

Service des Travaux Pratiques Électronique

Tutoriel du logiciel Quartus d'Altera

version du 23 octobre 2012





Table des matières

1	Créer un projet	2
	1.1 Création d'un dossier Windows spécifique à votre projet	2
	1.2 Lancer le logiciel Quartus	2
	1.3 Créer un projet sous Quartus avec l'assistant de création	2
2	Créer le ou les fichiers décrivant votre projet	3
	2.1 Création de fichier sous forme schématique	3
	2.2 Création de fichier sous forme de fichier VHDL	5
3	Schéma associé à une fonction	5
	3.1 Création de schéma associé à une fonction sous forme de fichier VHDL	5
	3.2 Utilisation du schéma associé à une fonction	6
4	Travailler avec plusieurs fichiers	6
	4.1 Travail avec plusieurs fichiers	6
	4.2 Ajouter ou éliminer des fichiers du projet	7
5	Compiler le projet	7
6	Configurer le composant	7
7	Tester le fonctionnement du composant configuré	8
8	Compléments	8
Ŭ	8.1 Simulation	8
	8.2 Quelques astuces	11
In	terfaçage carte d'acquisition - Maquette DE0	12
Т	ableau des noms des broches sur la maquette DEO	16

L'environnement de développement permettant de programmer les composants reconfigurables d'Altera se nomme Quartus. Il permet de réaliser l'intégralité des étapes permettant de configurer un composant programmable. Ce tutoriel est écrit pour la version 9.0 de Quartus avec laquelle fonctionne la carte de développement DEO.

Créer un projet 1

Création d'un dossier Windows spécifique à votre projet 1.1

Pour chacun des projets ou des versions de projets sur lesquelles vous allez travailler, il est très fortement conseillé de créer un dossier Windows spécifique, dans lequel seront sauvegardés le projet et les différents fichiers créés. On veillera donc à changer de dossier dès qu'on change de projet ou de version de projet.

1.2Lancer le logiciel Quartus

Lancez le logiciel Quartus II 9.0 Web Edition.

1.3Créer un projet sous Quartus avec l'assistant de création

- → Quartus fonctionnant par projet il est nécessaire de commencer par créer un projet à chaque début de conception.
- \rightsquigarrow Entrer ensuite dans le Project Wizard le nom choisi pour votre projet :



- \rightsquigarrow Choisir le FPGA présent sur votre carte en sélectionnant :
 - La famille de FPGA : Cyclone III
 - Le modèle : EP3C16F84C6
- \rightsquigarrow Cliquer 2 fois sur Next > Finish puis sur Le projet est créé.



15408 15408

15408

15408 15408

1.2

1.2V 1.2V

169

347 347

516096 516096

51609

516096 516096

112

112

112 112

×

•

•

•

٠

EP3C16F256I7 EP3C16F484A

EP3C16F484C

EP3C16E484C

C16EA8AC

2 Créer le ou les fichiers décrivant votre projet

Un projet est généralement constitué de plusieurs fichiers décrivant chacun une ou plusieurs fonctions numériques. Parmi les possibilités proposées par Quartus pour créer des fichiers, nous allons en voir deux :

- 1. Création sous forme schématique : on parle alors de *Block Diagramm/Schematic File* (extension .bdf des fichiers), traitée au paragraphe 2.1.
- Création sous forme textuelle en utilisant un langage de programmation de circuits : Langage VHDL (extension .vhd) ou bien langage Verilog (extension .v, langage non enseigné à l'IOgs), voir le paragraphe 2.2.

SOPC Builder System

AHDL File Block Diagram/Schematic File EDIF File State Machine File

Design Files

2.1 Création de fichier sous forme schématique

- \rightsquigarrow File>New (ou Ctrl N) ou icône lectionner le type de fichier : Block Diagram/Schematic file
- SystemVerilog HDL File Tol Script File Verilog HDL File Verilog HDL File nory Files kadecimal (Intel-Format) File Memory Initialization File Verification/Debugging Files In-System Sources and Probes File Logic Analyzer Interface File SignalTap II Logic Analyzer File Vector Waveform File - Other Files AHDI Include File Block Symbol File Chain Description File Synopsys Design Constraints File Text File ОК Cancel ĒD-**. A** Ð 🛛 ٦ ٦ ٦ • €, -60 đå ∆ 🗧 4₿

