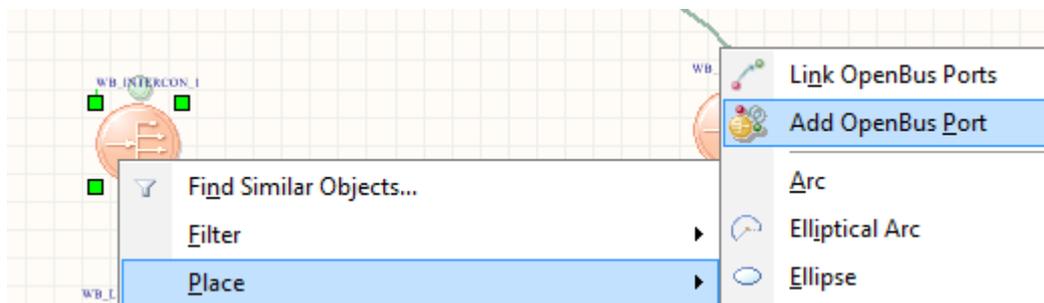
Le tableau ci-dessous identifie les composants OpenBus à placer sur le bus périphérique

Symbole	Nom du composant	Description
WB_INTERCON_2 	Interconnect	Permet de connecter les périphériques au bus du processeur.
WB_UART8_V2_1 	Serial Communications Port	Permet d'envoyer des caractères vers une liaison série asynchrone
WB_LED_CTRL_1 	LED Controller	Permet de contrôler les leds de la Nano Board
WB_SPI_1 	SPI	Permet de contrôler par un bus SPI le convertisseur ADC de la Nano Board
DIGITAL_IO_1 	Digital IO	Permet de lancer un instrument virtuel pour visualiser le résultat de la conversion AN

=> Placer les composants sur la feuille de schéma Open Bus



=> Ajouter un port OpenBus au composant WB\_INTERCON\_2 :

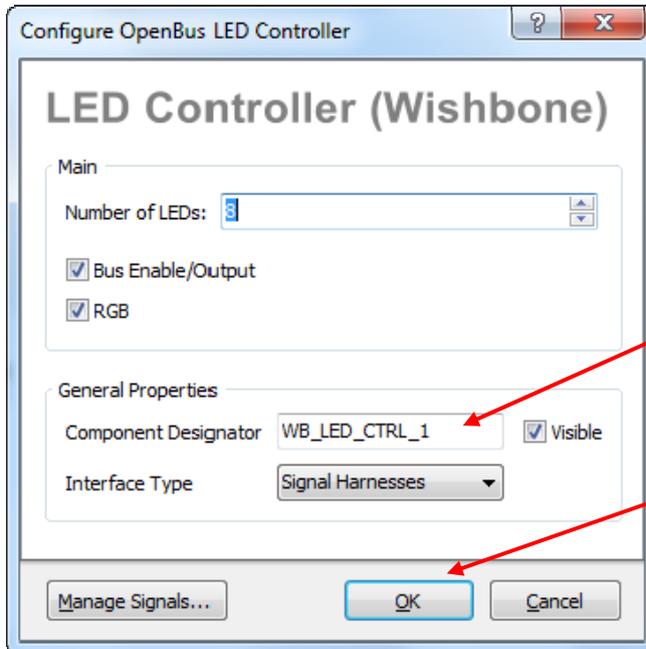


=> Dessiner les liens entre les éléments OpenBus :



=> Paramétrer et renommer le composant WB\_LED\_CTRL\_1

**Clic bouton droit** sur le composant WB\_LED\_CTRL\_1, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus LED Controller**



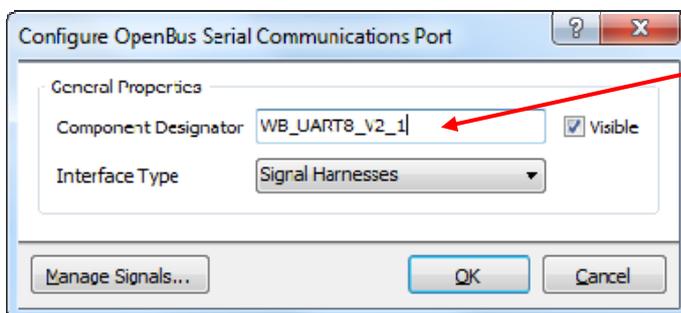
Renommer en NB\_LEDS

Cliquer sur OK pour valider les changements



=> Paramétrer et renommer le composant WB\_UART8\_V2\_1

**Clic bouton droit** sur le composant WB\_UART8\_V2\_1, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus Serial Communications Port**



Renommer en CTRL\_UART

Cliquer sur OK pour valider les changements

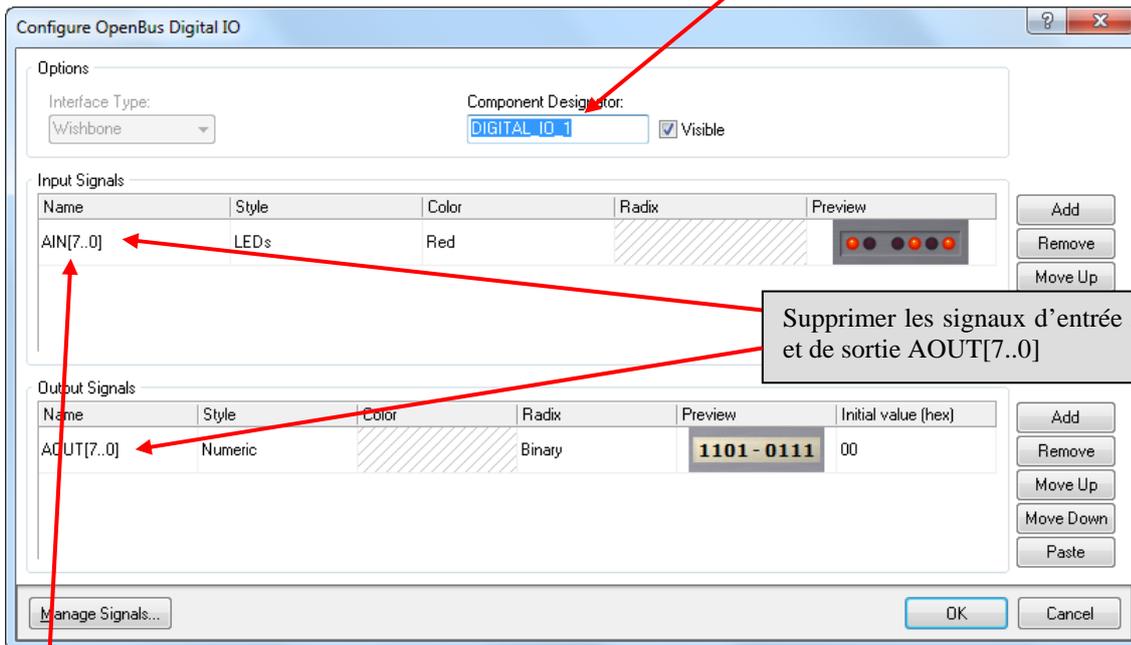


=> Paramétrer et renommer le composant port Digital\_IO\_1

**Clic bouton droit** sur le composant port Digital\_IO\_1, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus Digital IO**

Avant :

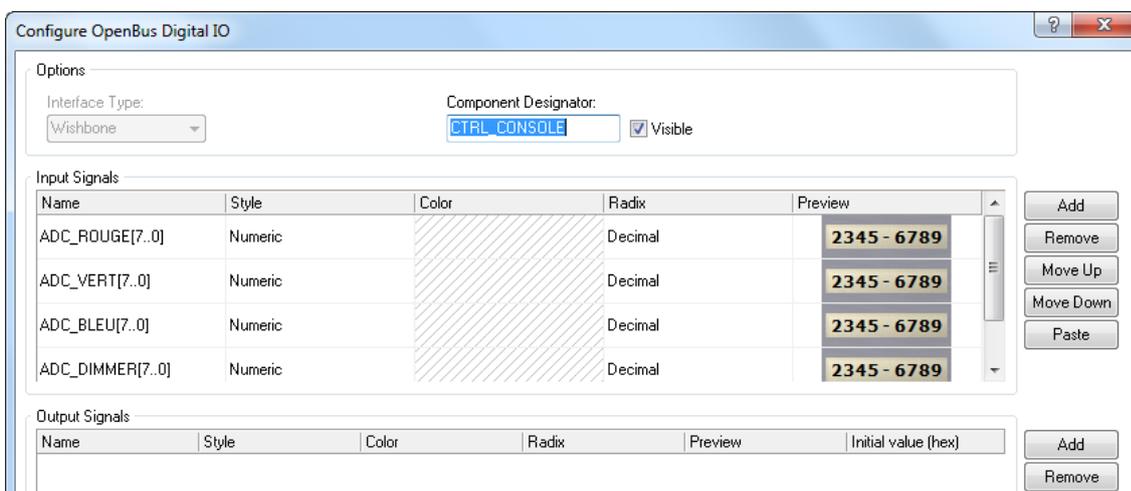
Renommer en CTRL\_CONSOLE



Supprimer les signaux d'entrée AIN[7..0] et de sortie AOUT[7..0]

Ajouter les signaux d'entrée ADC\_ROUGE [7..0], ADC\_VERT[7..0], ADC\_BLEU [7..0], et ADC\_DIMMER[7..0],

Après :

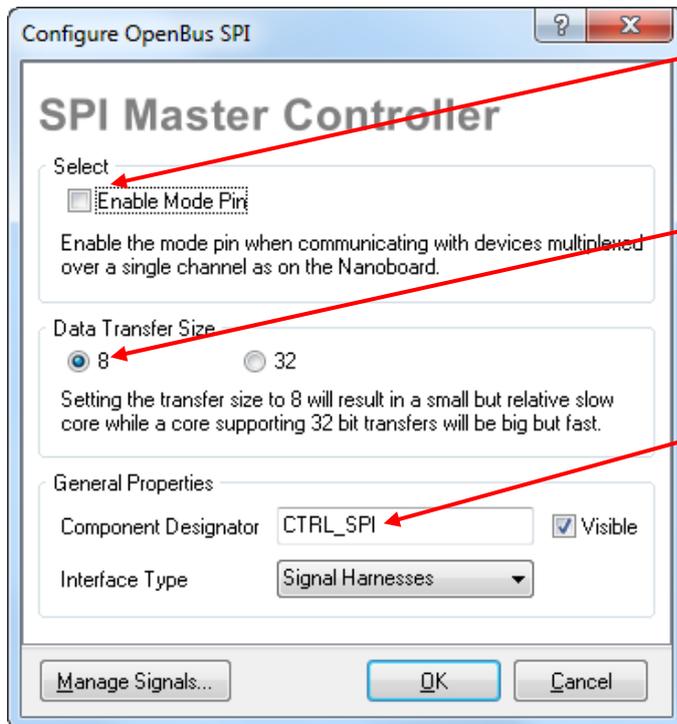


Cliquer sur OK pour valider les changements

=> Paramétrer et renommer le composant WB\_SPI\_1



**Clic bouton droit** sur le composant WB\_SPI\_1, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus SPI**



Décocher Enable Mode Pin

Echange de données sur le bus SPI en 8 bits avec le convertisseur ADC084

Renommer en CTRL\_SPI

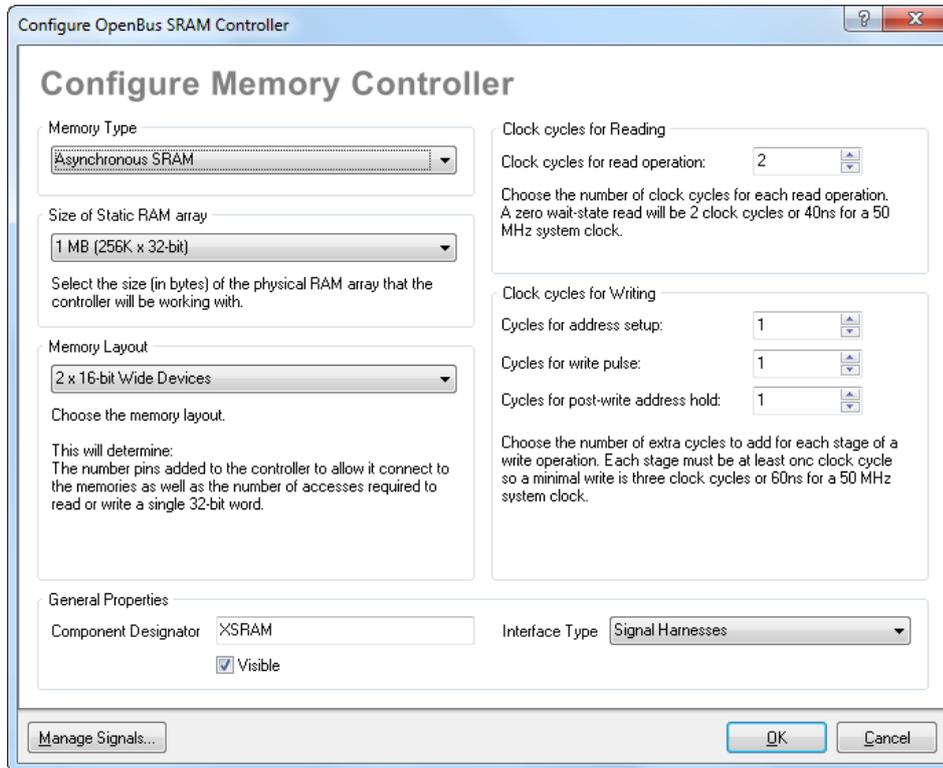
Cliquer sur OK pour valider les changements

