

# User Guide Zybo\_Camera

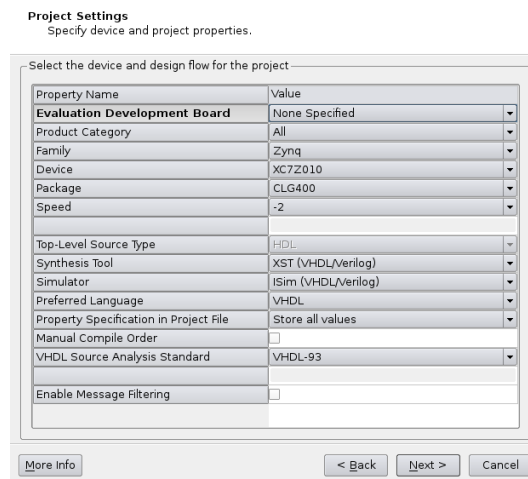
## Partie HW

Dans cette partie nous allons réaliser la partie matérielle du projet sous ISE. Elle contient les modules d'affichage et de synchronisation VGA ainsi que le module de récupération des données issues de la caméra et le stockage de ces données en mémoire RAM (voir Annexes A, B et C).

Ouvrez ISE Project Navigator, créer un nouveau projet (File -> New Project).

Sélectionner le répertoire de travail et nommer le projet, sélectionner HDL comme « Top-level source » puis cliquer sur le suivant.

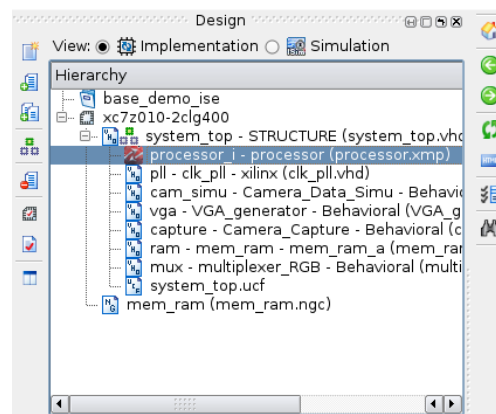
Nous allons maintenant configurer le type de carte comme ci-dessous.



Cliquer sur suivant puis valider.

Ajouter maintenant toutes les sources *.vhd* et *mem\_ram.ngc* au projet en cliquant droit sur la fenêtre « Hierarchy » puis « Add source », ajouter également les contraintes *system\_top.ucf*.

Copier coller le répertoire processor de l'archive dans le répertoire du projet créé. Ajouter maintenant la source *processor.xmp* contenu dans le répertoire processor copié. Cette source contient le design du projet avec les différentes IPs.

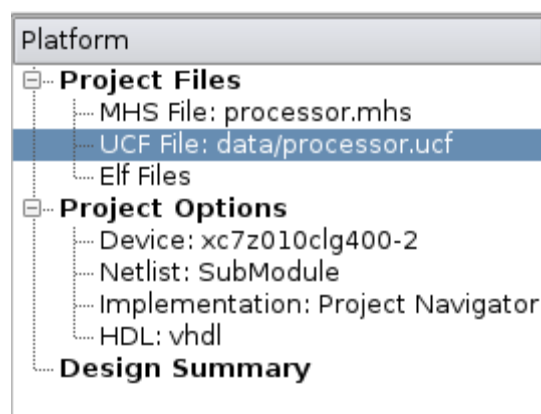


Double cliquer sur *processor*, vous allez être redirigé automatiquement vers le design sous EDK.

Dans l'onglet « System Assembly View » vous obtenez l'ensemble des différents blocs du Processing System. Les périphériques activés I2C0, UART1, SDO apparaissent en couleurs dans le bloc « I/O Peripherals », vous pouvez activer d'autres périphériques en cliquant sur ce bloc.

Sur l'onglet du haut « Ports », vous obtenez la liste de chaque port du système. Les ports *I2C0\_SDA* et *I2C0\_SCL* du *processing\_system7\_0 (IIC\_0)* sont connectés vers l'environnement extérieur de notre système. En effet ces ports seront associés au port PMOD de la carte Zybo.

Cliquer ensuite sur l'onglet du bas « Graphical Design View », vous obtenez la vue d'ensemble du design, c'est notamment ici que vous pouvez ajouter vos propres IPs disponibles dans l'onglet « IP Catalog » sur la gauche.

