

**Architecture des systèmes ordinés et VHDL**

ELE748

**Travail remis à**

Simon Pichette

**Laboratoire 2**

**Par**

Jonathan Lapointe (LAPJ05108303)   
Kévin Parent Legault (PARK22049009)

**Rédigé le**

**École de technologie supérieure**  
Département de Génie Électrique

Table des matières

[Introduction 3](#_Toc421474439)

[Architecture du système 4](#_Toc421474440)

[Description du système 4](#_Toc421474441)

[Configuration Qsys 5](#_Toc421474442)

[Vue globale 5](#_Toc421474443)

[Partie Cpu 5](#_Toc421474444)

[Partie Vidéo 5](#_Toc421474445)

[Code VHDL 6](#_Toc421474446)

[Architecture logicielle 9](#_Toc421474447)

[Description de logiciel 9](#_Toc421474448)

[Code C 10](#_Toc421474449)

[Display.c 10](#_Toc421474450)

[Display.h 14](#_Toc421474451)

[Mouse.c 15](#_Toc421474452)

[Mouse.h 20](#_Toc421474453)

[JtagUart.c 22](#_Toc421474454)

[jtagUart.h 23](#_Toc421474455)

[Hardware.h 24](#_Toc421474456)

[Main.c 25](#_Toc421474457)

[Discussion 26](#_Toc421474458)

[Conclusion 27](#_Toc421474459)

# Introduction

Dans le cadre du cours d’architecture des systèmes ordinés et VHDL, nous avons réalisé un deuxième système comportant une architecture modulaire. Cette architecture a été générée en utilisant l’utilitaire d’intégration système d’Altera, QSYS. Nous avons ensuite conçu une architecture logicielle fonctionnant sur ce système. Nous avons finalement déployé le tout sur le Cyclone V de la carte DE1 SOC d’Altera. Le système conçu est une application d’infographie simplifiée (NIOS draw) sur écran VGA qui sera contrôlée à l’aide d’une souris PS2.

# Architecture du système

## Description du système

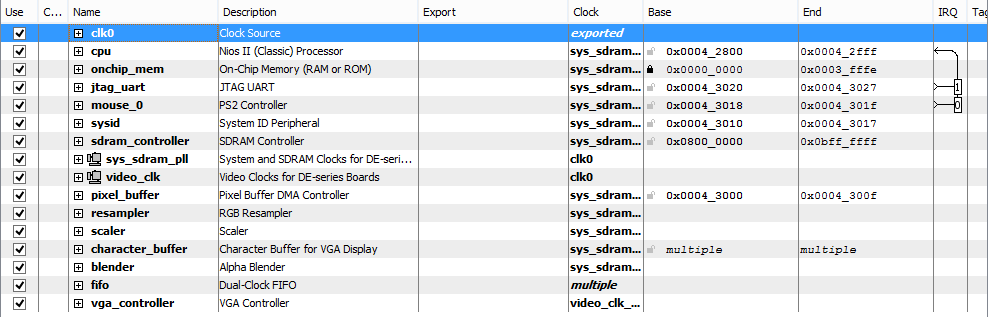
L’architecture système est composée de plusieurs modules tous générés à l’aide de QSYS. Ce dernier génère ensuite une composante VHDL qui contient les modules de notre architecture. Cette composante doit se faire connecter dans un fichier TOP au monde extérieur.

Notre architecture comprend les périphériques suivants :

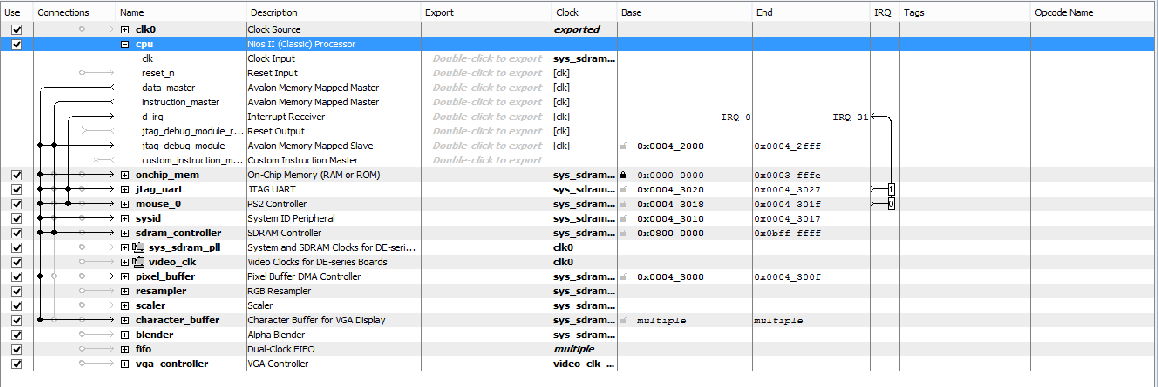
|  |  |  |
| --- | --- | --- |
| Composante | Module Qsys | Description |
| cpu | NIOS II(classic) | Processeur RISC 32 bits |
| Onchip\_mem | On-chip memory | Mémoire utilisé par pixel buffer (262143 octets, 32 bits) |
| Jtag\_uart | JTAG UART | Permet d’écrire sur la console NIOS |
| Mouse\_0 | PS2 controller | Communique avec la souris PS2 |
| Sysid | System id controller | Numéro d’identification système |
| Sdram\_controller | SDRAM Controller | Contrôleur pour la SDRAM qui est sur la carte de développement (64Mbytes) |
| Sys\_sram\_pll | SDRAM clock | Déphasage l’horloge pour faire fonctionner la SDRAM |
| Video\_clk | Video clock for DE board | Fourni une horloge de 25 Mhz au« vga controller » |
| Pixel\_buffer | Pixel buffer DMA controller | Affiche les pixel sur l’écran(320X240)(background) |
| Resampler | RGB Resampler | Reformate l’image 320X240 à 640X480 |
| Character\_buffer | Character Buffer for VGA display | Affiche des caractères ASCII (80X60) (foreground) |
| blender | Alpha Blender | Mélange le background et le foreground ensemble |
| fifo | Dual clock FiFO | Permet faire le pont entre les 2 domaines d’horloges (50Mhz à 25Mhz) |
| Vga\_controller | VGA Controller | Module qui gère le protocole VGA  (25 Mhz) qui est connecté avec le DAC de sortie |