XSRAM



=> Paramétrer le composant Contrôleur SRAM

**Clic bouton droit** sur le composant XSRAM, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus SRAM Controller**

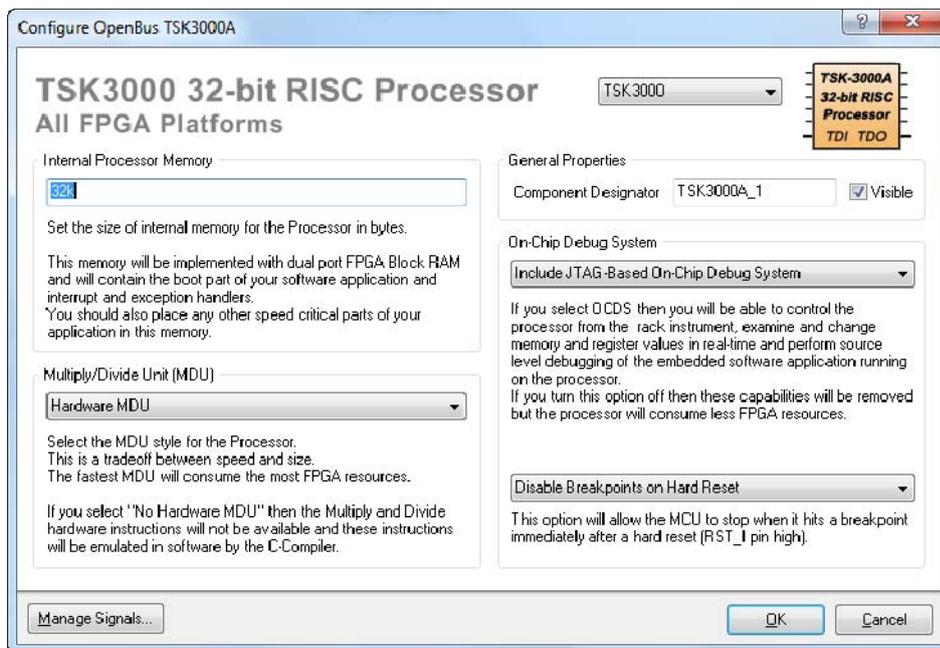


Cliquer sur OK pour valider les changements

=> Paramétrer le processeur comme ci-dessous



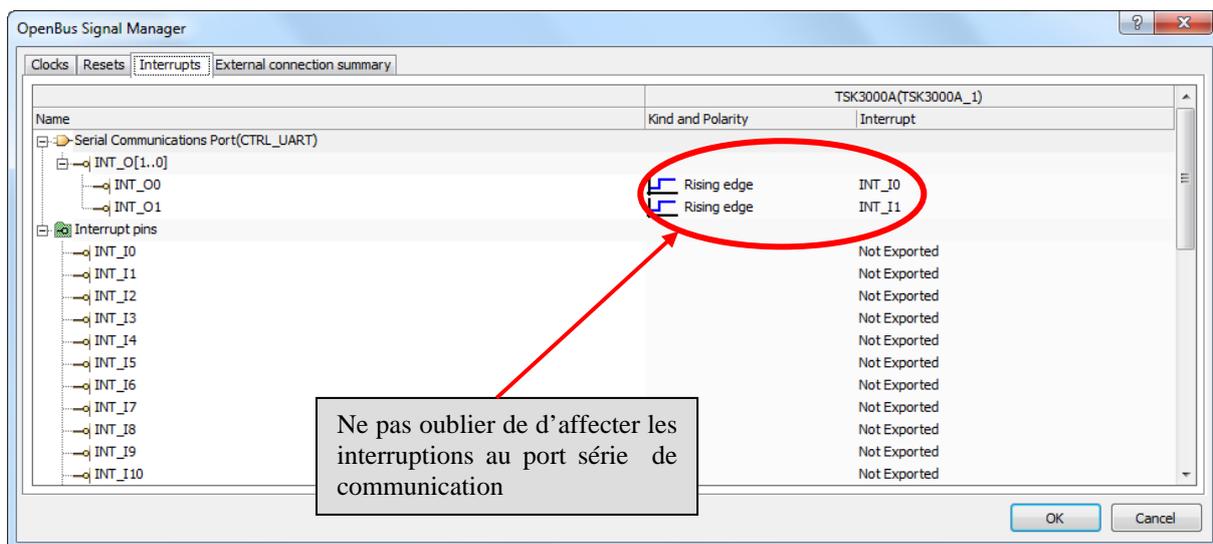
**Clic bouton droit** sur le composant port Processeur **TSK3000A\_1**, puis dans l'onglet qui apparaît sélectionner **Configure TSK3000A\_1**



Cliquer sur OK pour valider les changements

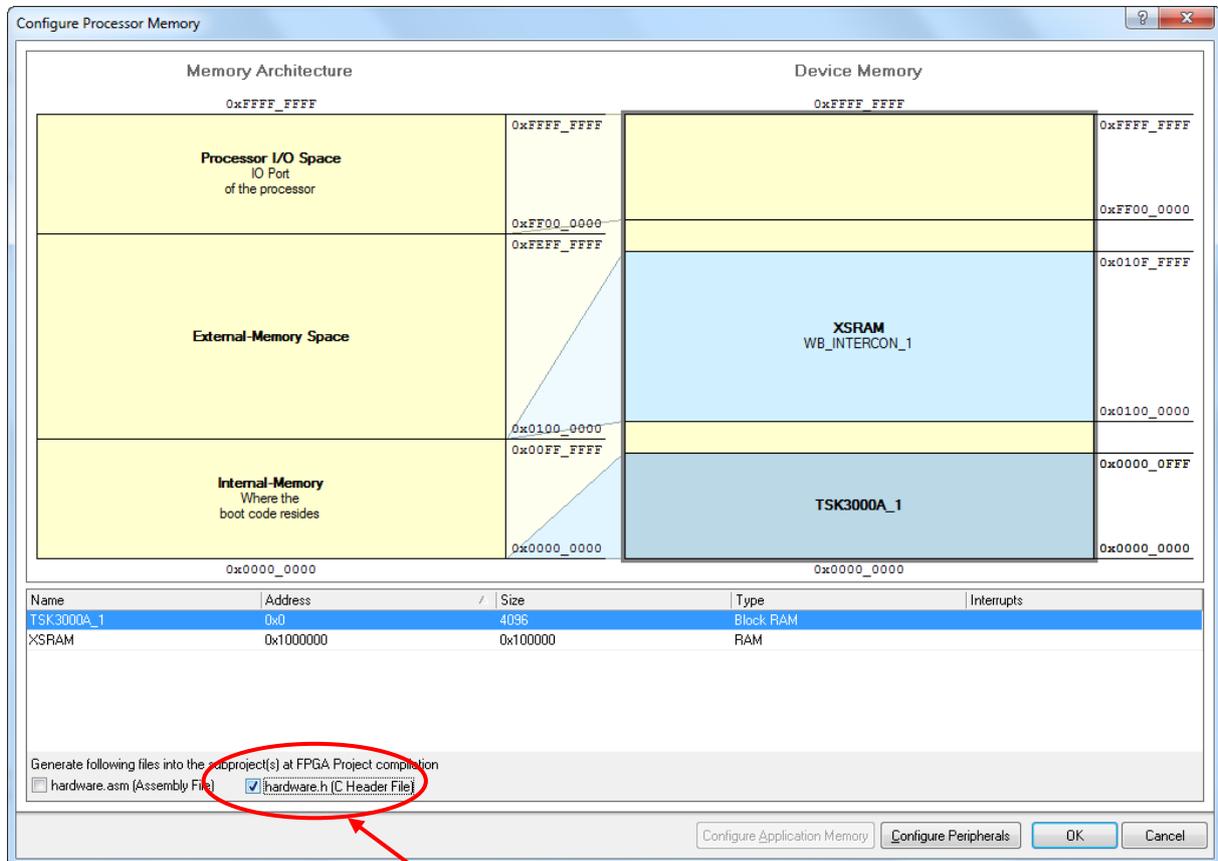
**Clic bouton gauche** sur le **Manage Signal**, puis dans l'onglet qui apparaît sélectionner **Interrupts**

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



=> 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**

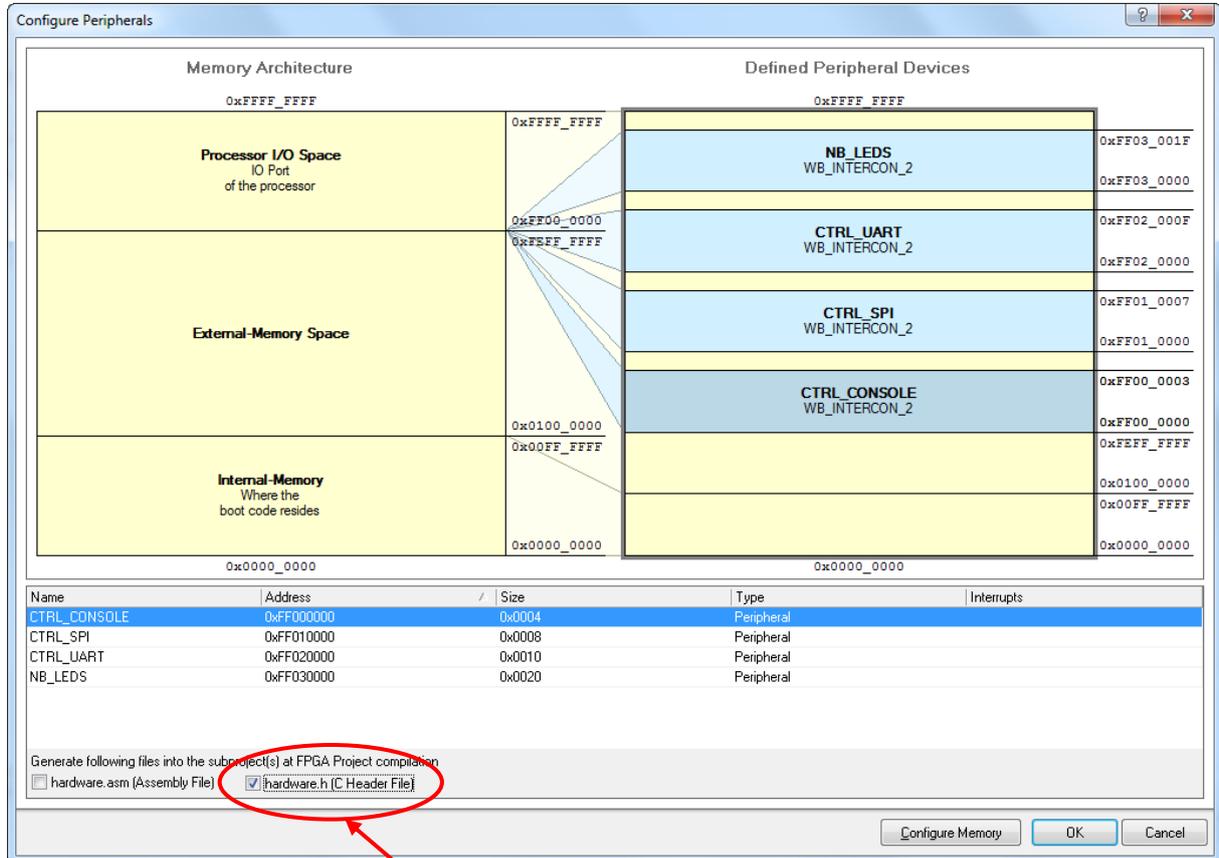


Ne pas oublier de cocher

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 apparait sélectionner **Configure Processor Peripherals**



Ne pas oublier de cocher

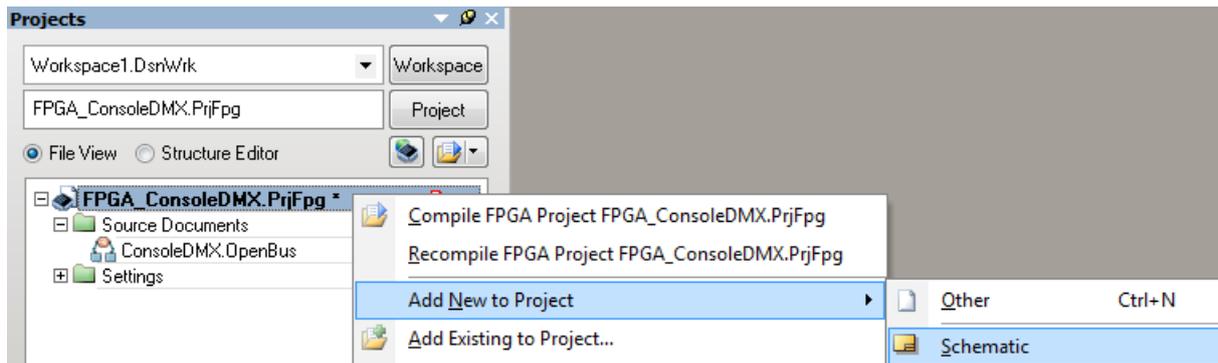
Cliquer sur OK pour valider les changements

### 3 Dessin du TOP schéma

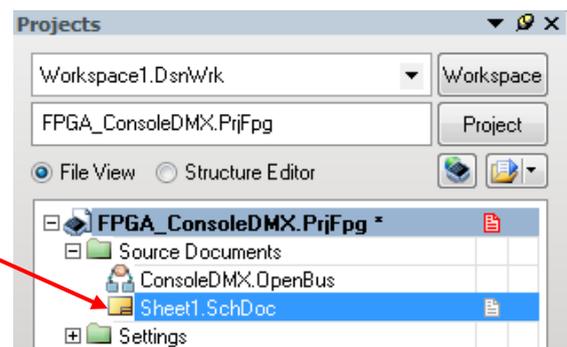
#### 3.1 Création du fichier schéma en tête du projet FPGA

=> Ajouter un nouveau schéma :

**Clic droit** sur le nom du projet FPGA dans l'onglet Projets et choisir la commande **Add New to Project >> Schematic**.