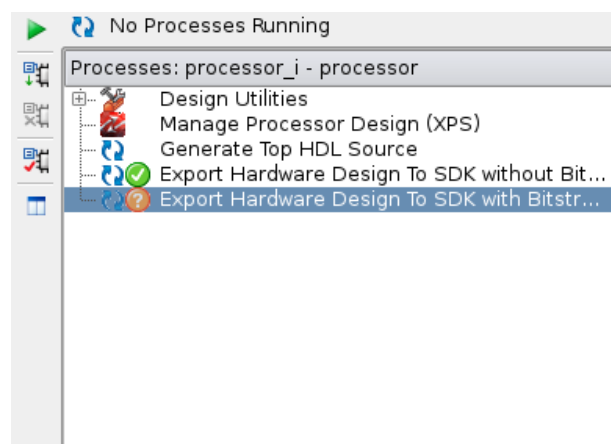


Le fichier *processor.ucf* contient les contraintes associées aux ports externes I2C0.

# PIN Constraints

```
NET processing_system7_0_I2C0_SDA LOC = T11 | IOSTANDARD = LVCMOS33;  
NET processing_system7_0_I2C0_SCL LOC = V15 | IOSTANDARD = LVCMOS33;
```

Fermer maintenant EDK et exporter le *bitstream* sous ISE vers SDK.

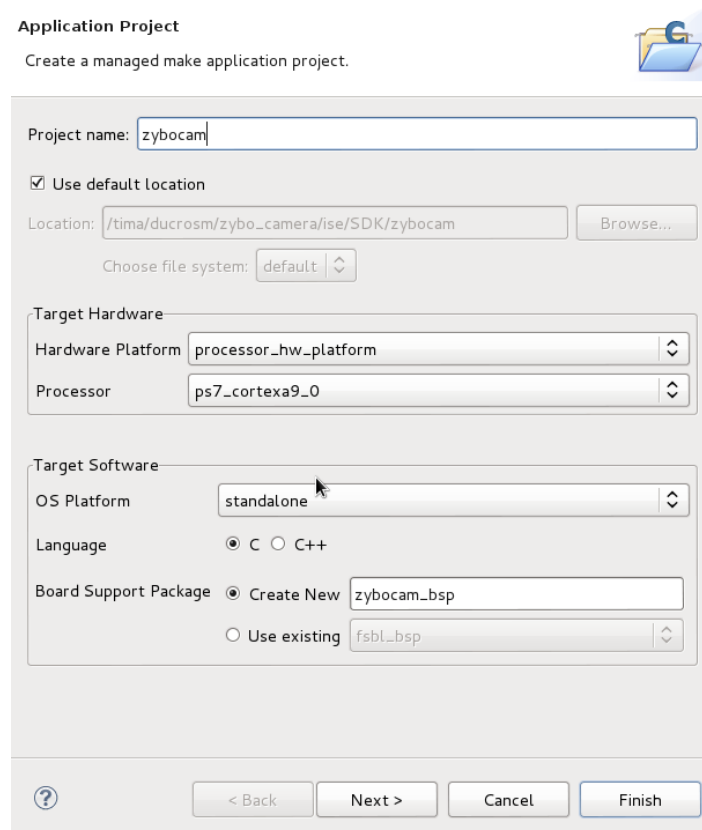


Générer et exporter le bitsream vers SDK en double cliquant sur « Export Hardware Design to SDK with Bitstream ». Attendez la fin de la génération, SDK s'ouvrira automatiquement.

## Partie SW

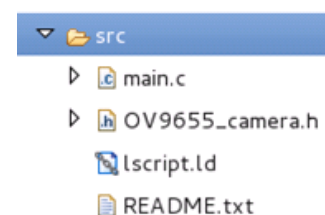
Dans cette partie nous allons réaliser la partie logicielle du projet sous SDK, à savoir la gestion de la communication I2C et UART pour la configuration de la caméra.

Sélectionner votre workspace puis créer un nouveau projet (File -> New -> Application Project). Nommer le projet et sélectionner « standalone » comme OS Platform puis « Create New » comme « Board Support Package ».

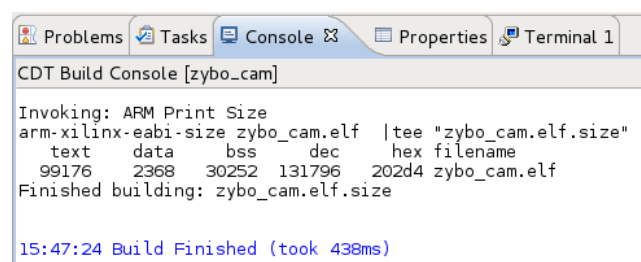


Cliquer sur suivant et sélectionner « Empty Application » puis valider.

Copier les sources *main.c* et *OV9655\_camera.h* du répertoire SDK de l'archive et coller les dans le répertoire src de votre projet. Cliquer sur la fenêtre « Project Explorer » sur la gauche et taper F5. Les fichiers sources ajoutés apparaîtront alors dans le dossier src.



Vous pouvez ajouter vos propres fonctions au main. Après chaque modification du projet **veiller à bien enregistrer le main** pour recompiler le projet, le message « Build Finished » doit s'afficher dans la console.