## Configuration Qsys

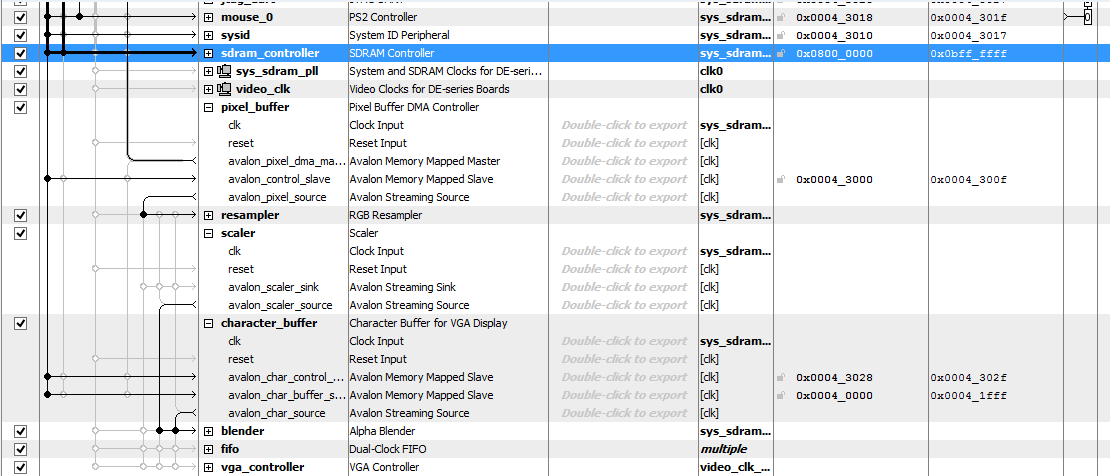
### Vue globale



### Partie Cpu



### Partie Vidéo

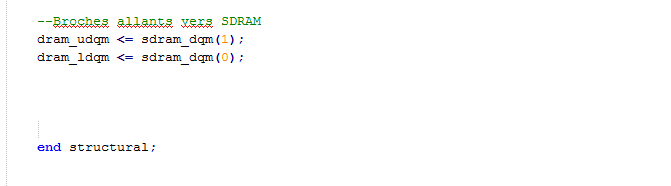


## Code VHDL

## 

## 

## 



# Architecture logicielle

Dans ce laboratoire, l’architecture logicielle permet d’effectuer plusieurs choses, tel que :

* Recevoir, mettre en mémoire et décoder les trames envoyées par la souris PS2;
* Rafraichir les pixels et effacer l’écran VGA ;
* Envoyer les coordonnées de position et les actions de la souris à la console NIOS;
* Afficher un curseur (caractère ASCII dans notre cas) qui se déplace sur l’écran en fonction de la position de la souris;
* Afficher la position du curseur au bas de l’écran VGA.

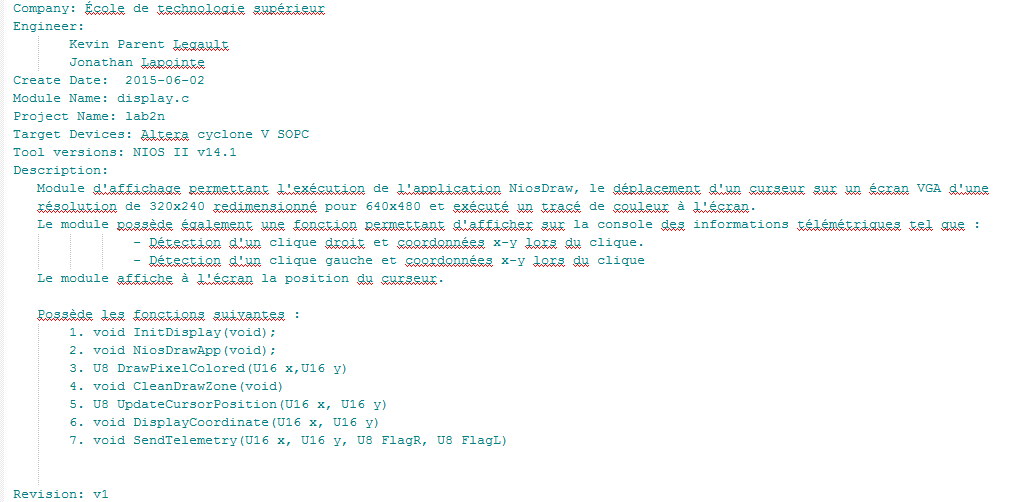
## Description de logiciel

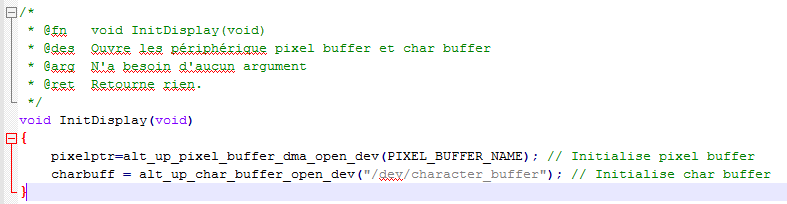
Voici un tableau qui décrit plus en détail les librairies conçues pour ce projet :

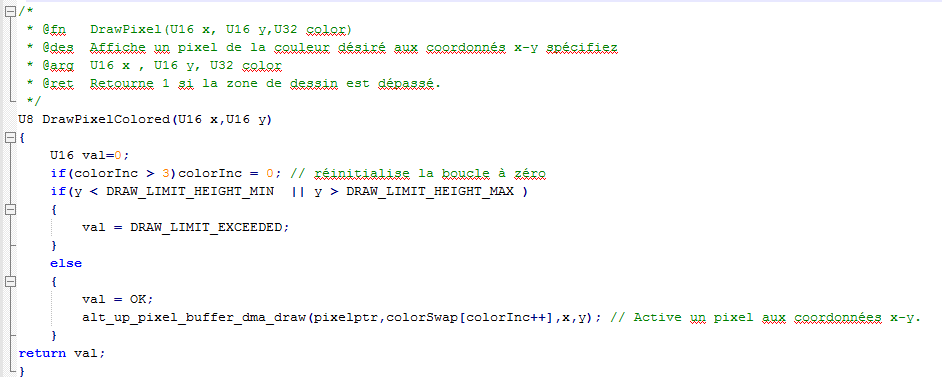
|  |  |  |
| --- | --- | --- |
| Librairie | Nom des Fonctions | Description |
| Main (.c) | main | Routine principale |
| Display(.c/.h) | InitDisplay | Initialise pixel buffer et char buffer |
| DrawPixel | Rafraichit la couleur d’un pixel selon sa position x et y |
| DrawPixelColored | Affiche un pixel avec couleur en alternance selon x,y |
| CleanDrawZone | Efface la zone de dessin |
| UpdateCursorPosition | Rafraichit les valeurs de la position du curseur à l’écran |
| NiosDrawApp | Routine principale de rafraichissement de l’écran |
| DisplayCoordinate | Affiche les coordonnées au bas de l’écran |
| SendTelemetry | Rafraichit les valeurs de télémétrie envoyées à la console Nios |
| Mouse(.c/.h) | ps2\_isr | Routine d’interruption : mise en mémoire de X,Y,SW des évènements souris dans des FIFO |
| mouseInit | Initialise la souris et les interruptions |
| mouseGetNbEvent | Retourne le nombre d’évènement souris reçu |
| mouseGetX | Extrait une valeur de X du fifo |
| mouseGetY | Extrait une valeur de Y du fifo |
| mouseGetSwL | Extrait une valeur de bouton gauche du fifo |
| mouseGetSwR | Extrait une valeur de bouton droit du fifo |
| mousePtrOutInc | Incrémente le pointeur de sortie du fifo et décrémente «NbEvent »; doit être appelé après avoir fait tous les « get » |
| JtagUart(.c/.h) | jUartSendString | Envoie sur console Nios une chaîne de caractère ou un tableau se terminant par un «NULL» |
| jUartSendVar | Envoie une valeur décimale sur la console Nios |
| Hardware (.h) |  | Définition des types et inclusion des librairies utilisées dans le projet (BSP) |