**Remarque :** à ce stade le schéma Sheet1.SchDoc est hiérarchiquement sous le fichier OPEN\_BUS.



=> Renommer le nouveau schéma :

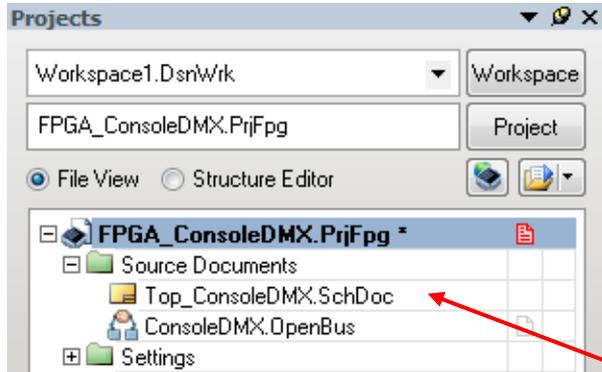
**Clic bouton droit** sur le nom du nouveau schéma (*Sheet1.SchDoc*) dans l'onglet Projets et choisir la commande sauvegarder le document **Save As** avec le nom *Top\_ConsoleDMX.SchDoc* dans le répertoire de travail « **\TP72\_ConsoleDMX** ».

=> Sauvegarder le projet :

**Clic droit** sur le nom du projet FPGA (*FPGA\_ConsoleDMX.PrjFpg*) dans l'onglet Projets et choisir la commande **Save Project**

=> Recompiler le projet :

**Clic droit** sur le nom du projet FPGA (*FPGA\_ConsoleDMX.PrjFpg*) dans l'onglet Projets et choisir la commande **Compile FPGA Projet FPGA\_ConsoleDMX.PrjFpg**



**Remarque :** suite à la compilation du projet FPGA, le schéma est remplacé en tête du projet

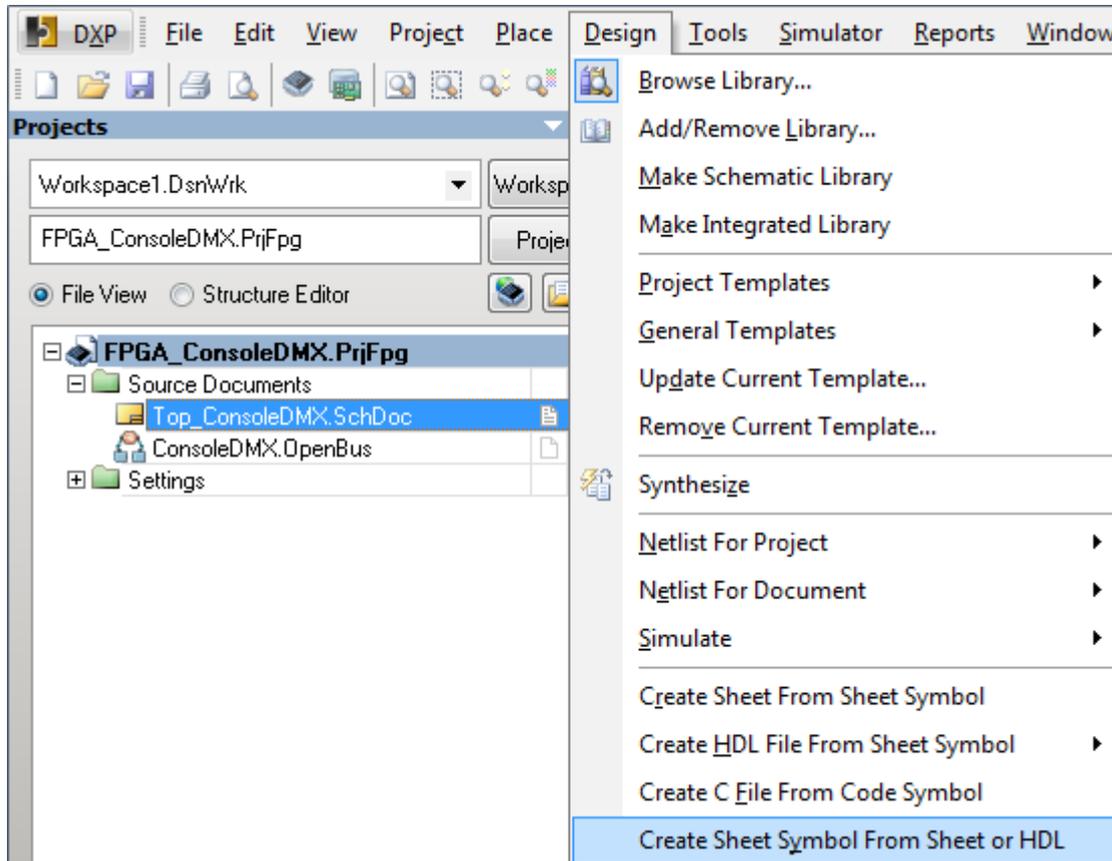
=> Dans la zone message doit apparaître le résultat de la compilation :

Messages			
Class	Document	Source	Message
[Info]	FPGA_ConsoleDMX.PrjFpg	Compiler	Compile successful, no errors found.

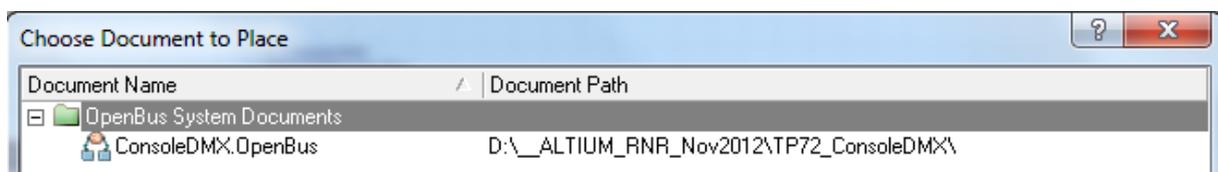
Si la compilation n'est pas réussie corriger vos erreurs.

### 3.2 Placer dans le schéma le symbole créé à partir du fichier OPEN BUS

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

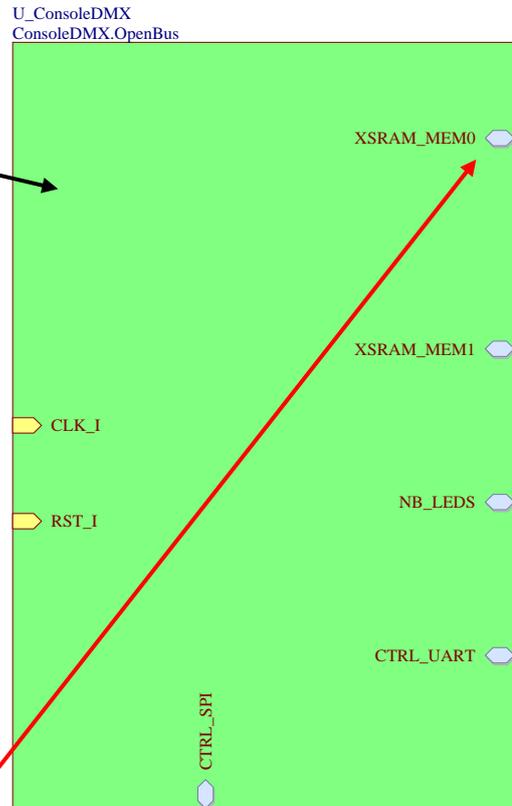


=> Cliquer sur le nom du fichier « *ConsoleDMX.OpenBus* »



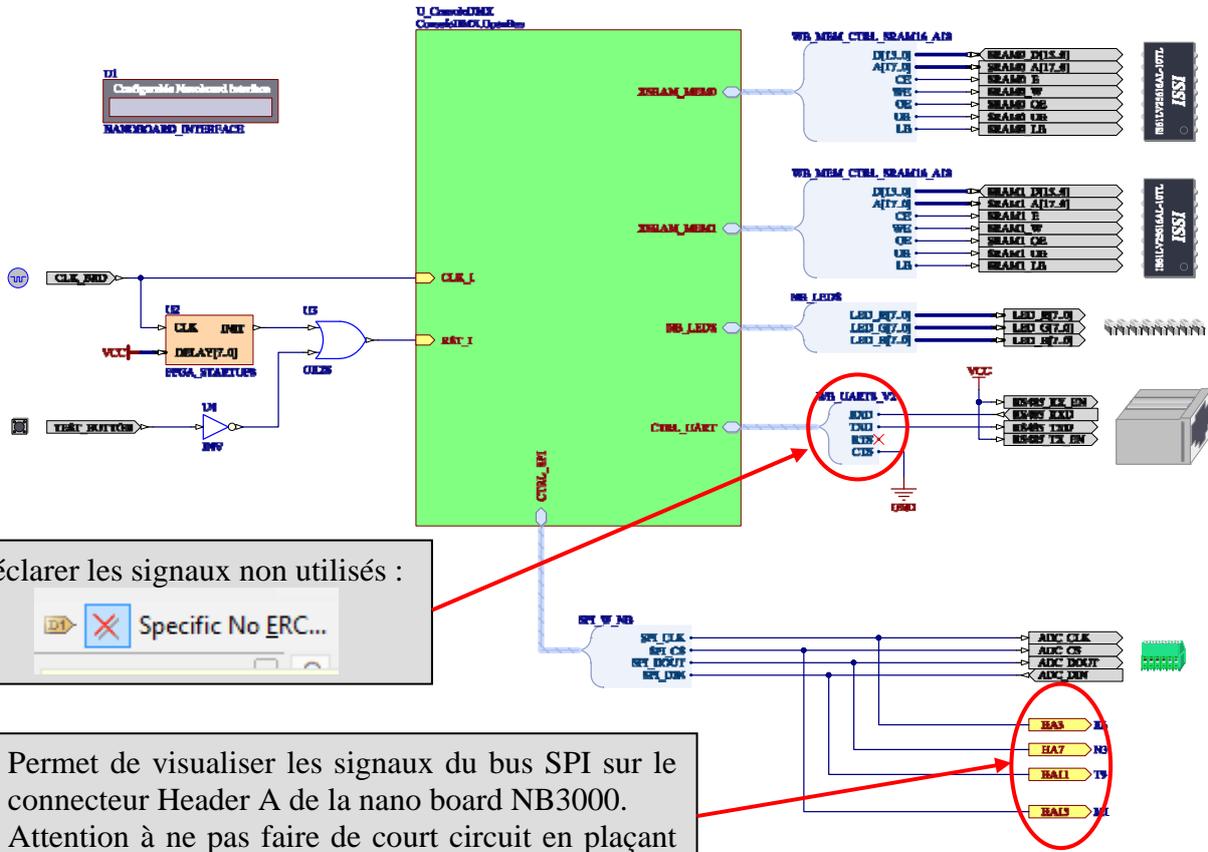
=> Organiser le corps du symbole comme ci-dessous en déplaçant les ports bleu et jaune :

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



Pour relier le composant mémoire au symbole, clic droit sur X\_SRAM\_MEN0 et sélectionner **Sheet entry Actions >> Place Harness connector of Type**

Schéma TOP complet à obtenir :



Déclarer les signaux non utilisés :

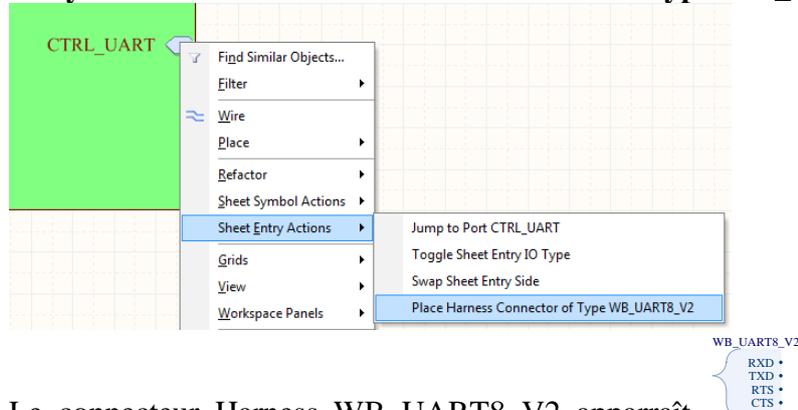
Permet de visualiser les signaux du bus SPI sur le connecteur Header A de la nano board NB3000. Attention à ne pas faire de court circuit en plaçant les sondes d'oscilloscopes

**Remarque :**

Le port de couleur jaune permet la connexion d'un fil.  
 Le port de couleur bleu permet la connexion d'un bus vers un connecteur Harness.