## Problèmes :

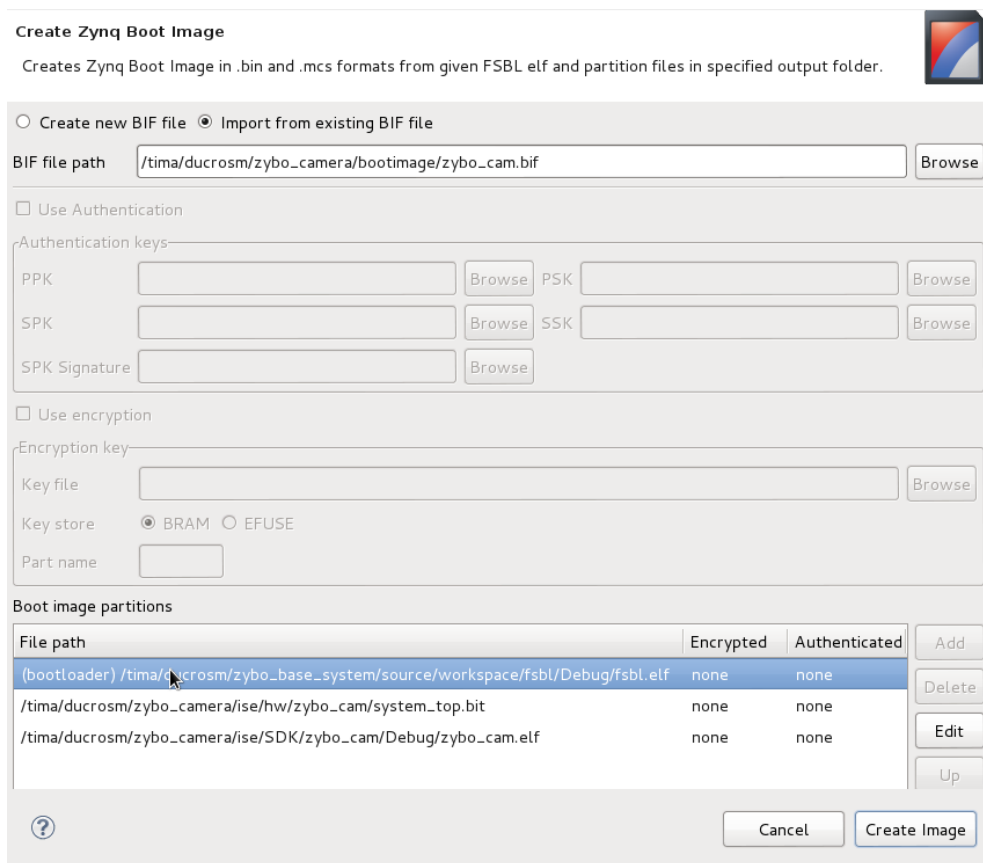
Si lors de la compilation une erreur concernant le fichier *xil-crt0.S* du BSP apparaît, modifier la ligne 149 de ce fichier en *bl main.c*.

## Boot sur la carte SD :

Après avoir enregistré le *main.c* pour compiler le projet, créer la Zynq Book Image (Xilinx Tools -> Create Zynq Book Image).

Pour cela créer un *fichier.bif* et ajouter le *fsbl.elf* contenu dans l'archive comme « bootloader » puis ajouter comme « datafile » votre *bitstream* généré à l'étape précédente et enfin votre *code.elf* contenu dans le répertoire Debug de votre projet.

Veiller à bien respecter l'ordre d'ajout des fichiers. Sélectionnez le répertoire de sortie « Output Path » et nommer le fichier *boot.bin*.



Vous pouvez ensuite rapatrier le *boot.bin* généré et le coller sur la carte SD.

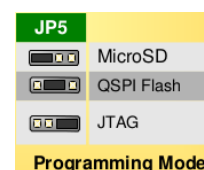
## Flash par JTAG :

Mettre le cavalier JP5 de la carte Zybo en position JTAG.

Cliquer droit sur votre projet puis sélectionner « Build Project ».

Flasher la carte (Xilinx Tools -> Program FPGA).