X

- $\overset{\sim}{\longrightarrow} \text{Dans le fenêtre qui s'ouvre, sélectionner l'outil de sélection de composants D puis, dans la fenêtre Symbol, pour accéder par exemple, à une porte ET à 2 entrées, il faut descendre ensuite dans le menu de sélection de composants en choisissant : <math>C:/altera/quartus/librairies/primitives/logic.$
- $\rightsquigarrow\,$ Sélectionner le composant, puis cliquer sur Ok.
- ∽→ On bascule alors sur le schéma; chaque clic souris dessine une nouvelle porte; on sort de ce mode par la touche Echap (ou Esc).
- → Pour tracer des connexions, on sélectionne l'outil (Orthogonal Node Tool), et on relie les 2 terminaisons concernées entre elles :

→ Il faut maintenant dessiner les entrées/sorties de la fonction logique (les in ou les out de la description VHDL). Pour cela sélectionner à nouveau l'outil D puis, dans la fenêtre Symbol, choisir le chemin C :/altera/quartus/librairies/primitives/pin et choisir input pour les entrées ou output pour les sorties. Reprendre les mêmes opérations que pour les portes pour la fin de ce schéma. On obtient alors :



À ce niveau,

- soit vous souhaitez utiliser cette fonction dans d'autres fichiers (voir la commande component en VHDL, et la notion d'instanciation), dans ce cas lire le paragraphe 3.
- soit vous voulez relier votre fonction numérique à des broches physiques de la carte DEO. C'est l'objet de la suite de ce paragraphe.

Afin de faire correspondre les entrées/sorties avec les signaux de la maquette il faut choisir leur nom se sorte qu'ils correspondent à des broches physiques de la carte. Pour cela, il existe un fichier descriptif de la carte, DEO_Pins_Assignments.xls. Ce fichier décrit les 484 broches du composant (d'où son nom) et nous permet de sélectionner facilement les broches de la carte. Le tableau 2 page 16

indique les noms des broches et leur correspondance sur la maquette DEO.

- \rightsquigarrow Ce fichier est dans le répertoire /DE0 pour TPELEC/ que vous devez recopier à partir du serveur dans vos documents.
- → Pour relier la 1ère entrée du schéma au bouton poussoir 0, on double-clique sur le *pin name* du connecteur d'*Input*, puis dans la fenêtre apparaissant on renseigne le nom en indiquant ORG_BUTTON[0]. Puis on valide.
- → Pour relier la 1ère entrée du schéma au Switch 8, on renseigne le nom en indiquant SW[8].
- → Pour relier la sortie du schéma à la Led verte
 3 on renseigne le nom du connecteur Output
 en indiquant LEDG[3].

I	Pin Properties							
	General Format							
	To create multiple pins, enter a name in AHDL bus notation (for example, ''name[30]''), or enter a comma-separated list of names.	_						
	Pin name(s): ORG_BUTTON[0]							
	Default value: VCC							

Pour permettre au logiciel de faire la correspondance entre les noms des broches de la carte, et les broches du composant. Il faut importer le fichier DEO_Pins_Assignments dans le projet.

→ Pour cela, choisir Assignments > Import Assignments, puis aller sélectionner le fichier à inclure.

 → La conception de votre 1er circuit sous forme graphique se termine ici. Reste à compiler le projet puis à la télécharger sur la carte. Vous pouvez donc, pour ce 1er circuit, passer directement au paragraphe 5.



2.2 Création de fichier sous forme de fichier VHDL

- → File>New (ou Ctrl N) ou icône □, puis sélectionner le type de fichier : VHDL File.
- \rightsquigarrow Tapez le code VHDL.
- → Pour associer les Entrées /sorties de votre code VHDL à des broches physiques du composant, le plus simple est de créer un schéma bloc associé à votre code VHDL (voir paragraphe 3)
- → Puis de relier les broches de ce schéma bloc aux broches physique comme indiqué dans le paragraphe
 2.1. Il vous faut donc d'abord passer à la partie 3.

3 Schéma associé à une fonction

Quartus proposant une interface graphique, il est utile de pouvoir associer n'importe quelle fonction à un schéma; cela s'applique indifféremment aux fonctions décrites sous forme de schéma, ou sous forme de code VHDL. On peut ensuite associer entre elles les nouvelles fonctions décrites sous formes de schémas, et ainsi de suite. On voit donc apparaître la notion de conception hiérarchique.