=> Placer un connecteur Harness pour le bus du port série CTRL\_UART:

**Clic droit** sur le port de couleur bleu « CTRL\_UART », puis choisir la commande **Sheet entry Actions >> Place Harness connecteur of Type WB\_UART\_V2**



Le connecteur Harness WB\_UART8\_V2 apparaîtra sur le schéma, vous pouvez le placer sur le

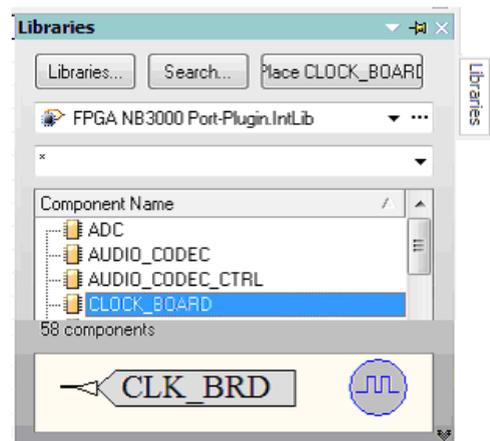
### 3.3 Placer et paramétrer les composants sur le schéma :

⇒ Placer dans le schéma les éléments suivants :

Description	Nom de la fonction	bibliothèque
Test / Reset Button	TEST_BUTTON	FPGA NB3000 Port-Plugin.IntLib
Commande du Barre-graphe 8 LEDs 3 couleurs	LEDS_RGB	FPGA NB3000 Port-Plugin.IntLib
Entrée de l'horloge paramétrable	CLOCK_BOARD	FPGA NB3000 Port-Plugin.IntLib
Mémoire Statique SRAM 0	SRAM0	FPGA NB3000 Port-Plugin.IntLib
Mémoire Statique SRAM 1	SRAM1	FPGA NB3000 Port-Plugin.IntLib
Interface matérielle RS 485	RS485CNTR	FPGA NB3000 Port-Plugin.IntLib
Convertisseur ADC	ADC	FPGA NB3000 Port-Plugin.IntLib
Porte logique	OR2S	FPGA Generic.IntLib
Compteur générique	FPGA_STARTUP8	FPGA Generic.IntLib
Inverseur générique	INV	FPGA Generic.IntLib
Interface Nanoboard	NANOBOARD_INTERFACE	FPGA Instruments.IntLib

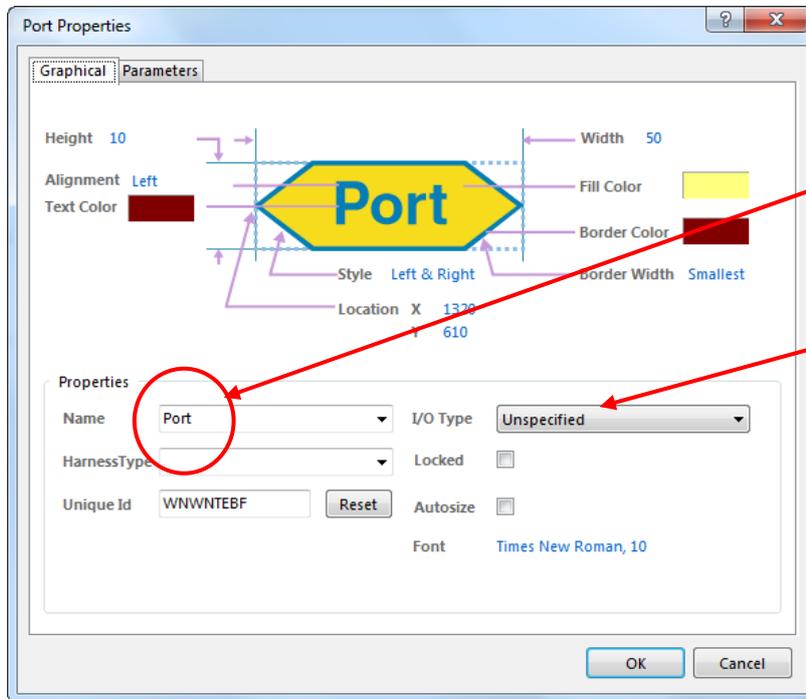
#### Pour placer un nouveau composant :

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



### 3.4 Placer et paramétrer quatre ports en sortie :

Dans la barre de menu choisir la commande **Place >> Port**

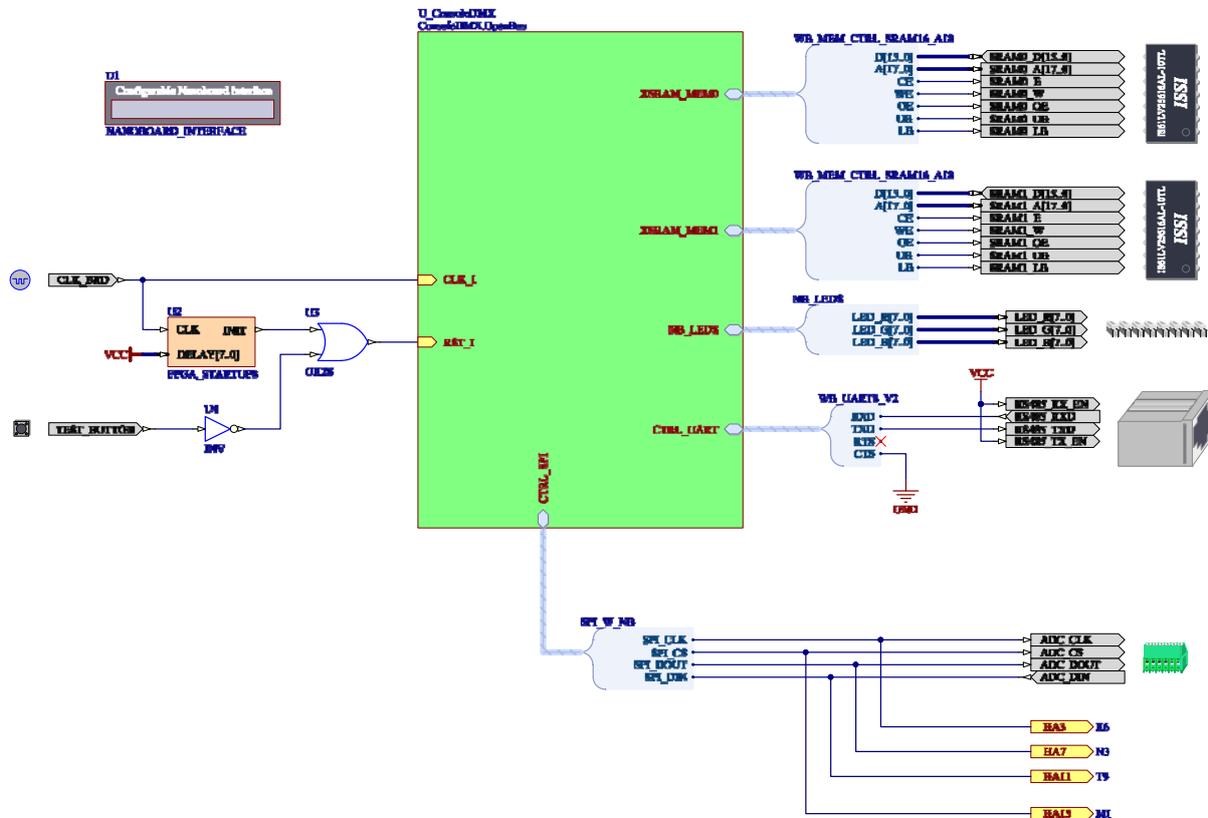


Donner un nom au port : HA3  
Ce port va permet de visualiser un des signaux du bus SPI

Déclarer le port en sortie

Répéter cette commande pour les autres ports : HA7, HA11 et HA15

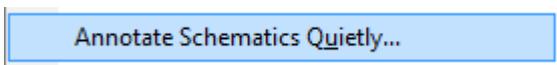
3.5 Compléter le schéma comme ci-dessous :



⇒ Relier les fils entre eux

3.6 Numérotation des composants :

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



**A ce stade du projet :**

Nous pourrions, après ajout des fichiers contraintes, compiler, synthétiser, construire, et programmer le FPGA, puis lancer le programme.  
 Il serait à même d'être exécuté sur la Nanoboard 3000.  
 Nous n'aurions toutefois pas d'outils nous permettant de contrôler la validité de notre programme et d'analyser le fonctionnement du FPGA.  
 Aussi avant de programmer le FPGA nous allons rajouter à notre projet **des instruments de mesures virtuels**.

## **4 Définir les fichiers de contraintes.**

Les fichiers 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.

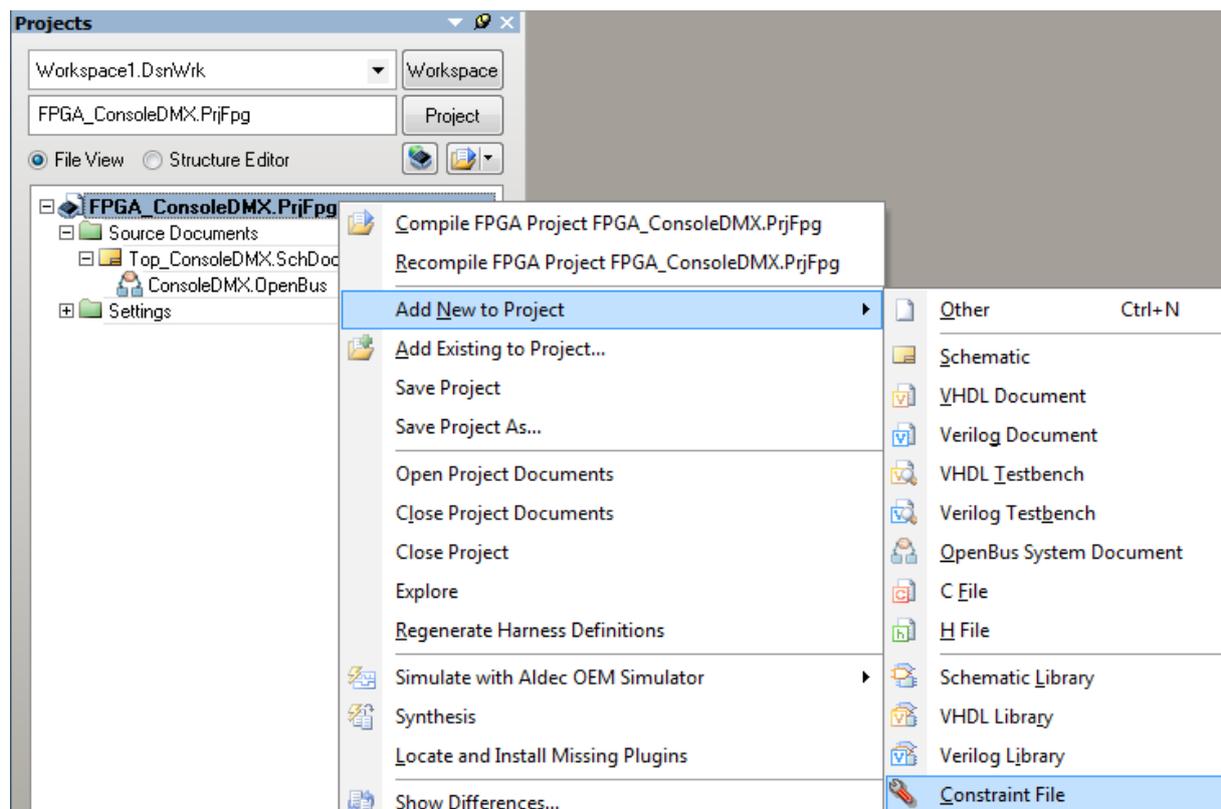
Autres rôles des fichiers contraintes:

- ⇒ paramétrer les broches spécifiques telles 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 contraintes des horloges

⇒ Ajouter un nouveau fichier contrainte au projet :

**Clic bouton droit** sur le nom du projet FPGA dans l'onglet Projets et choisir la commande **Add New to Project >> Contrainte File**



⇒ Il apparaît un fichier texte contrainte à compléter :

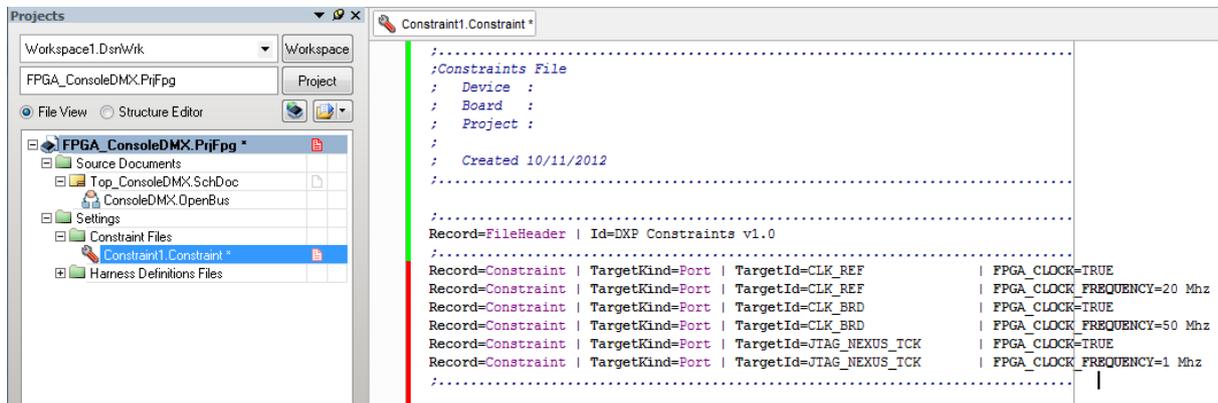


⇒ Pour compléter ce fichier contrainte nous allons copier les lignes suivantes dans le fichier de contrainte.

```
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
/...../
```

**Clic bouton droit** sur le nom du fichier contrainte (*Constraint1.Constraint*) dans l'onglet Projets et choisir la commande sauvegarder le document **Save** dans le répertoire de travail « \TP72\_ConsoleDMX ».