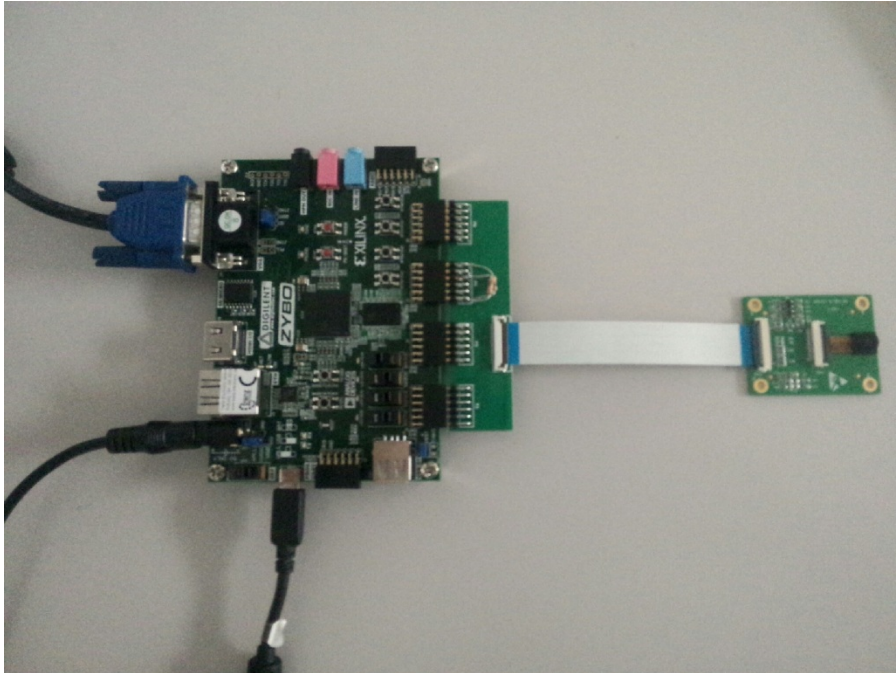
Enfin cliquer droit sur le projet puis sélectionner Run As -> Launch on Hardware.



## Mise en route

Relier :

- La carte d'interface caméra aux PMODs de la carte
- Le câble d'alimentation 5V à la carte
- Le port UART de la carte au port USB du PC
- Le port VGA de la carte à l'écran



Insérer la carte SD dans la Zybo.

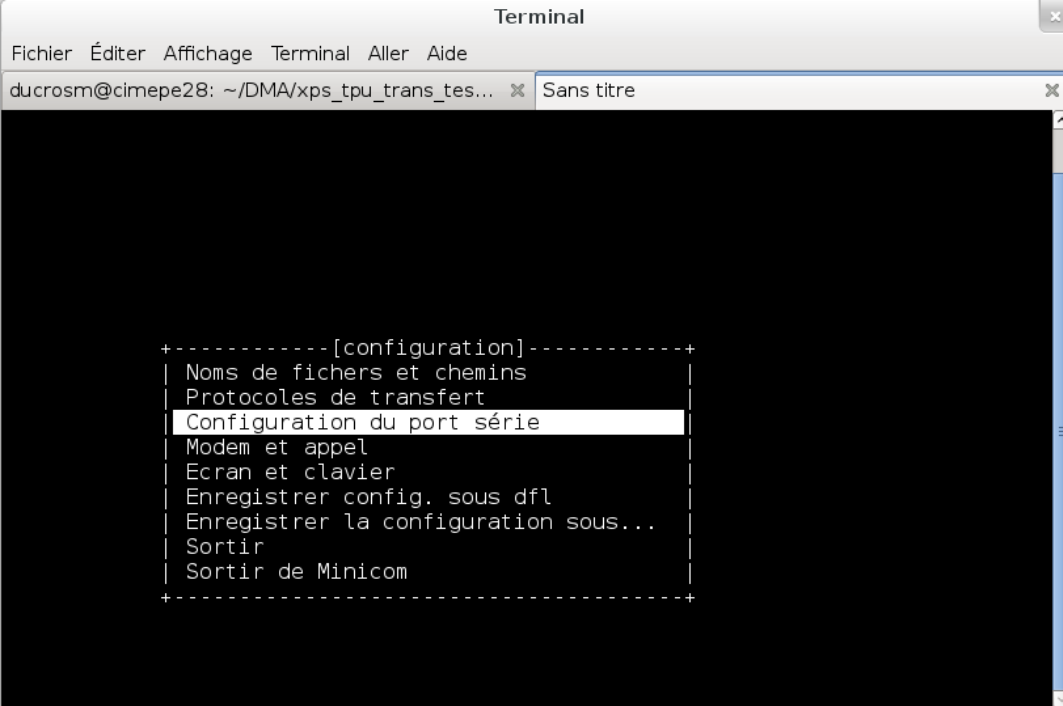
Mettre sous tension la carte (SW4 sur ON), la led DONE doit s'allumer en vert. Si ce n'est pas le cas recopier le fichier boot.bin sur la carte SD.

Lors de la mise sous tension de la carte, veillez à ce que la switch *SW0* soit en position basse afin de ne pas afficher les données en provenance de la caméra. Si aucune image ne s'affiche après la mise en position haute de *SW0*, appuyez sur le bouton reset de la carte *PS-SRST*.

## Configuration Minicom

La carte doit être alimentée avant de continuer.

Ouvrir un terminal et entrer la commande `minicom -s` afin de lancer le mode de configuration.