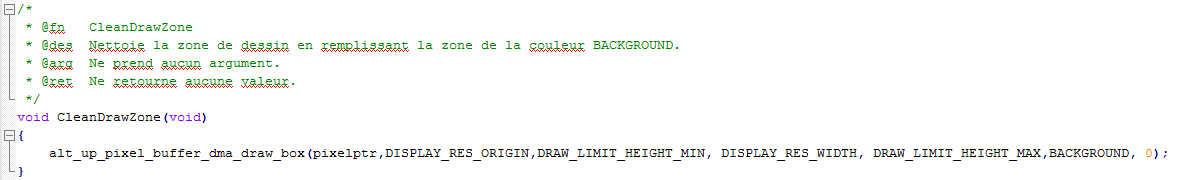
## Code C

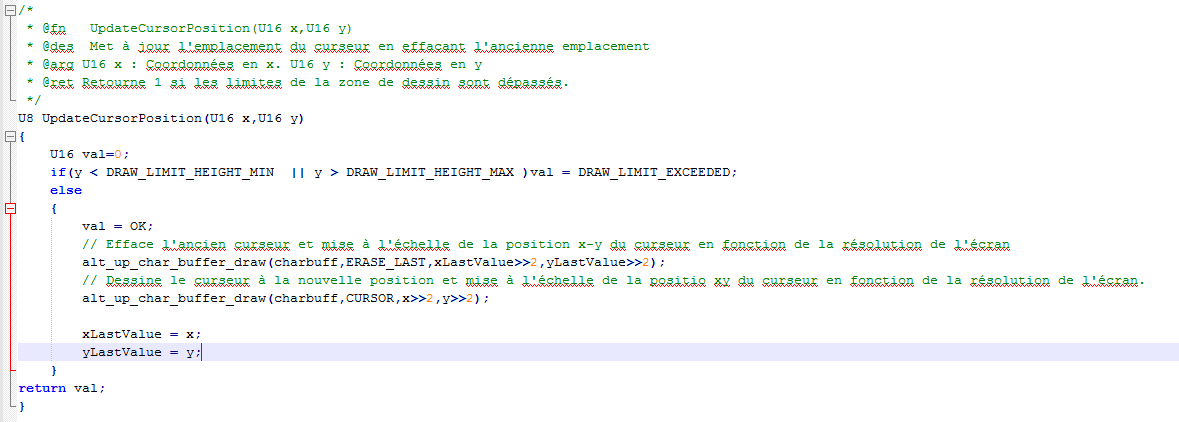
### Display.c

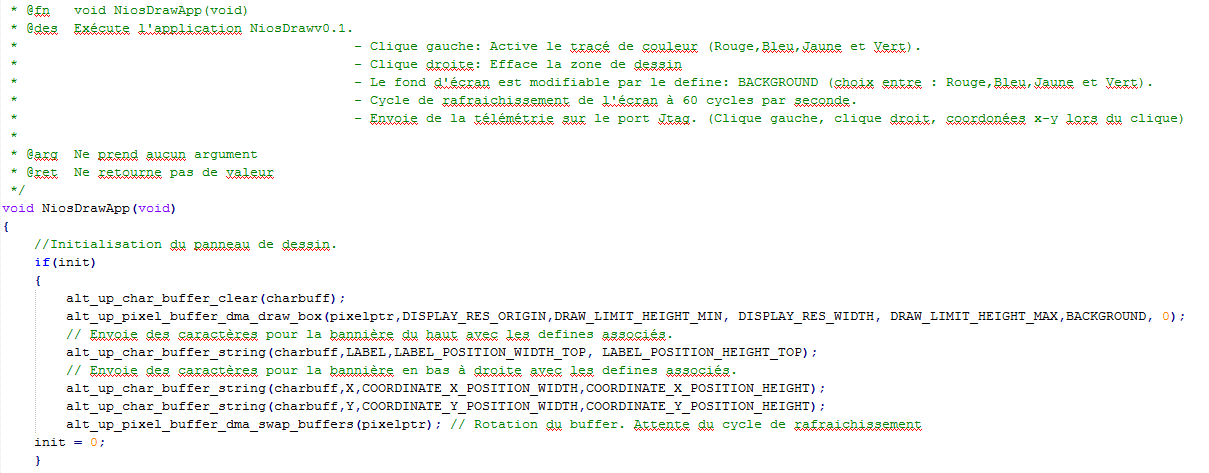


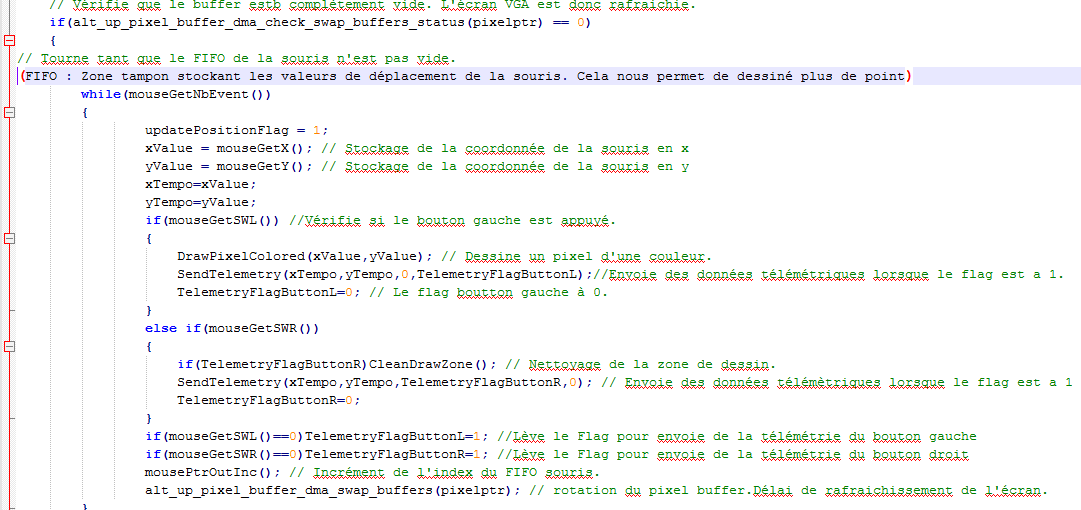


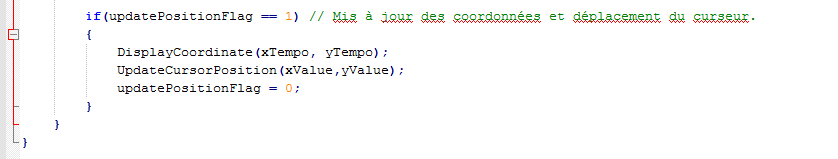


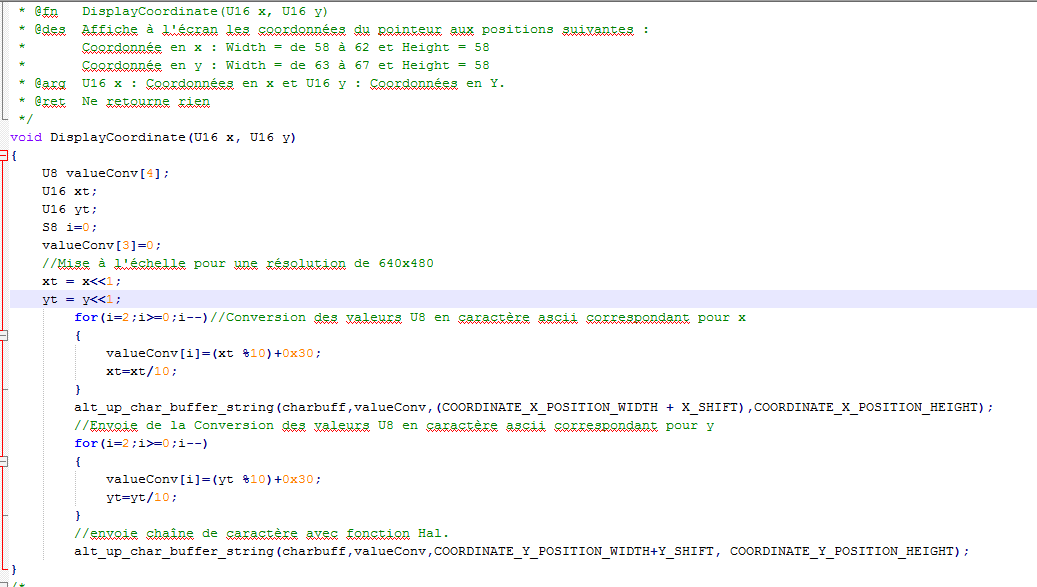


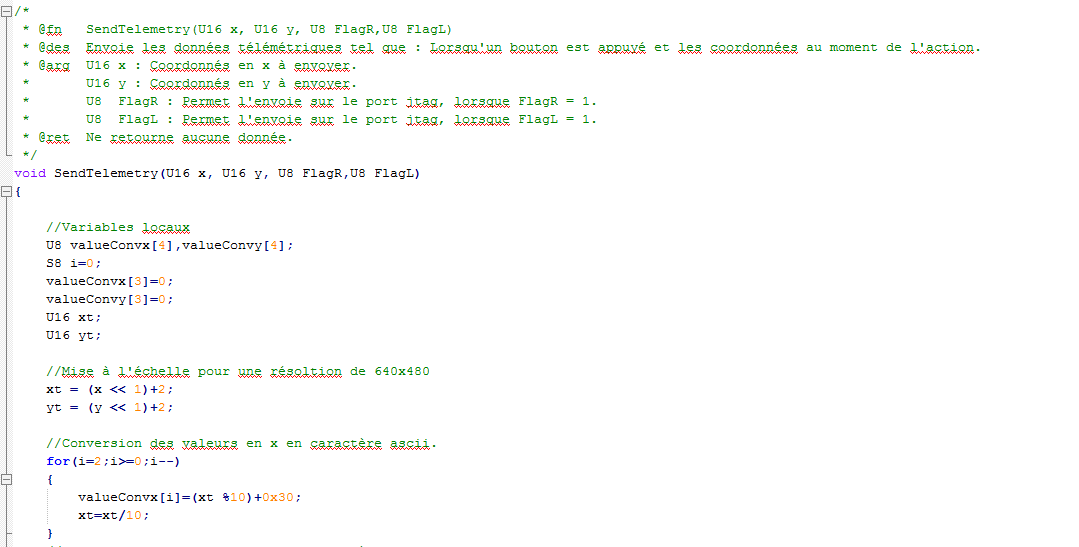


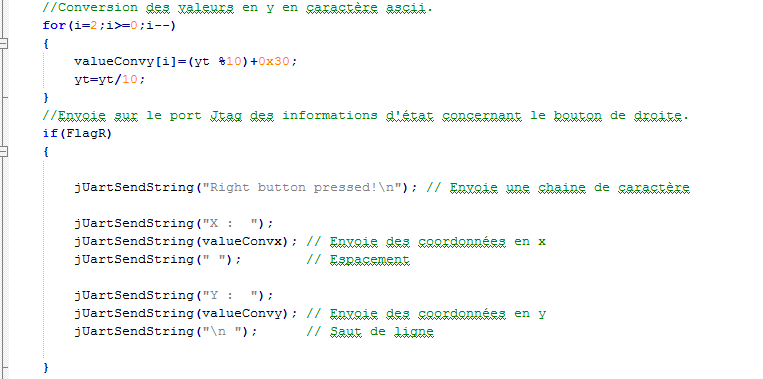


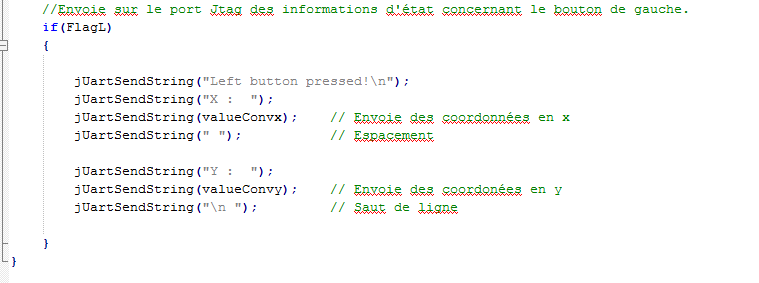




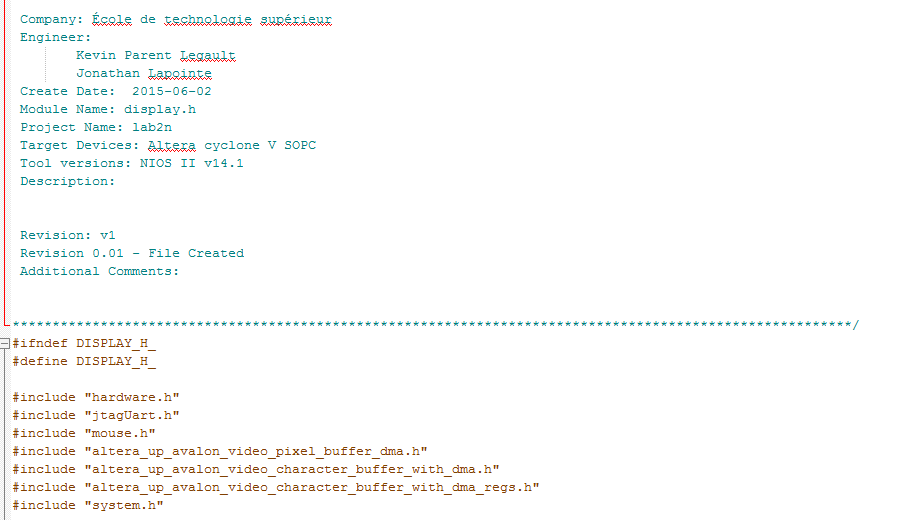


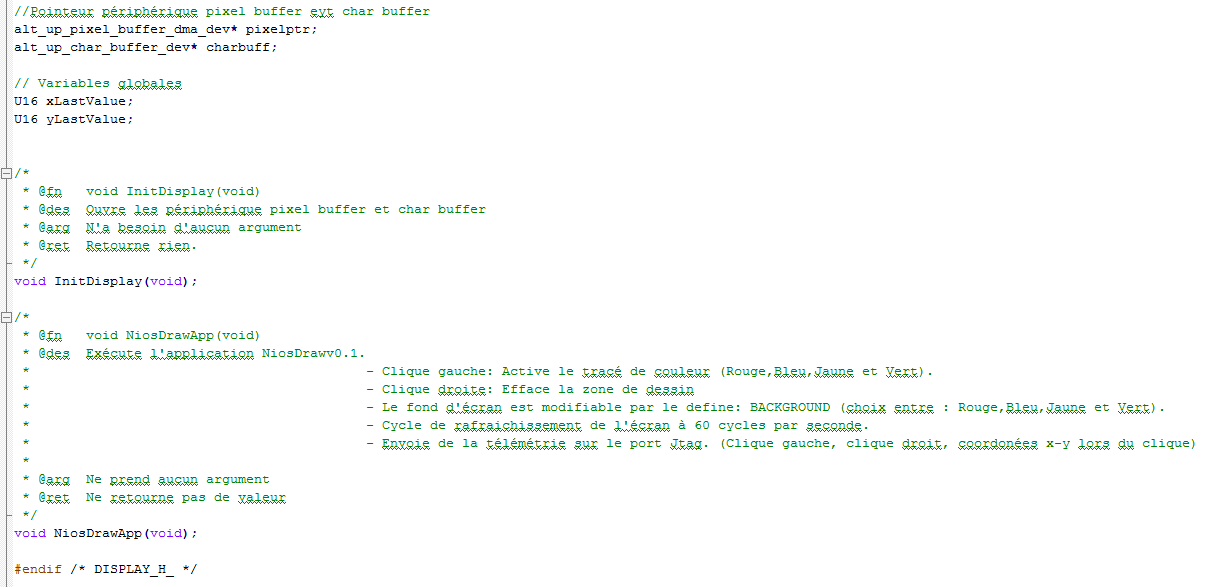