Après :

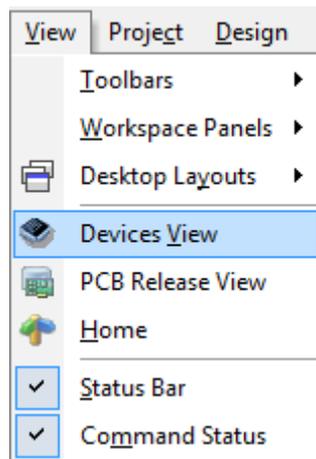


4.2 Le fichier contrainte liant les broches du FPGA aux périphériques implantés sur la Nanoboard 3000 :

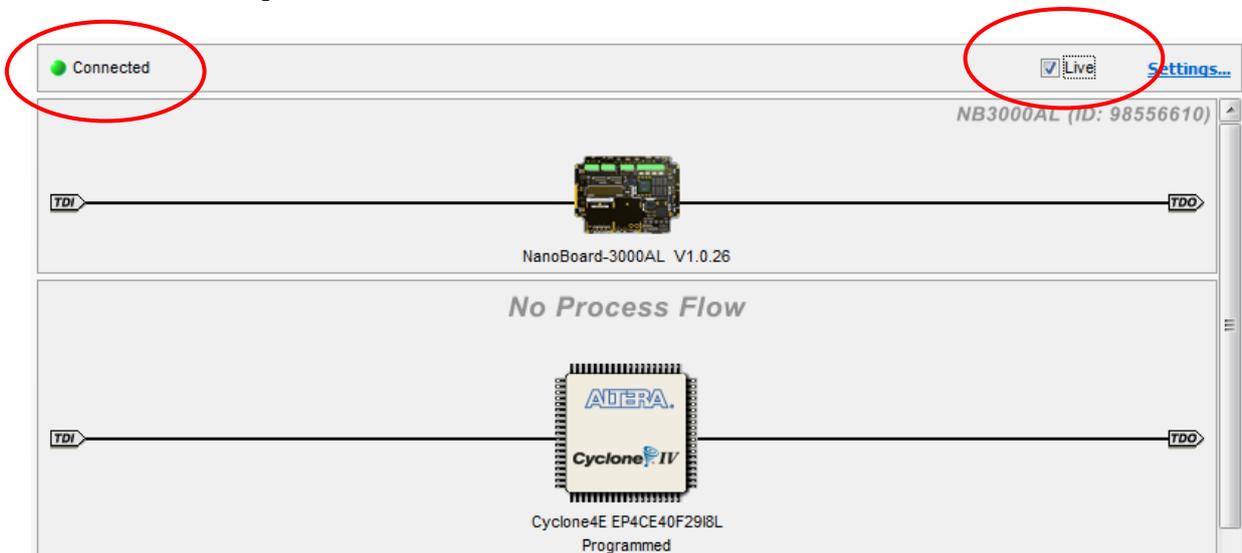
**Ce fichier contrainte existe déjà ! Inutile de l'écrire.**

Pour adjoindre ce fichier contrainte à votre projet vous devez d'abord connecter votre PC à la **Nanoboard**.

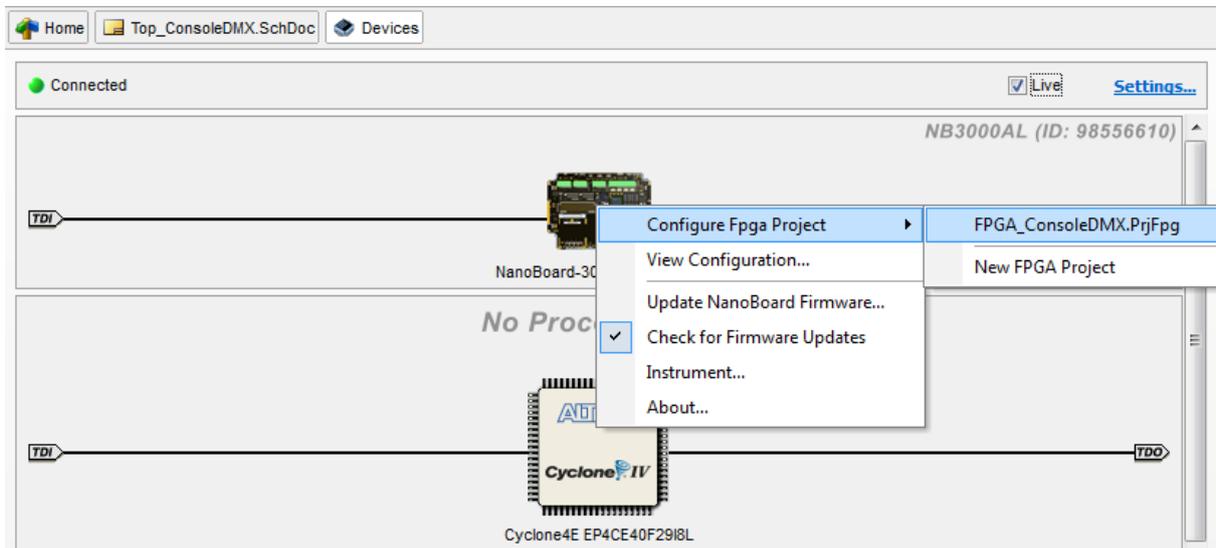
⇒ Faites apparaître la fenêtre de visualisation des composants :



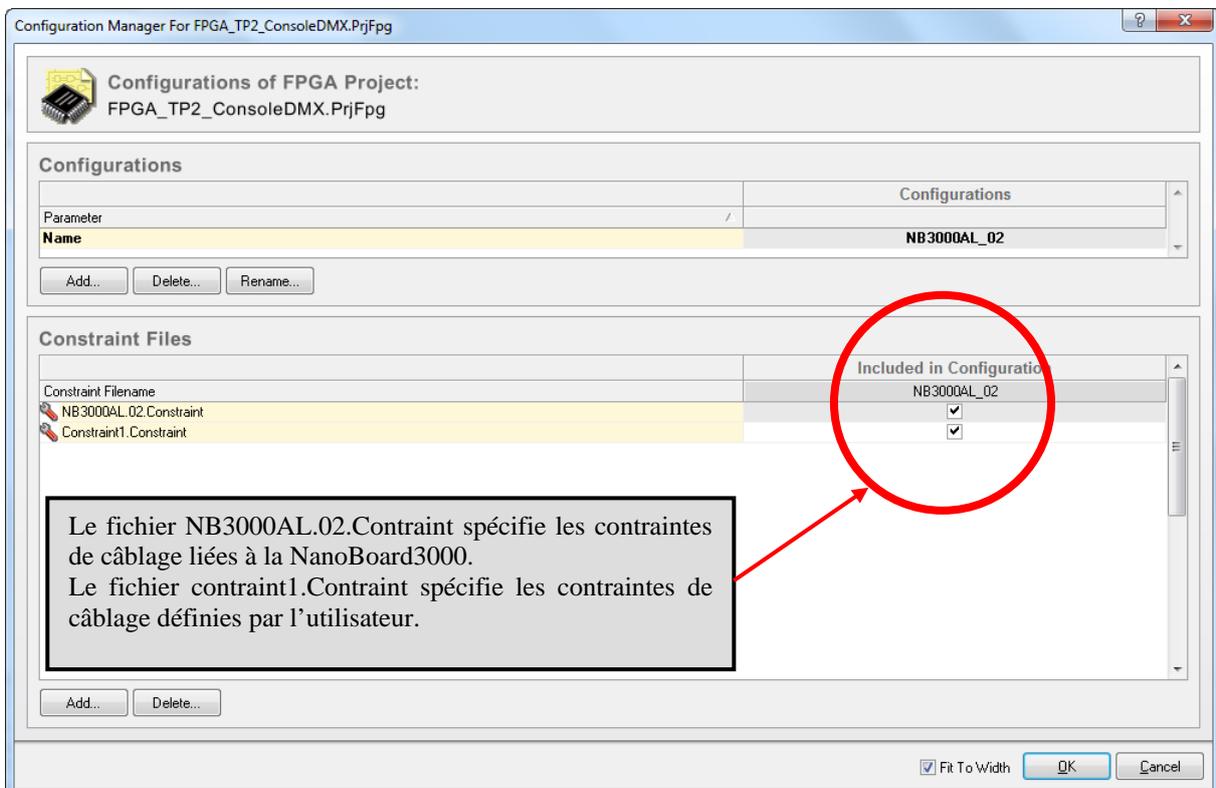
⇒ Assurez-vous que la Nanoboard est bien connectée à votre PC :



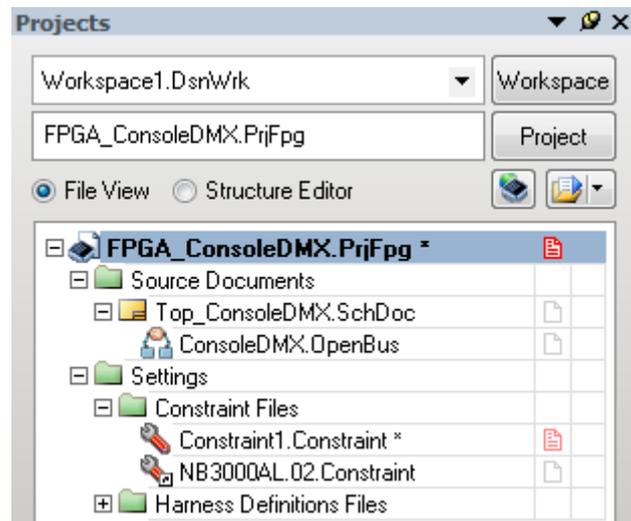
⇒ **Clic bouton droit** sur l'icône de la Nanoboard et configurer le projet FPPGA



⇒ La fenêtre ci-dessous vous invite à spécifier les fichiers contraintes que vous voulez utiliser pour votre prochaine phase de **Compilation / Synthèse / Programmation**.



⇒ A l'issue de cette étape un deuxième fichier contrainte « **NB3000AL.02.Constraint** » est lié à votre projet :



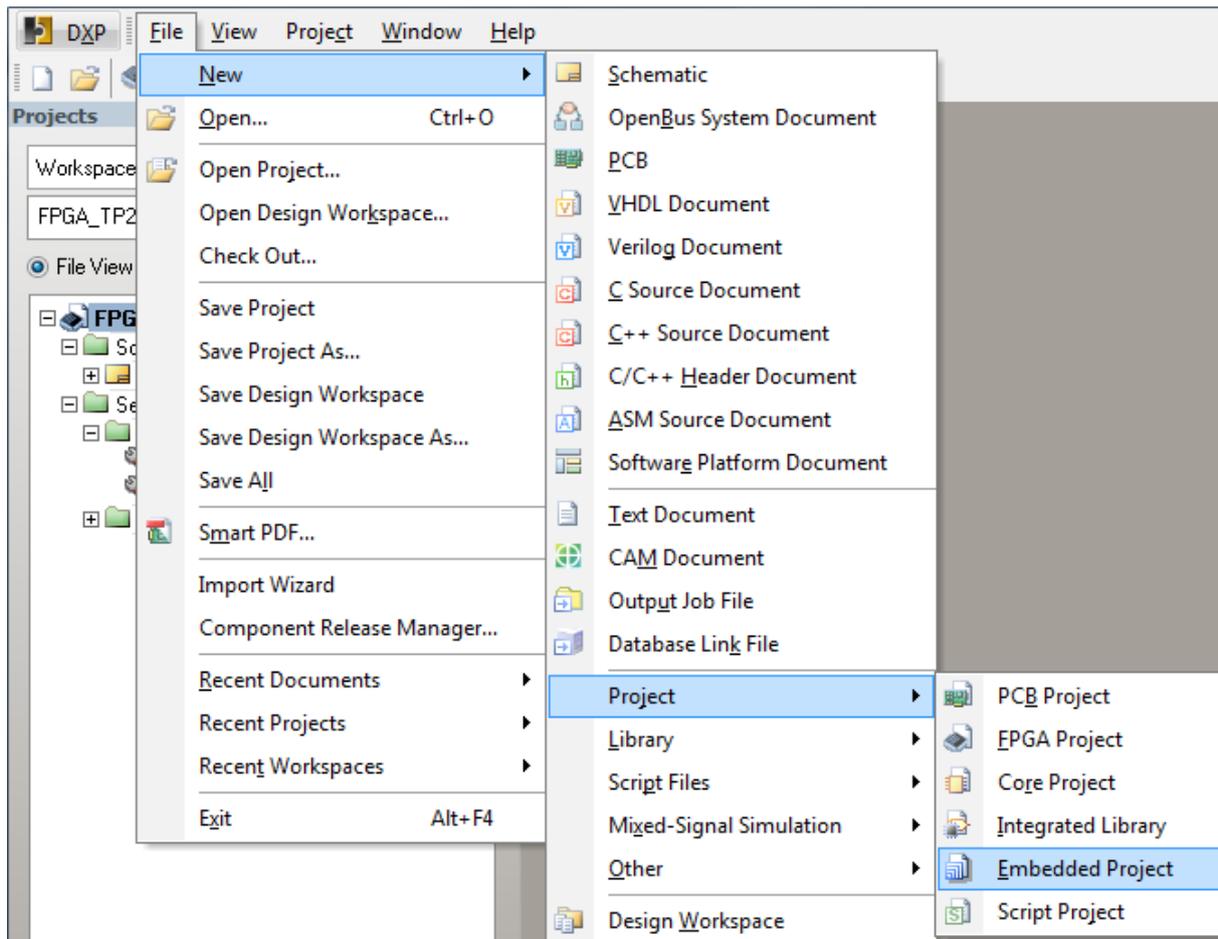


## 5 Création du projet embarqué

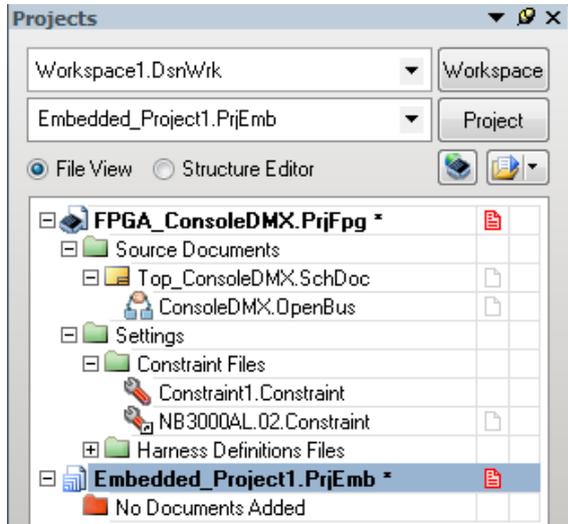
### Programmation du code C : SOFTWARE DESIGN FLOW