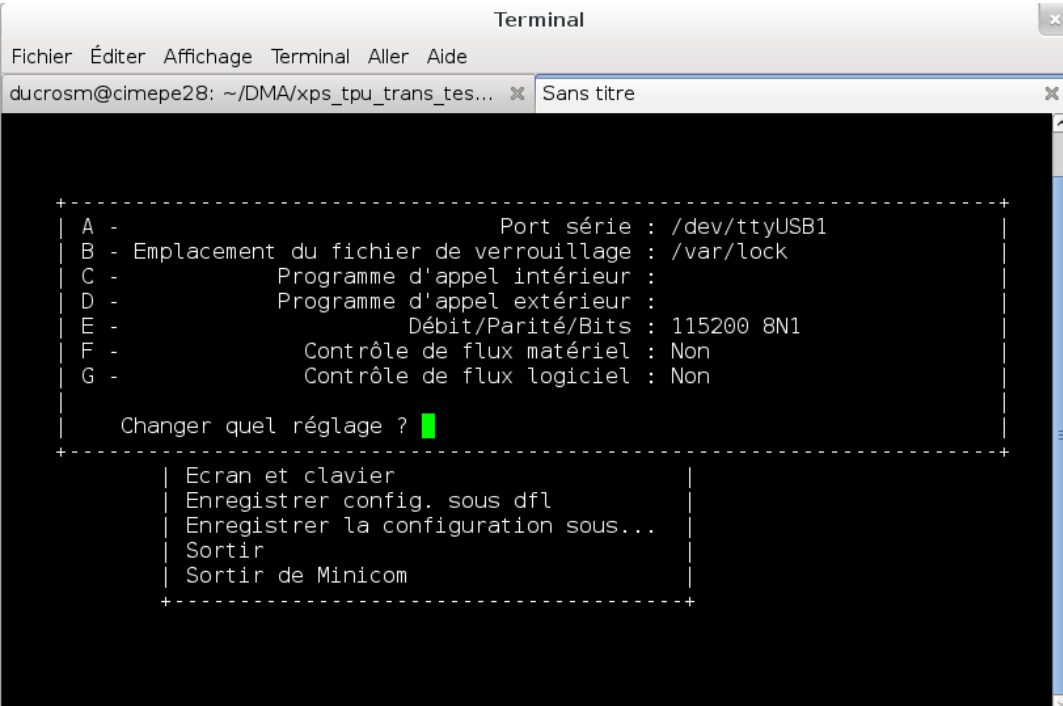


```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
ducrosm@cimepe28: ~/DMA/xps_tpu_trans_tes... x Sans titre

+-----[configuration]-----+
| Noms de fichiers et chemins |
| Protocoles de transfert     |
| Configuration du port série |
| Modem et appel              |
| Ecran et clavier            |
| Enregistrer config. sous dfl|
| Enregistrer la configuration sous...|
| Sortir                      |
| Sortir de Minicom           |
+-----+


```

Sélectionner « Configuration du port série ».



```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
ducrosm@cimepe28: ~/DMA/xps_tpu_trans_tes... x Sans titre

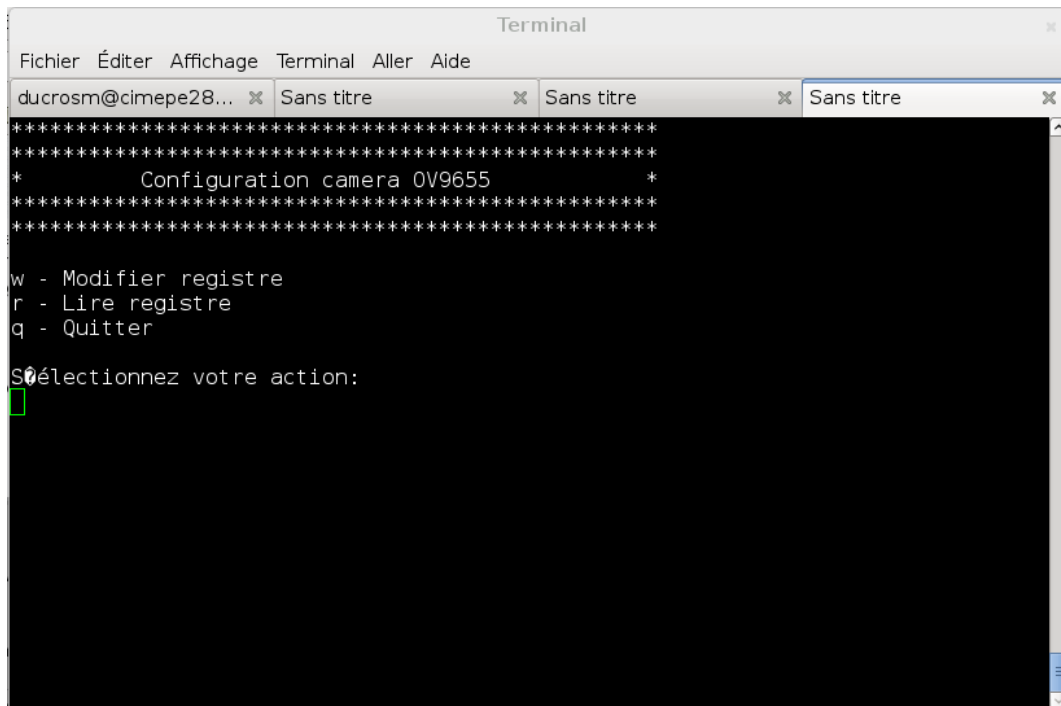
+-----+
| A - Port série : /dev/ttyUSB1 |
| B - Emplacement du fichier de verrouillage : /var/lock |
| C - Programme d'appel intérieur : |
| D - Programme d'appel extérieur : |
| E - Débit/Parité/Bits : 115200 8N1 |
| F - Contrôle de flux matériel : Non |
| G - Contrôle de flux logiciel : Non |
| Changé quel réglage ? █ |
+-----+
| Ecran et clavier |
| Enregistrer config. sous dfl |
| Enregistrer la configuration sous... |
| Sortir |
| Sortir de Minicom |
+-----+