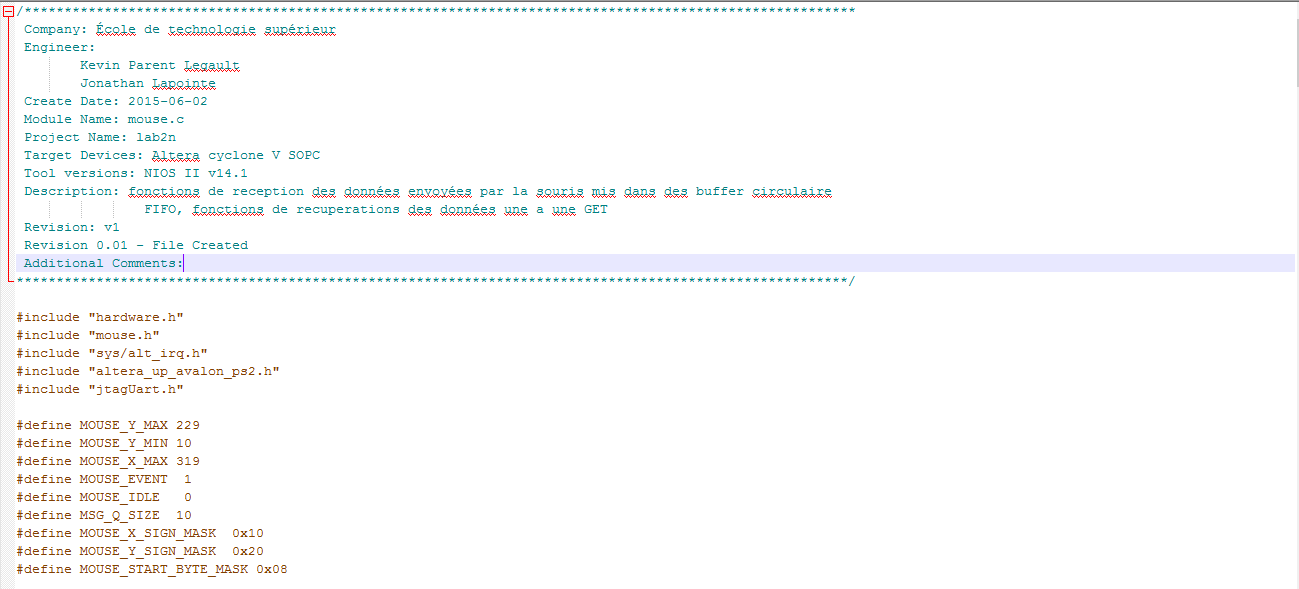


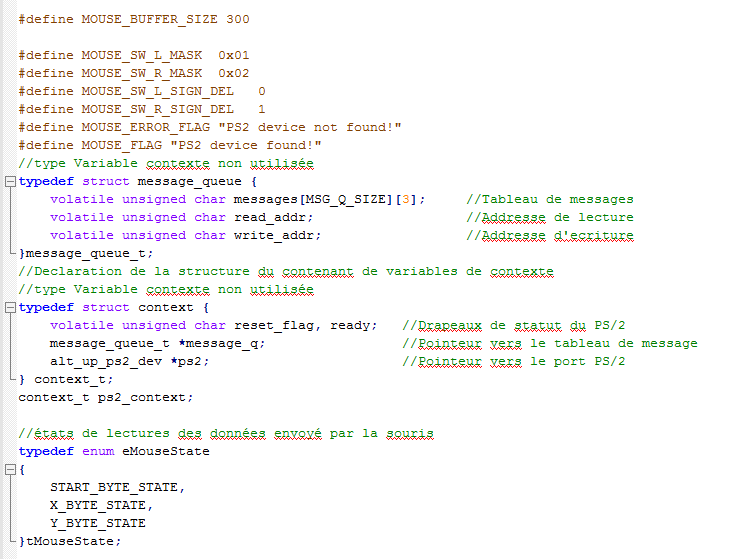
### Display.h

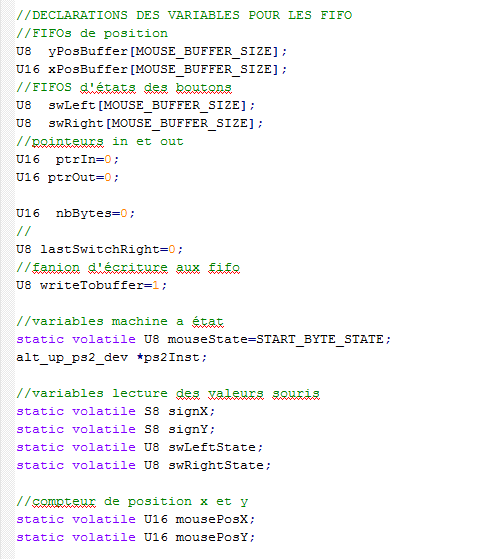


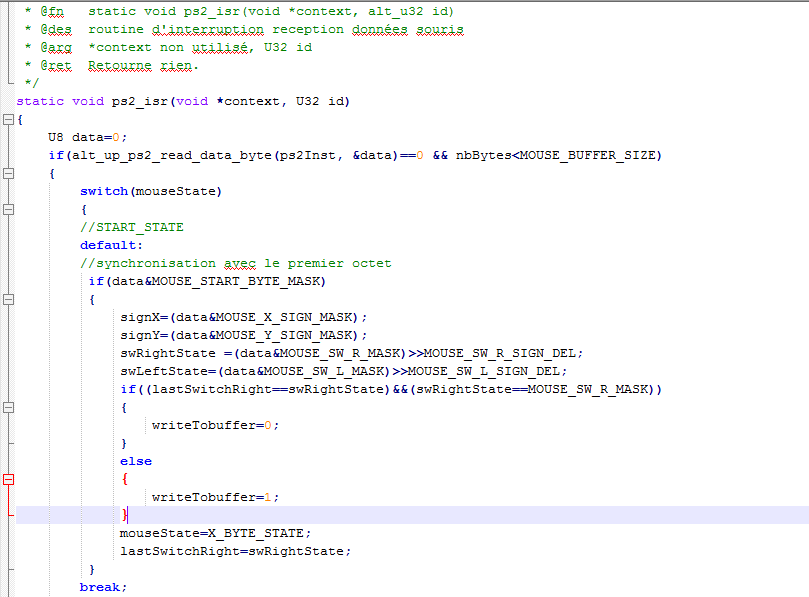


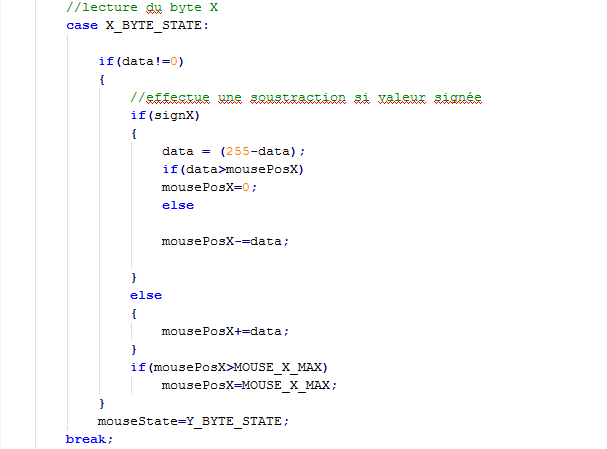
### Mouse.c

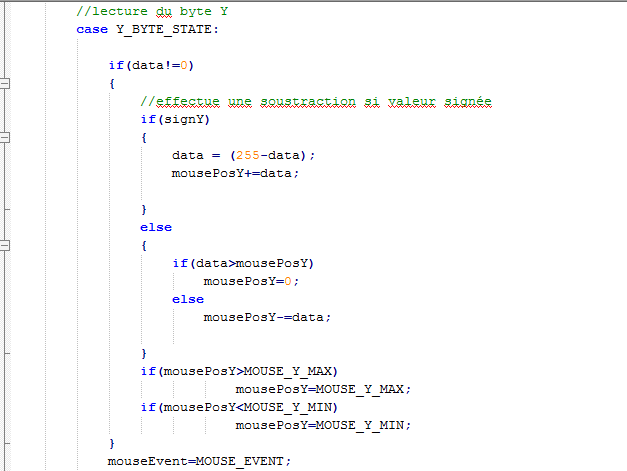


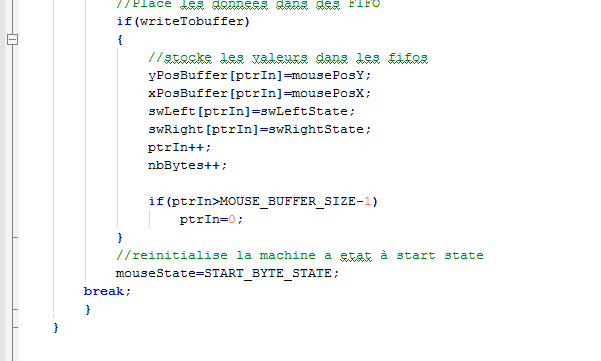


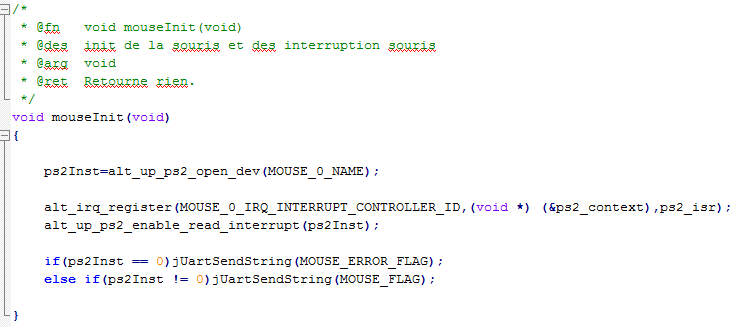


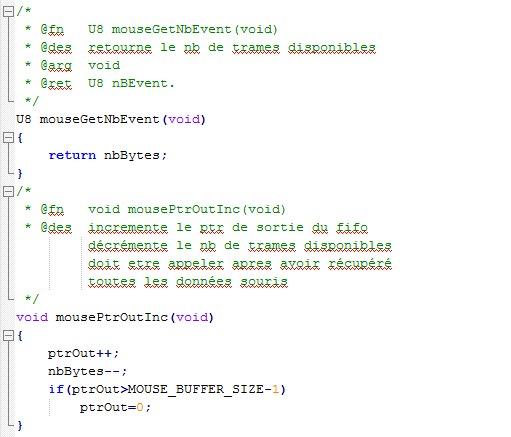


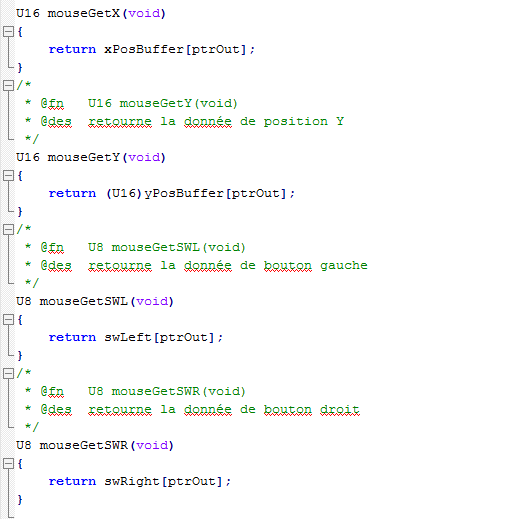




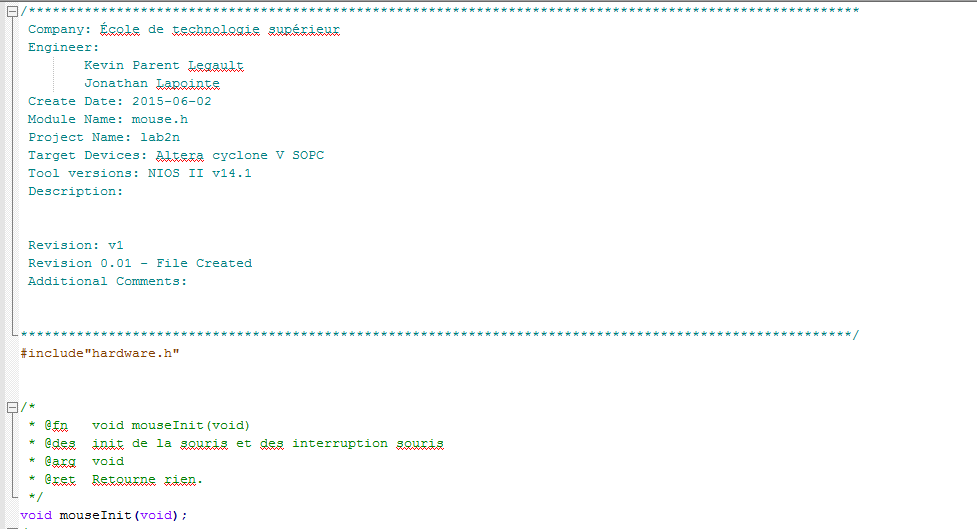