#### 5.1 Ajouter un projet embarqué à votre environnement

Créer un nouveau projet en utilisant la commande : **File >> New >> Embedded Project Project.**



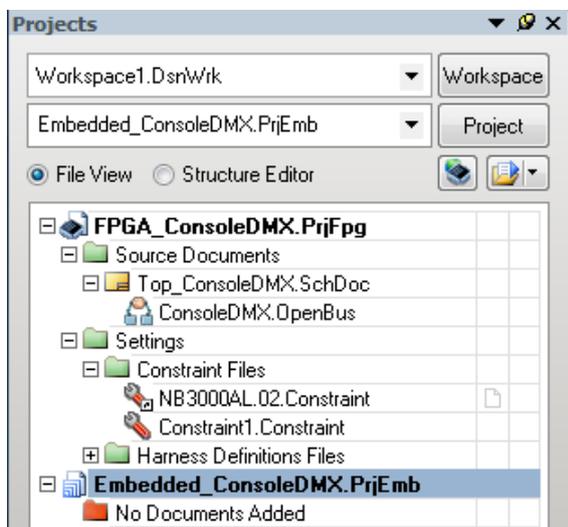
Un projet nommé (*Embedded\_Projet1.PrjEmb*) apparaît dans l'onglet gestion de projet.



Créer un dossier « Embedded » dans le répertoire de travail « **\TP72\_ConsoleDMX** ».

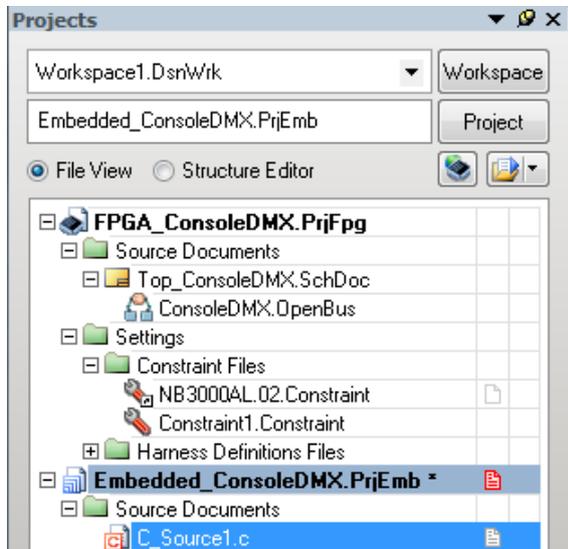
**Clic bouton droit** sur le nom du nouveau projet (*Embedded\_Projet1.PrjEmb*) dans l'onglet Projects et choisir la commande **Save Project as « Embedded\_ConsoleDMX.PrjFpg »** pour sauvegarder le projet dans le répertoire de « **\TP72\_ConsoleDMX\Embedded** ».

Après :

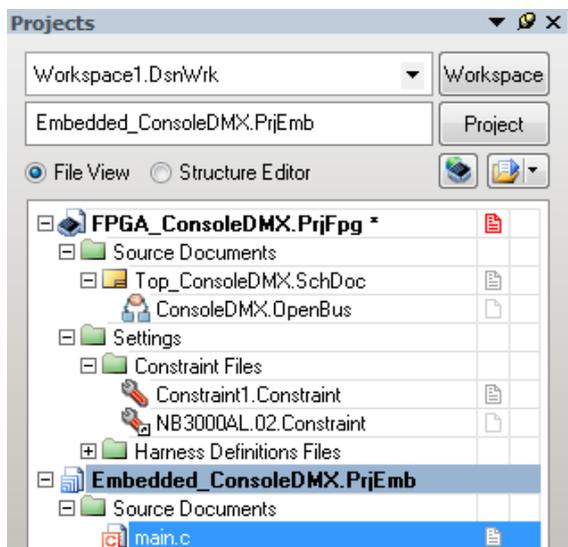


## 5.2 A ce projet joignez les fonctions C qui seront exécutées par le processeur TSK 3000

**Clic bouton droit** sur le nom du nouveau projet (*Embedded\_TP2\_ConsoleDMX.PrjEmb*) dans l'onglet Projets et choisir la commande **Add New to Project >> C File**

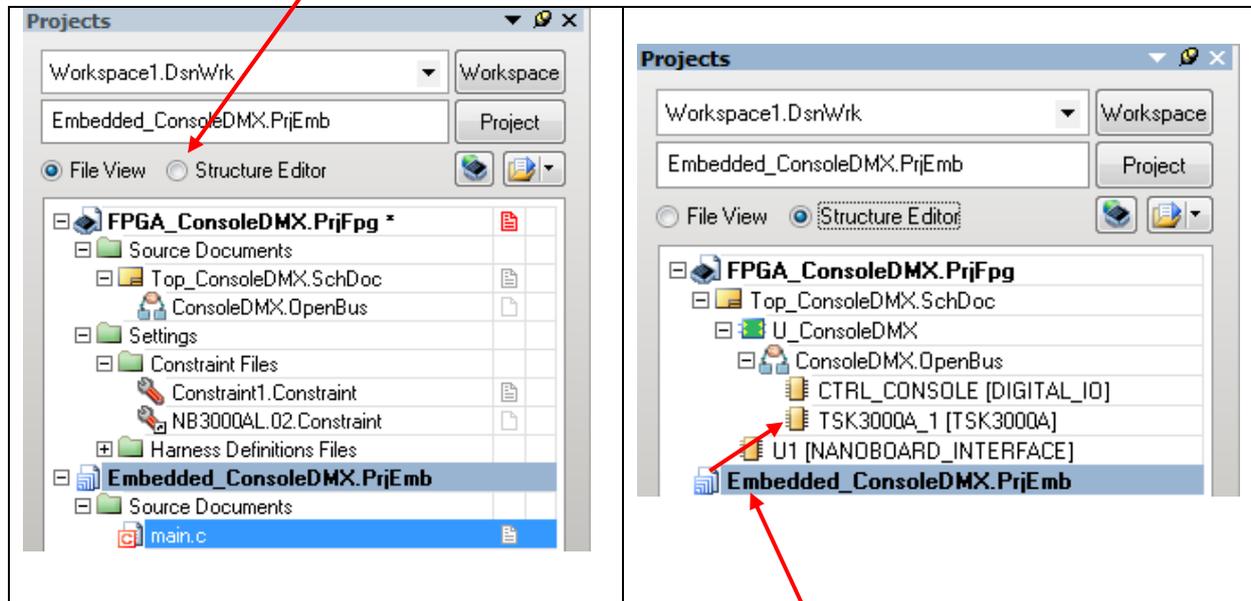


**Clic bouton droit** sur le nom du nouveau fichier (*C\_Source1.c*) dans l'onglet Projets et choisir la commande **Save As >> « main.c »** pour sauvegarder le fichier dans le répertoire de travail dans le sous dossier **Embedded** que vous créez à cet effet sous votre projet courant.



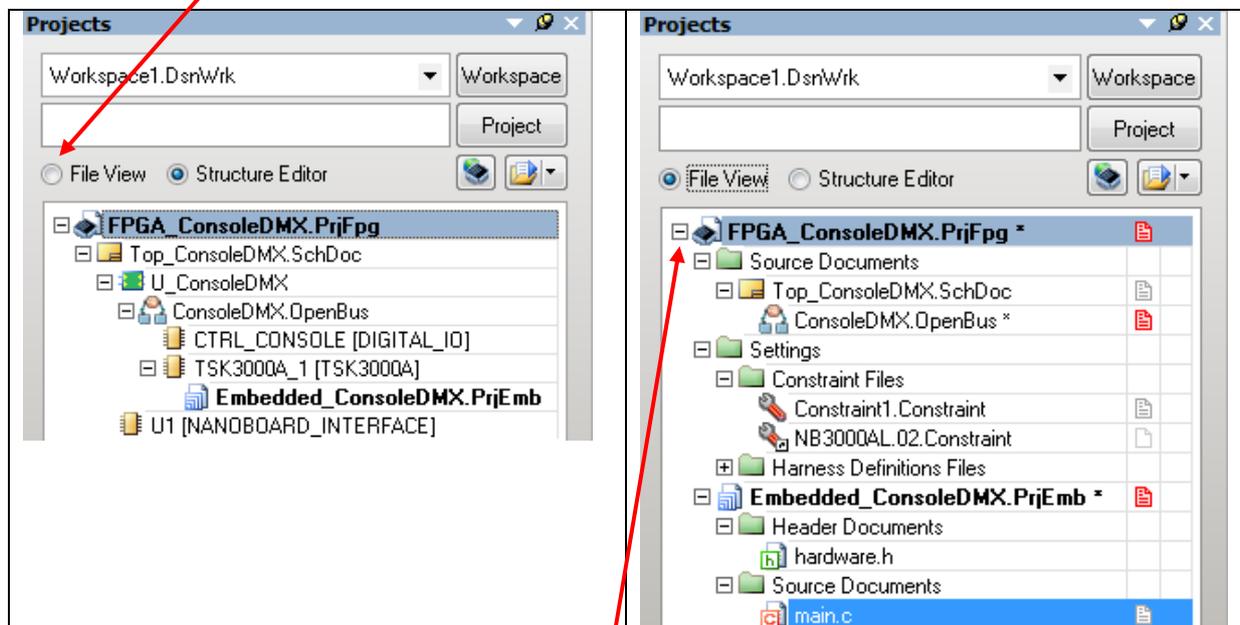
### 5.3 Intégration du projet embarqué sous le projet FPGA

Cliquer sur Structure Editor



Cliquer sur File View

Faites glisser le projet Embedded\_consoleDMX.PrjEmb sous le processeur TSK3000A



Le projet embarqué est maintenant sous le projet FPGA

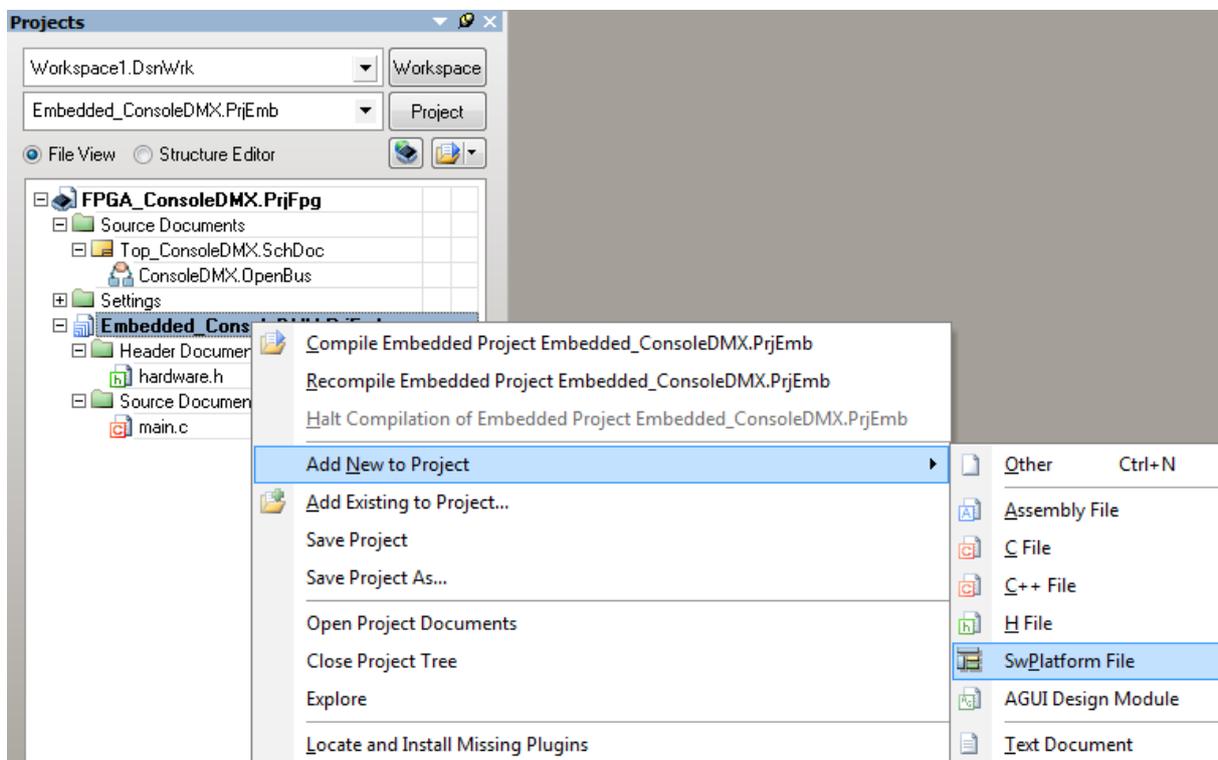
## **6 Construction du fichier « Software Platform » Mise en place des API**

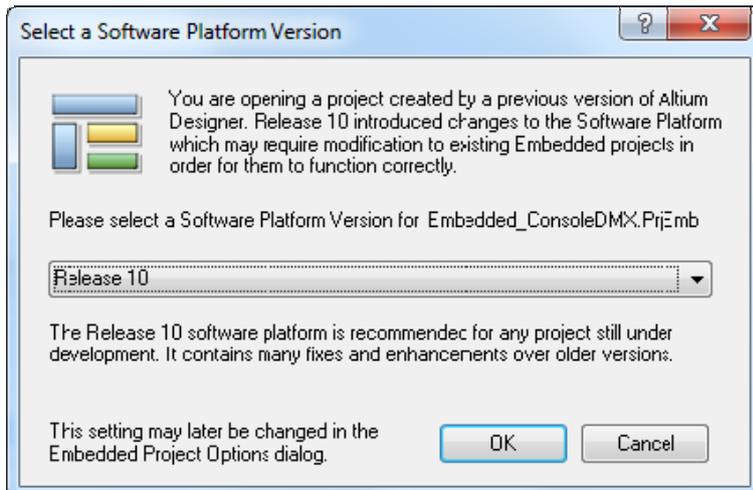
### **API : Application Programming Interface**

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).