```

Taper [A] pour changer le port série et [F] pour désactiver le contrôle de flux matériel. Quitter le mode de configuration (Sortir). Maintenant chaque caractère pressé au clavier est envoyé au port sélectionné et chaque caractère reçu est affiché à l'écran.

## Configuration registres caméra

Après avoir configuré minicom (vu précédemment), l'interface de configuration ci-dessous s'affiche.



```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
ducrosm@cimepe28... x Sans titre x Sans titre x Sans titre x
*****
*****
* Configuration camera 0V9655 *
*****
*****
w - Modifier registre
r - Lire registre
q - Quitter
Sélectionnez votre action:
█
```

### Modification d'un registre :

Tapez [w] puis entrez l'adresse du registre à modifier suivi de [Entrée], ensuite tapez la nouvelle valeur du registre suivi de [Entrée]. Si les valeurs affichées correspondent à votre requête tapez [y] sinon tapez [n] et recommencez l'opération.

### Lecture d'un registre :

Tapez [r] puis entrez l'adresse du registre à lire suivi de [Entrée], la valeur lue s'affiche à l'écran.

### Quitter:

Tapez [q] pour mettre fin à la configuration des registres. Pour relancer le menu il sera nécessaire de reseter la carte.

### Problèmes :

Si lors de la configuration minicom reste figer lors de l'envoi des trames :

- Vérifier que minicom est bien configuré.
- Vérifier que la carte est bien alimentée et la caméra connectée.
- Vérifier dans *fsbl\_hooks.c* de la fsbl de la démo que la lecture de l'adresse MAC (au sein de la fonction *FsblHookBeforeHandoff*) de la carte est bien commenté car cette fonction utilise les ports I2C et perturbe la communication.

### Initialisation caméra :

Dans cette partie sont détaillés les registres initialisés dans la fonction `init_OV9655`.

La définition de la résolution de l'image doit être définie en premier lieu pour ne pas engendrer du bruit pour la suite.

#### *Résolution VGA 640x480*

Adresse	Registre	Valeur hexa	Valeur binaire	Valeur décimale
0x32	HREF	0xBF	1011 1111	
0x17	HSTART	0x1D	0001 1010 111	239
0x18	HSTOP	0x6D	0110 1010 111	879
0x03	VREF	0x12	0001 0010	
0x19	VSTART	0x01	0000 0001 100	10
0x1A	VSTOP	0x3D	0011 1101 100	490

Les bits [5 : 3] et [2 : 0] du registre HREF correspondent respectivement au LSB de HSTOP et HSTART. De même pour les bits [5 : 3] et [2 : 0] du registre VREF correspondent au LSB de VSTOP et VSTART. Les bits [7 : 6] de HREF doivent restés à « 10 » car il configure l'offset de sortie des données.

#### *Résolution 1280x1024*

Adresse	Registre	Valeur hexa	Valeur binaire	Valeur décimale
0x32	HREF	0xA4	1010 0100	
0x17	HSTART	0x1A	0001 1010 100	212
0x18	HSTOP	0xBA	1011 1010 100	1492
0x03	VREF	0x12	0001 0010	
0x19	VSTART	0x01	0000 0001 010	10
0x1A	VSTOP	0x81	1000 1101 010	1034

Pour utiliser cette résolution il faut également modifier la partie hardware car les synchro VGA et la capture d'images a été configurée pour du 320x240 (taille frame mémoire).

Avec cette configuration, une bande horizontale (rafraichissement) apparait sur l'image.

#### *Mode RGB 15fps*

Adresse	Registre	Valeur hexa
0x12	COM7	0x03

L'image est assez saccadée car la fréquence de capture de 8bits est de 24MHz or un pixel est défini sur 16bits (RGB565) et la fréquence du VGA est de 24MHz. Pour augmenter le nombre de frames il faut augmenter Pclk par défaut à 24MHz (voir section configuration pixel clock).