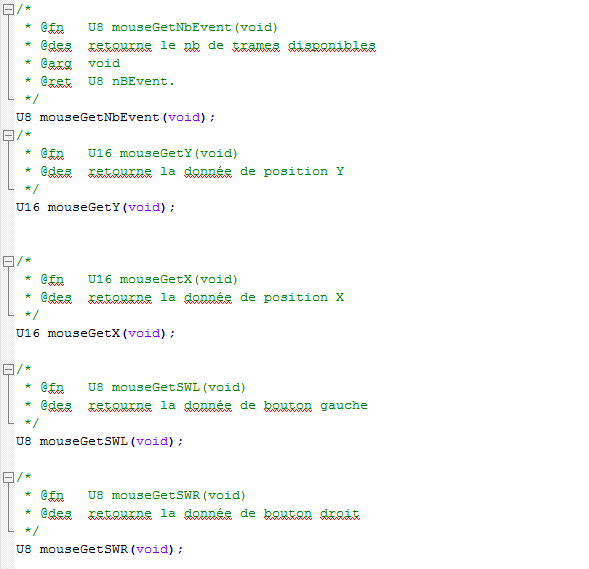


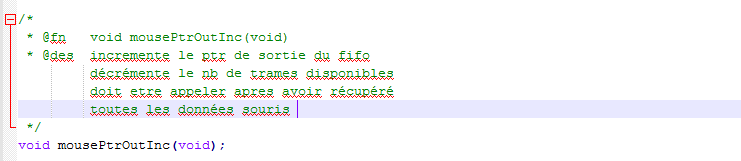




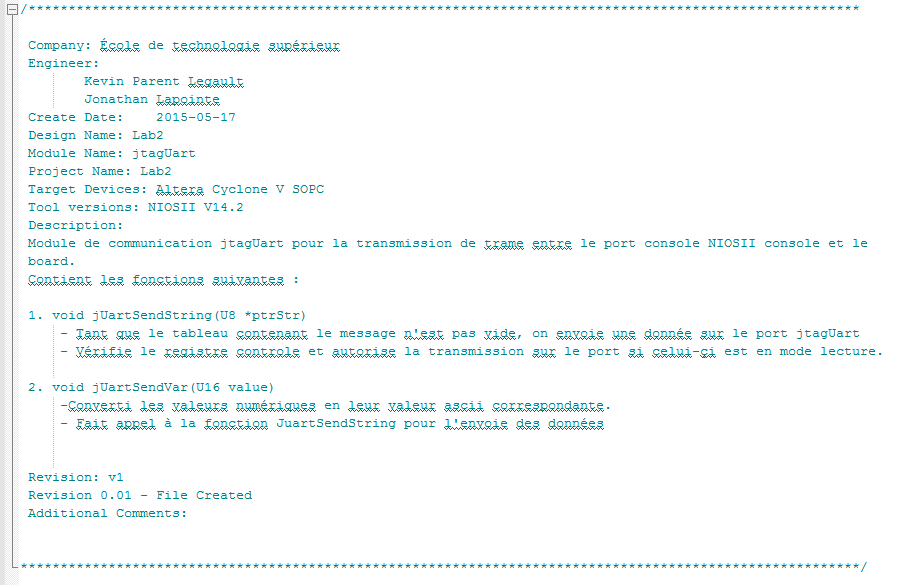
### Mouse.h

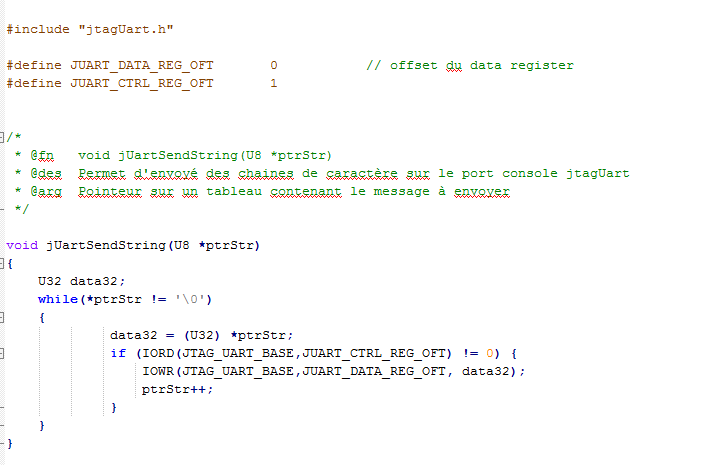


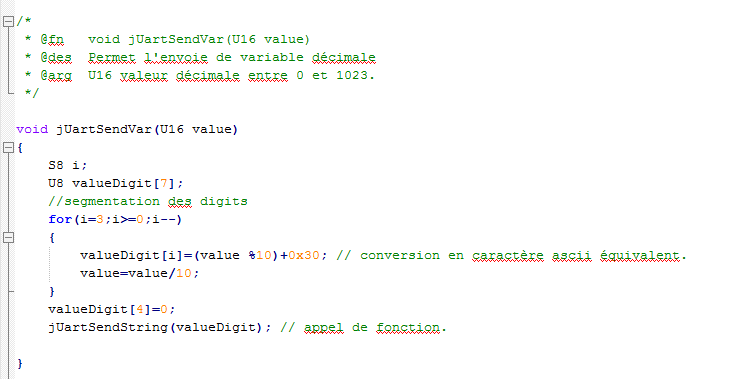




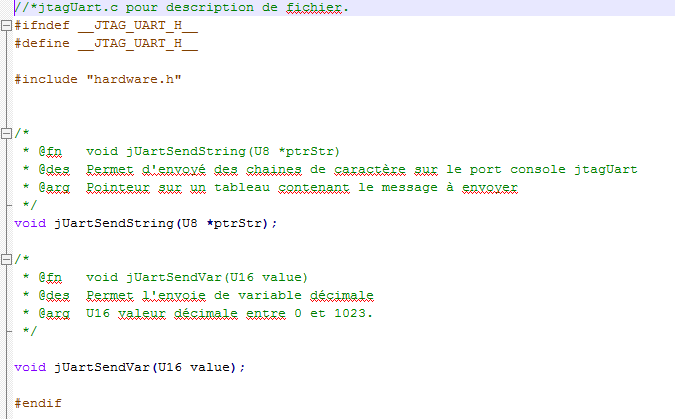
### JtagUart.c



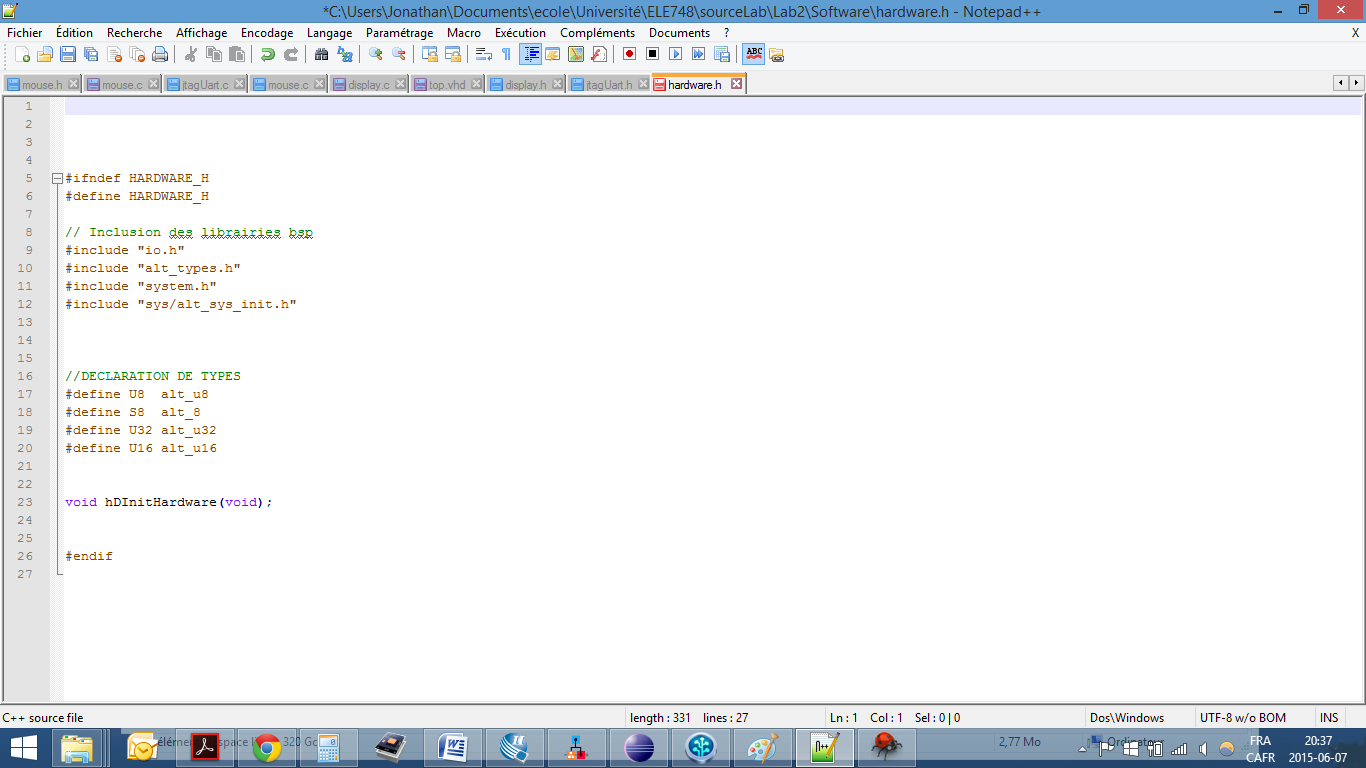




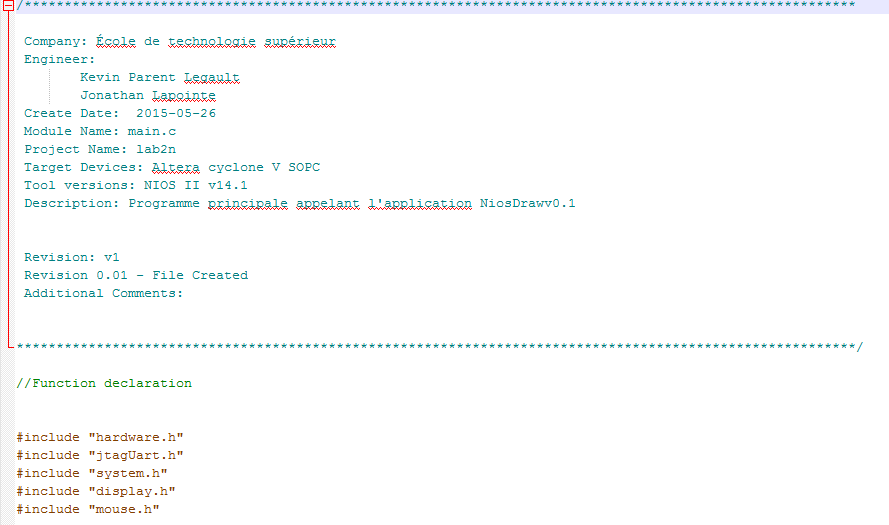
### jtagUart.h

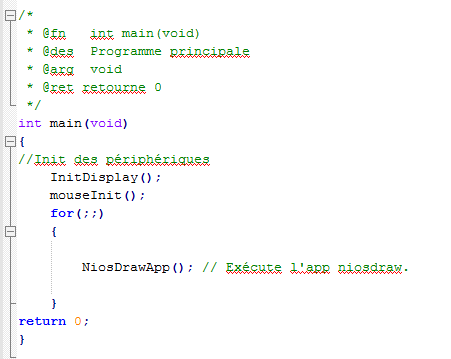


### Hardware.h



### Main.c





# Discussion

Ce laboratoire, servant à nous