*Clic bouton droit* sur le nom du nouveau projet (*Embedded\_ConsoleDMX.PrjEmb*) dans l'onglet Projets et choisir la commande **Add Neww to Projet >> SwPlatform file**





Afin d'effectuer une connexion bas niveau avec les modules décrits dans le fichier OpenBus cliquer sur **IMPORT FROM FPGA**

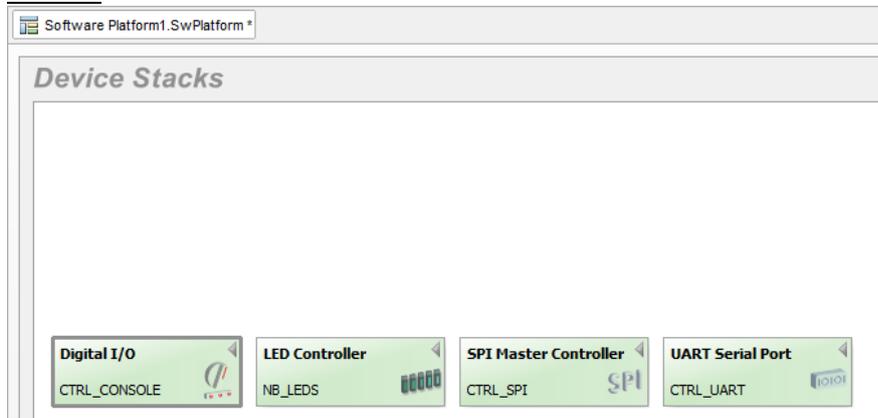


Pour obtenir la couche supérieure du port Digital I/O

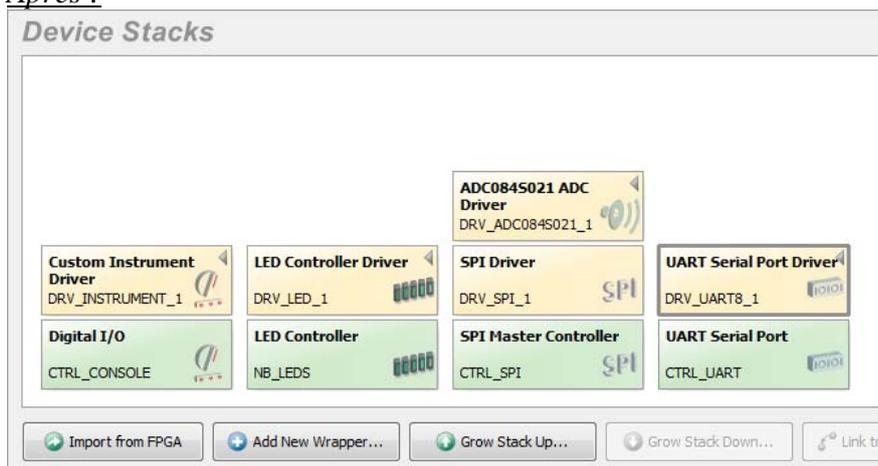
⇒ **Clic bouton droit** sur l'icône **vert Digital I/O** puis dans l'onglet qui s'ouvre et cliquer sur Grow Stack Up

Répéter cette opération sur LED Controller, SPI Master Controller et UART Serial Port.

Avant :



Après :



Ces interfaces donnent accès aux différentes fonctions prédéfinies (API) de pilotage des périphériques.

Paramétrage de l'API SPI Master Controller:

The screenshot shows the 'Device Stacks' window with the following components:

- Custom Instrument Driver (DRV\_INSTRUMENT\_1)
- Digital I/O (CTRL\_CONSOLE)
- LED Controller Driver (DRV\_LED\_1)
- LED Controller (NB\_LEDS)
- ADC0845021 ADC Driver (DRV\_ADC0845021\_1)
- SPI Driver (DRV\_SPI\_1)
- SPI Master Controller (CTRL\_SPI)

An inset window titled 'Cliquer sur l'API SPI Driver pour paramétrer l'API' shows the configuration for the SPI Driver and SPI Master Controller:

<b>SPI Driver</b>	
ID	DRV_SPI_1
SPI Mode	MODE3
SCLK frequency	100000
NanoBoard multip	<input type="checkbox"/>
<b>SPI Master Controller</b>	
ID	CTRL_SPI
Import Settings	<input checked="" type="checkbox"/>
Base Address	0xFF010000
Data Transfer Siz	8
Mode pin	<input type="checkbox"/>

Paramétrage de l'API UART Serial Port :

The screenshot shows the 'Device Stacks' window with the following components:

- Custom Instrument Driver (DRV\_INSTRUMENT\_1)
- Digital I/O (CTRL\_CONSOLE)
- LED Controller Driver (DRV\_LED\_1)
- LED Controller (NB\_LEDS)
- ADC0845021 ADC Driver (DRV\_ADC0845021\_1)
- SPI Driver (DRV\_SPI\_1)
- SPI Master Controller (CTRL\_SPI)
- UART Serial Port Driver (DRV\_UART8\_1)
- UART Serial Port (CTRL\_UART)

An inset window titled 'Cliquer sur l'API UART Serial Port Driver, puis paramétrer l'API' shows the configuration for the UART Serial Port Driver and UART Serial Port:

<b>UART Serial Port Driver</b>	
ID	DRV_UART8_1
Baudrate	250000
Parity	NONE
Databits	8
Stopbits	1
Handshake	NONE
TX Buffer Size	0
TX Blocking	<input checked="" type="checkbox"/>
RX Buffer Size	0
RX Blocking	<input checked="" type="checkbox"/>
TX Interrupt Value	0
RX Interrupt Value	1
<b>UART Serial Port</b>	
ID	CTRL_UART
Import Settings	<input checked="" type="checkbox"/>
Base Address	0xFF020000
TX Interrupt	0
RX Interrupt	1

=> cliquer sur  pour générer les fichiers « swplatform.h » et « swplatform.c »

## 7 Ajout du fichier C principal

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

```

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

/*****
/* Déclaration des équivalences */
*****/
#define MODE_PORT 0 //
#define LED_PORT 1 //

/*****
/* Déclaration des variables */
*****/
unsigned char DMX_Table[512]; // Déclaration d'un tableau de 512 caractères

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

int Initialize(void)
{
    drv_instrument_1 = instrument_open(DRV_INSTRUMENT_1); // Ouverture du périphérique
    virtuel instrument DRV_INTRUMENT
    drv_led_1 = led_open(DRV_LED_1); // Ouverture du périphérique DRV_LED
    led_turn_all_off(drv_led_1); // Extinction des leds

    drv_uart8_1 = uart8_open(DRV_UART8_1); // Ouverture du périphérique port série UART
    // 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); // 250kbauds, pas de
    parité, 8 bits de données et 2 bits de stops.

    drv_adc084s021_1= adc084s021_open(DRV_ADC084S021_1); // Ouverture du port ADC084S021
    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]=adc084s021_read(drv_adc084s021_1,i+1); // 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
        }

        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'émission est vide
            uart8_putchar(drv_uart8_1, DMX_Table[i]); // Envoi du caractère vers l'UART
        }
    }
}
/**** FIN DU PROGRAMME PRINCIPAL ****/

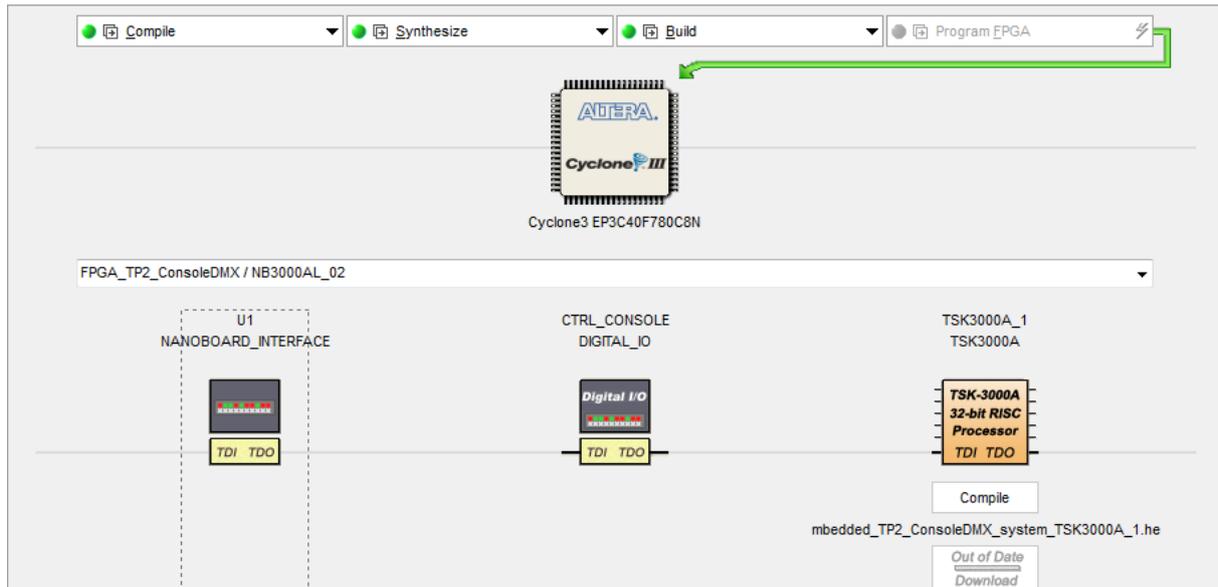
```

## **8 Compiler, Synthétiser, construire, Programmer le FPGA.**

⇒ Cliquer sur Compile, cliquez sur Synthesize, cliquer sur Build.

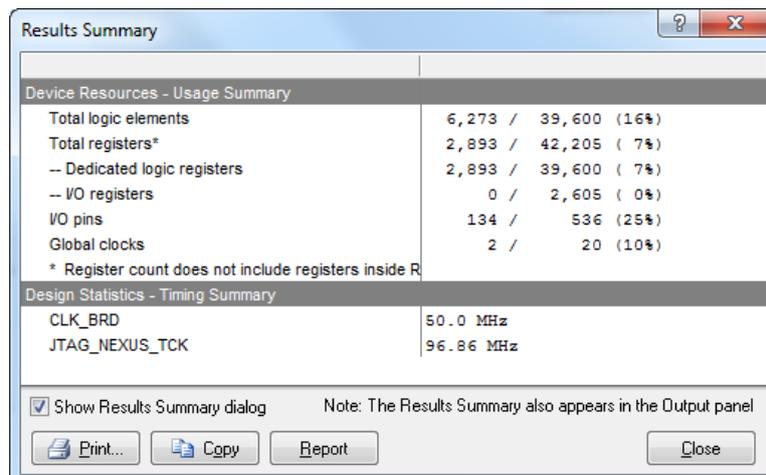
⇒ Si une erreur apparaît vous devez la corriger en modifiant le fichier source identifié à partir du message d'erreur :

Sources d'erreurs possibles :           ⇒ le fichier VHDL  
   ⇒ le schéma TOP  
   ⇒ les fichiers de contraintes



**Remarque :** Nous retrouvons un instrument virtuel mis en œuvre dans le projet : CTRL\_CONSOLE

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



Activité 3 Mini projet DMX-FPGA : **Implanter un processeur dans le FPGA**  
Durée estimée 4h



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

Cliquer sur **Compile** pour compiler le programme C.

Cliquer sur **Up to Date Download** pour télécharger le programme c dans le FPGA.