#### *Mode RGB565avec échelle maximum des données de sortie*

Adresse	Registre	Valeur hexa
0x40	COM15	0xD0



Données RGB sur 16bits : rrrr rggg gggg bbbb avec une plage de valeur définie sur [0x00 : 0xFF].

#### *Configuration de la valeur maximum du correcteur de gain*

Adresse	Registre	Valeur hexa	Valeur gain
0x32	COM9	0x0A	2x
		0x1A (val init)	4x
		0x2A	8x
		0x3A	16x
		0x4A	32x
		0x5A	64x
		0x6A	128x

Plus ce gain augmente plus l'image apparaît bruitée. Les valeurs moyennes du gain sont calculées automatiquement.

#### *Configuration pixel clock*

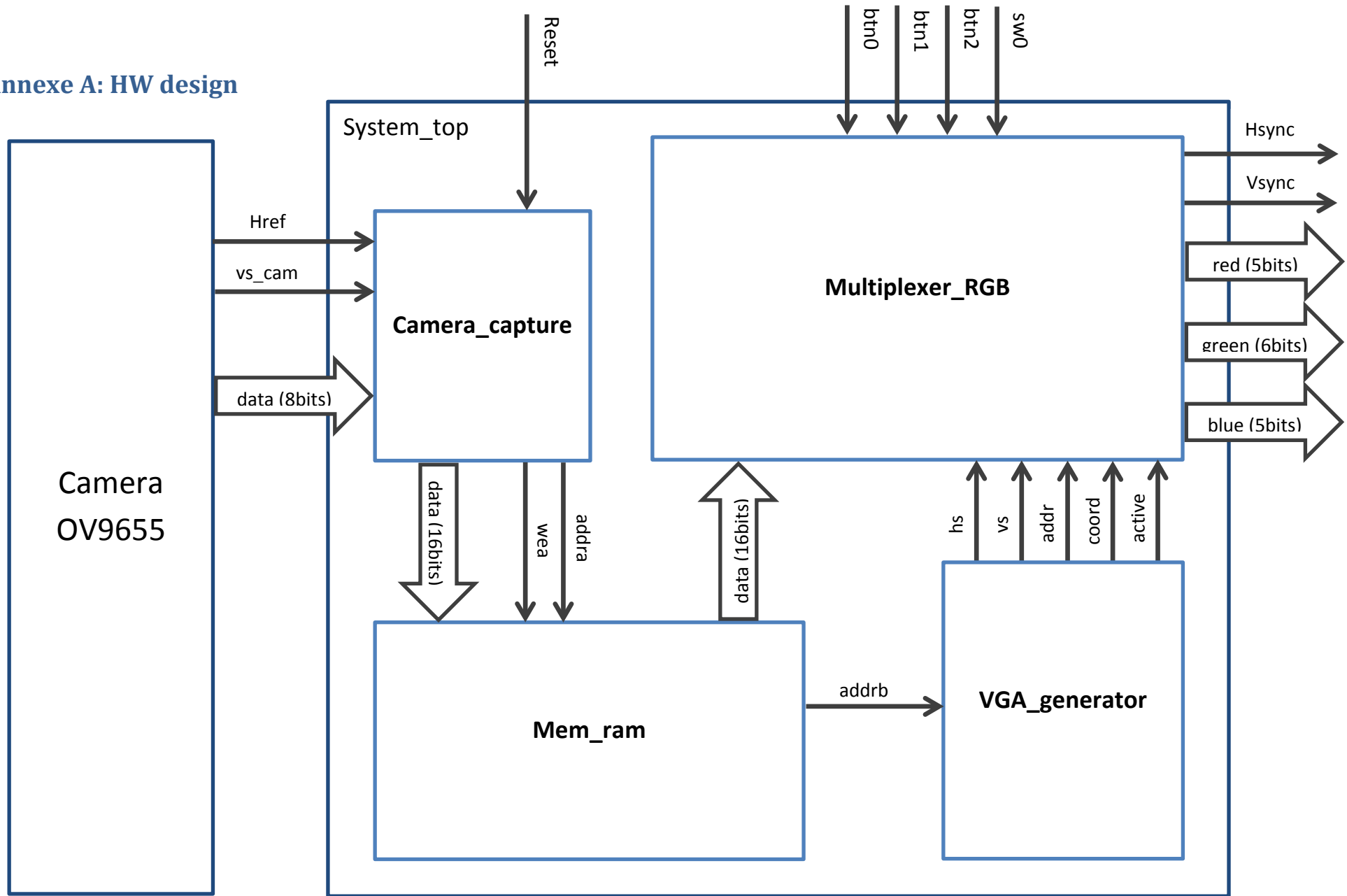
Adresse	Registre	Valeur hexa	Valeur gain
0x11	CLKRC	0x00 (val init)	÷1
		0x01	÷2
		0x02	÷3
		val-1	÷val
0x6B	DBLV	0x0A (val init)	Bypass
		0x4A	4x
		0x8A	6x
		0xCA	8x

L'image est fluide pour Pclk défini à 96MHz (CLKRC à 0x00 et DBLV à 0x4A) mais également bruitée (apparition de bandes).