initier avec les périphériques avancés de même que les librairies HAL, a été difficile à réaliser sur plusieurs aspects. En effet, lors de l’implantation de l’architecture physique, un des fichiers Qsys était corrompu ce qui nous a obligé à préparer un environnement de développement sur nos ordinateurs personnels ayant ainsi un impact considérable sur l’échéancier fixé préalablement. De plus, la mise à jour des librairies HAL et les documents référentiels pas à jour, à augmenter de façon considérable le niveau de difficulté concernant la réalisation du projet. De plus, lors de notre phase de validation on a remarqué que la souris ne s’activait pas tout le temps, et ce, même si le périphérique est bel et bien initialisé. Ce problème persiste et on n’a toujours pas trouvé la cause de cette défaillance. En contrepartie, la plupart des spécifications exigées dans le devis ont été rencontrées. En effet, on pense que notre système répond de façon honnête aux exigences. Finalement, on a pu observer qu’avec les modules« IP cores» il était possible d’intégrer et de rapidement concevoir un système complexe.

# Conclusion

En terminant, ce laboratoire n’a pas été facile à réaliser à cause des outils de développement. Nous avons, cependant, beaucoup appris à les connaître et surtout à nous débrouiller par nous même pour les faire marcher. Il s’agissait d’une expérience enrichissante. Ce laboratoire nous permettra probablement d’être plus à l’aise avec les outils pour réaliser le projet qui suivra.