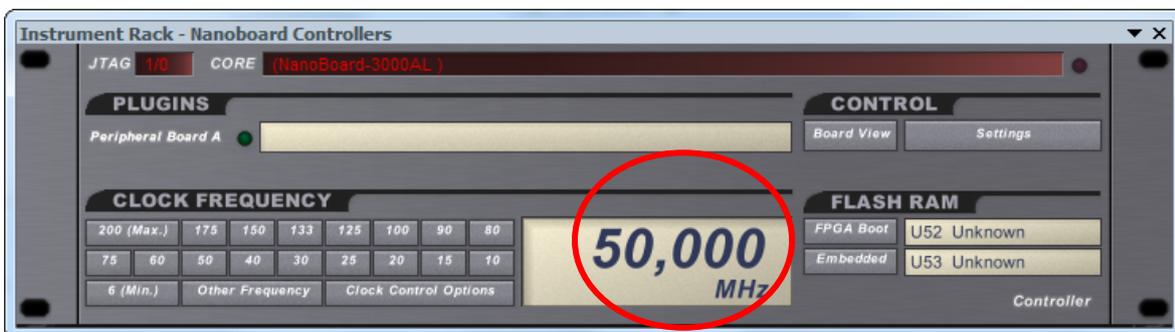
## 9 Mettre en œuvre les instruments de mesures virtuels

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

=> Cliquer sur NanoBoard-3000AL:



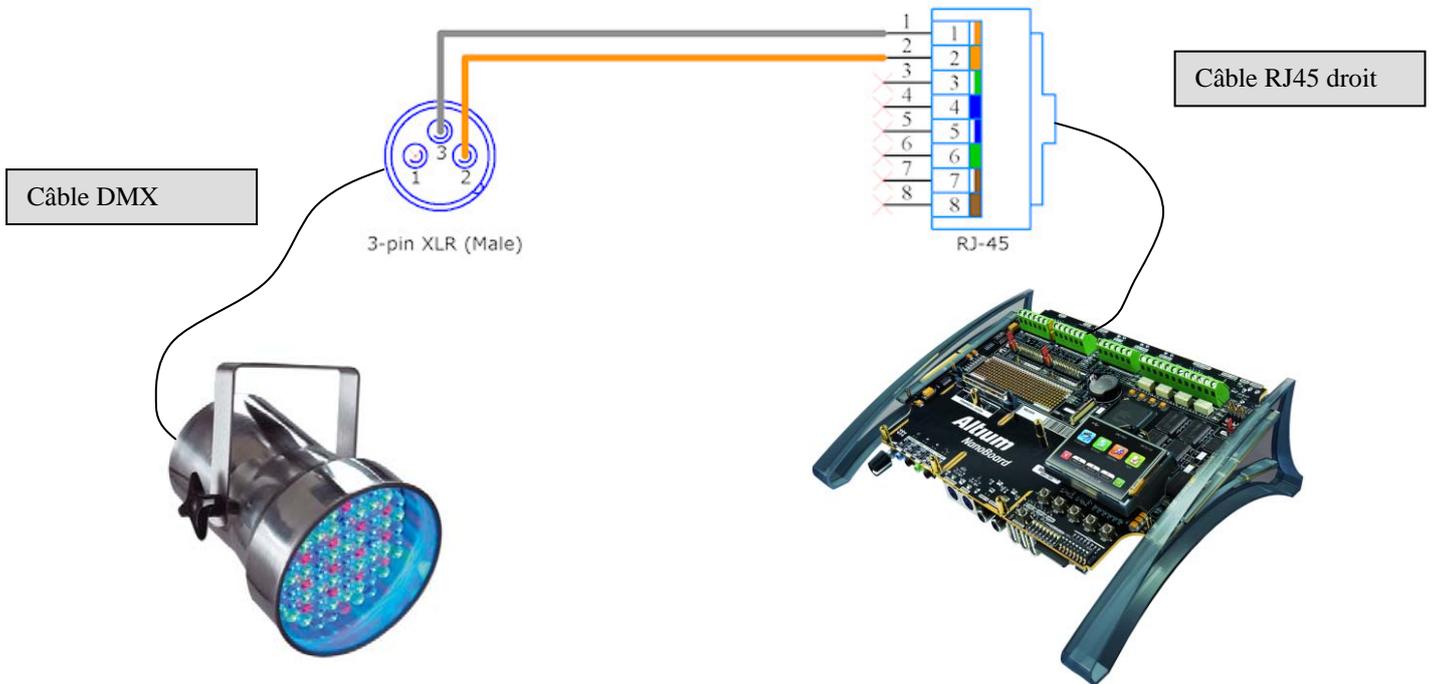
=> Régler la fréquence à 50 MHz.



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.



### 9.3 Injecter une tension continue sur les entrées analogiques de la NB3000

A l'aide d'un pont diviseur réalisé avec des potentiomètres de 50 k ohms, injecter les tensions suivantes sur le connecteur ADC de la NB3000

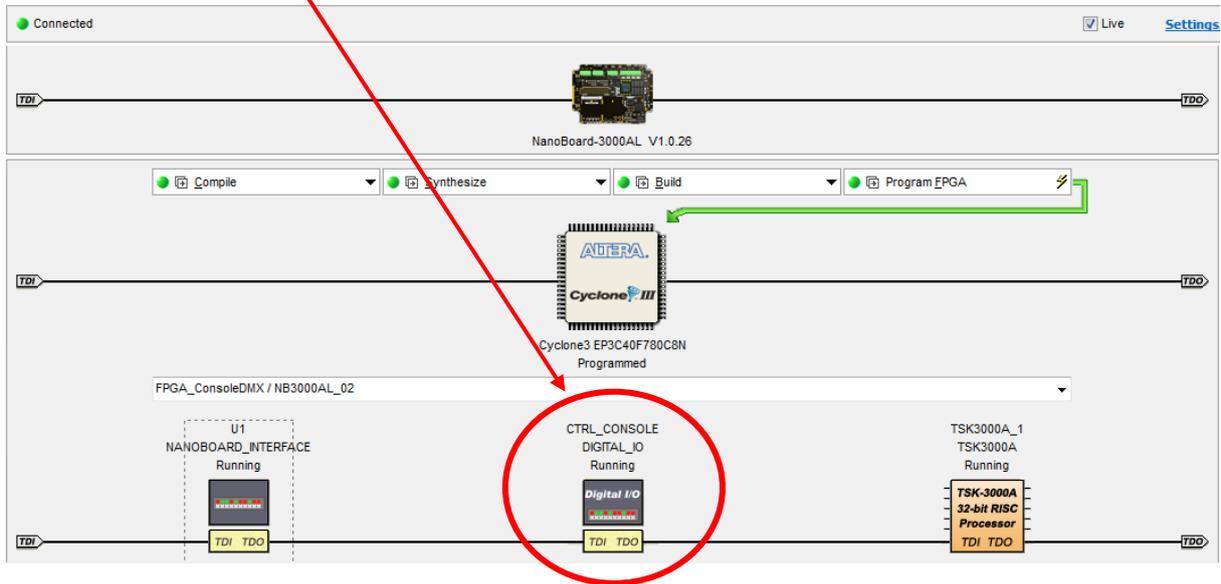


**ATTENTION :**  
 NE PAS DEPASSER LA TENSION DE 3.3 volts A L'ENTREE DU CONVERTISEUR ADC 084S02

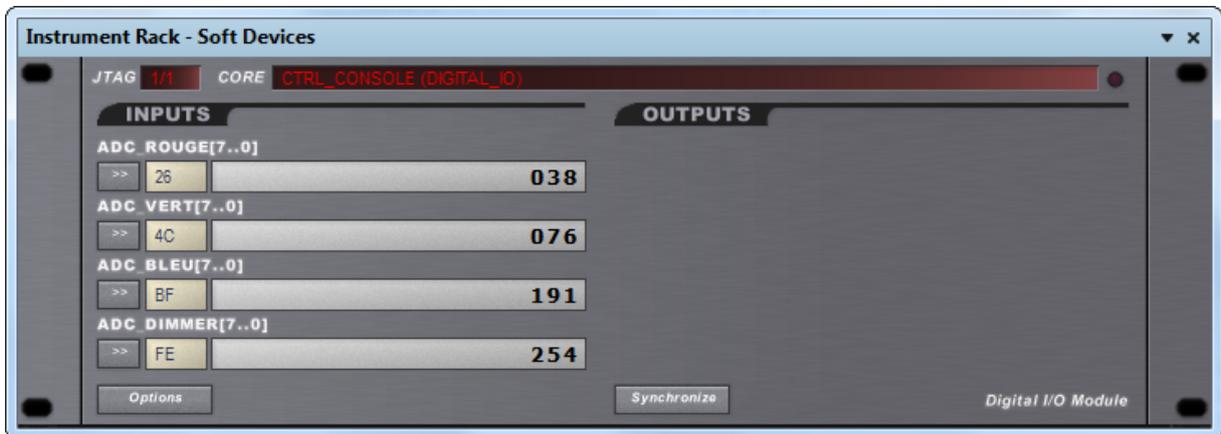
Voie	Tension
ADC0	0.5v
ADC1	1v
ADC2	2.5v
ADC3	3.3v

### 9.4 Affichage du résultats du CAN à l'aide de l'instrument virtuel

=> Cliquer sur le composant Digital\_IO

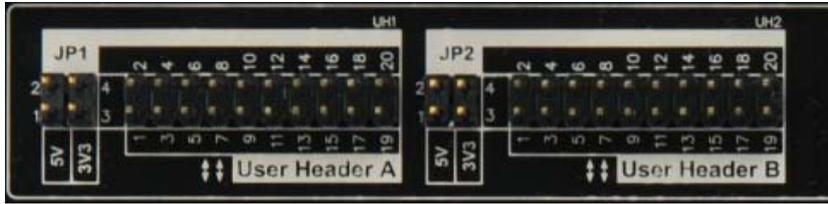


=> Vous pouvez ainsi visualiser les valeurs issues du convertisseur ADC084S21 des 4 premiers octets envoyés sur la liaison DMX



### 9.5 Relevés des chronogrammes sur le connecteur Header A de la NB3000

Effectuer le relevé de la lecture des voies ADC0 ,ADC1 , ADC2 et ADC3

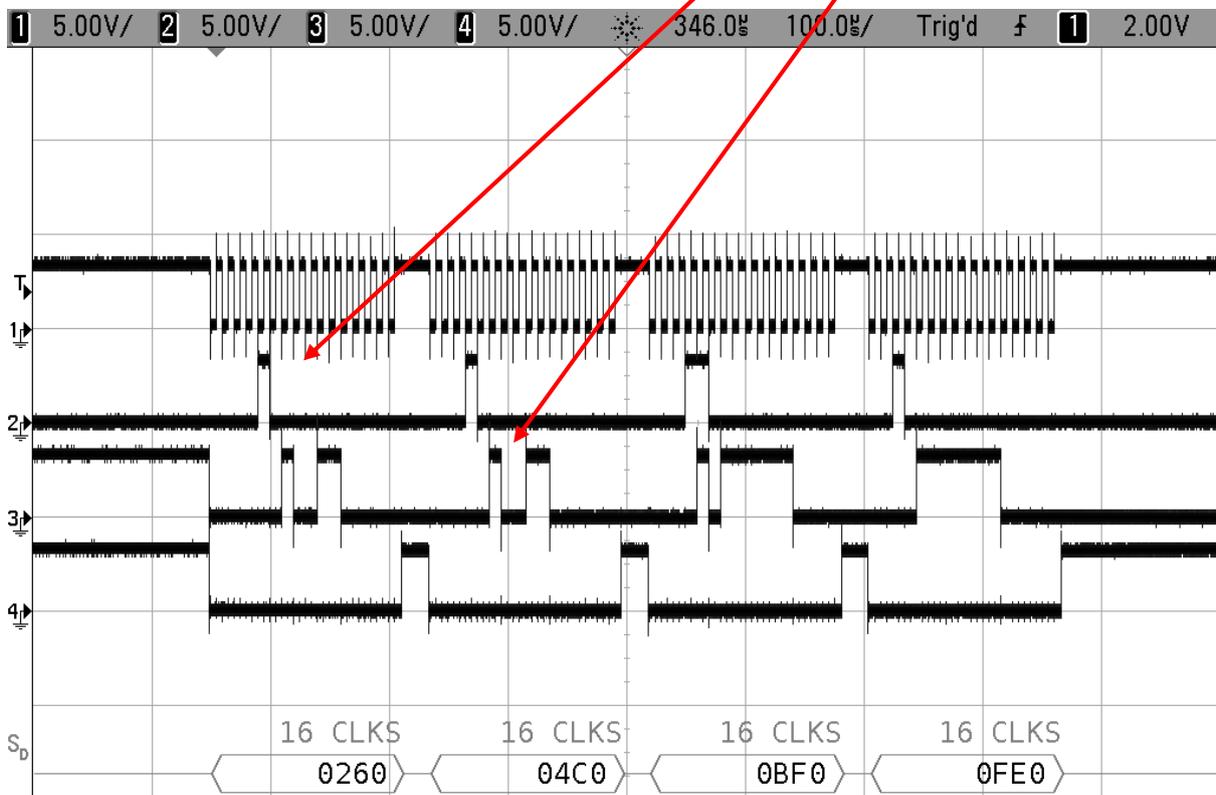


Voie 1 : SPI\_CLK  
 Voie 2 : SPI\_Dout (MOSI)  
 Voie 3 : SPI\_Din (MISO)  
 Voie 4 : SPI\_CS

borne 3 Header A  
 borne 7 Header A  
 borne 11 Header A  
 borne 15 Header A

Sélection de la voie 1

Résultat de la conversion de la voie 1



Mesurer la fréquence de l'horloge ? **100kHz**

Déterminer la durée de la conversion sur une voie ? **16 coups horloges**

Quelle est l'information qui circule sur de la voie SPI\_Dout (MOSI) ? **numéro de la voie d'entrée du CAN (1 à 4)**

Quelle est l'information qui circule sur la voie SPI\_Din (MISO) ? **résultat de la conversion sur 8 bits, seul les bits DB11 à DB4 sont utiles (cf page 6 documentation technique du convertisseur ADC084S21)**



Annexe 1:

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