#### *Autres options*

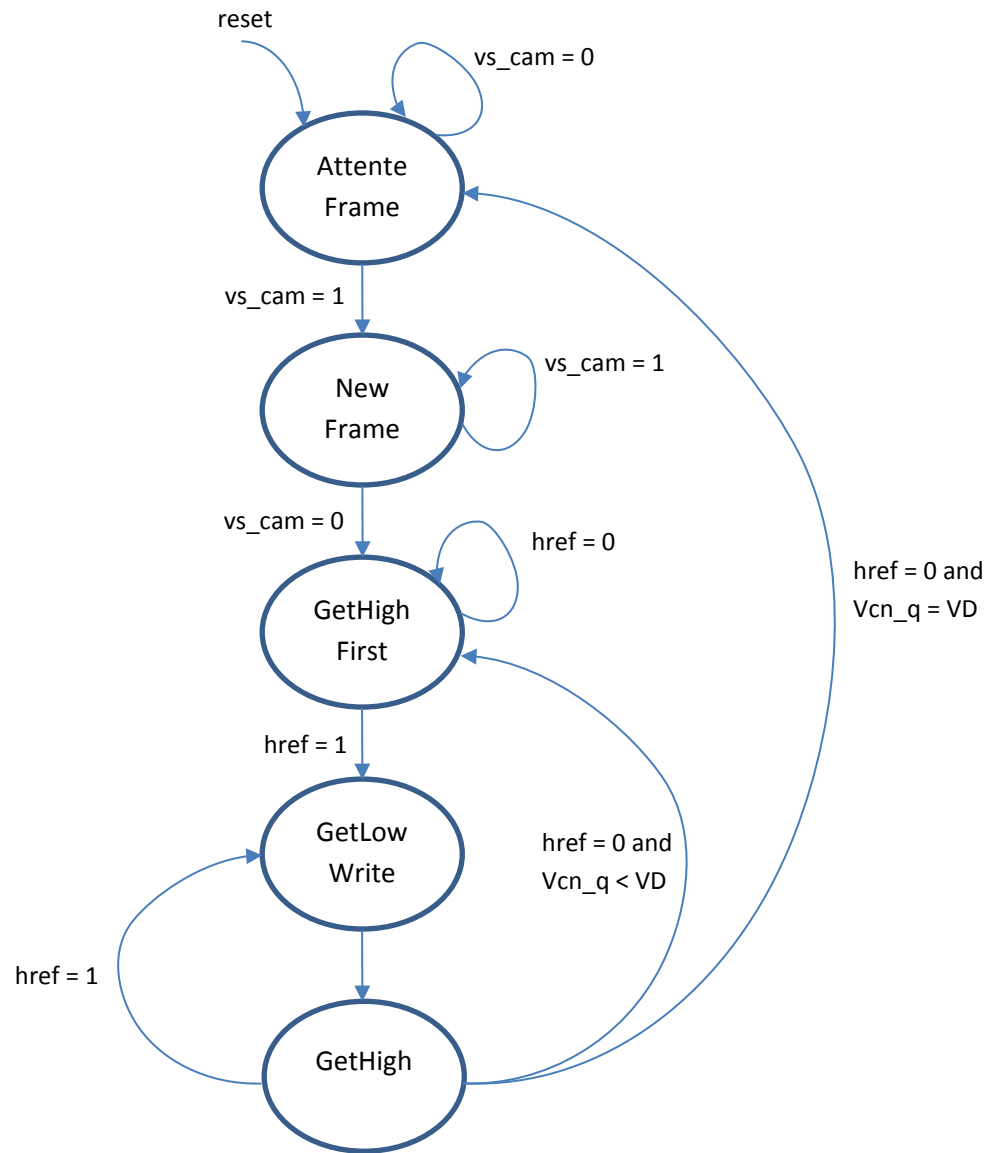
Les autres valeurs des registres sont commentées dans la fonction d'initialisation de la caméra (voir OV9655\_camera.h).

## Annexe A: HW design



\*Les signaux d'horloge des modules ne sont pas indiqués. CLK\_CAM = CLK\_VGA = 24MHz générées par le module CLK\_PLL.

## Annexe B: FSM camera capture



## Annexe C: VGA generator

Dans cette annexe est rappelé le principe de fonctionnement du module VGA\_generator avec l'utilisation des compteurs (horizontal et vertical) pour respecter le timing VGA.