3.1 Création de schéma associé à une fonction sous forme de fichier VHDL

- → Ouvrir le fichier (schéma ou code VHDL)
- → Sélectionner Create/Update >
 - Create Symbol File for current File, voir la figure ci-contre (nb : pour pouvoir créer un schéma associé à un fichier, le fichier doit avoir été sauvegardé)
- → Sauvegarder le schéma (fichier .bdf) ainsi créé dans le dossier du projet en lui donnant un nom explicite différent du nom du fichier VHDL ainsi que de celui de l'entité (*entity*).

S Qui	artus II - C:/Users/je	in-marie/Do	cuments/m	es doc	S/ 1P C11,	PROV	NUL/ESSAJ	4/CS	5414	- ES	Sd
📸 Fil	e Edit View Project	Assignments	Processing	Tools	Window	Help					
[C] <u>N</u> ew	Ctrl+N	Ca Es	sai4			-	X	2	3	R
- 📜 🖬	Open	Ctrl+O	-			:1					
Pro	Close	Ctrl+F4		•	- i 🔤			:::		:::	÷
브			- <u> </u>		_			111		:::	÷
4 🔀	New Project Wizard					A		111	111	:::	1
l 🧟	Open Project	Ctrl+J			T.	n		111		:::	1
	Convert MAX +PLUS TU	Project						111		:::	÷
	Convertiniax mgos III	-roject				ן ר				:::	1
	Save Projec <u>t</u>				l ~	4		111	:::	:::	1
	Close Project					-		111		:::	÷
			<u> </u>		1 ,1	Ξ,		111		:::	1
_ <u>4</u> 🖿	Save	Ctrl+S				- #4		:::		:::	1
_	Save As		—							:::	÷
Tas	Save Current Report S	ection As		<u> </u>	<u>×</u> 4	4		111			÷
Flov					- L			111		:::	1
To	Eile Properties			_	- As			:::		:::	÷
						-					
	Create / Update	•	Create	HDL Des	sign File fo	r Currer	nt File				
	Export		Create	Symbol	Files for C	urrent F	ile				
	Convert Programming F	iles	Create	AHDL In	dude Files	for Cu	rent File				

3.2 Utilisation du schéma associé à une fonction

- ∽→ Créer un nouveau fichier Block Diagramm/Schematic (ou ouvrir un fichier existant)
- \rightsquigarrow Sélectionner l'outil Symbol Tool $\stackrel{\mathbf{D}}{ imes}$
- → Ouvrir le dossier *Project* et sélectionner le composant par son nom
- \rightsquigarrow Se servir ensuite de ce composant comme d'un composant standard dans des *Block Dia*gramm/Schematic de plus haut niveau.
- → On peut par exemple associer les broches physiques de la carte au Entrées/sorties d'un code écrit en VHDL.
- → Ne pas oublier de désigner le fichier "schéma"

 (.bdf) comme étant le fichier de plus haut niveau. Pour cela, sélectionner dans le Project Navigator le fichier de plus haut niveau;
 faire un clic droit et choisir Set as Top-Level Entity comme sur la figure ci-contre.



4 Travailler avec plusieurs fichiers

Cette partie ne concerne pas vos tous premiers développements.

4.1 Travail avec plusieurs fichiers

Pour le développement d'un projet, on est très rapidement amené à avoir plusieurs fichiers. Quartus nous permet d'utiliser des fichiers de tous types (graphiques ou textuels). Pour les associer entre eux, le plus simple est d'associer chaque fichier textuel à un schéma puis d'associer les schémas entre eux, et enfin de nommer les entrées-physiques sur le schéma global avec les noms adéquats (voir tableau 2 page 16). Le mode opératoire est donc :

- 1. Créer les différents fichiers
- 2. Associer chaque fichier à un schéma
- 3. Associer les schémas entre eux
- 4. Désigner le fichier de plus haut niveau (Set as Top-Level Entity)

4.2 Ajouter ou éliminer des fichiers du projet

- → Pour ajouter (ou d'ailleurs éliminer) des fichiers au projet, on sélectionne Project
 > Add/Remove Files in Project puis on choisit les divers fichiers à inclure.
- → Pour éliminer des fichiers du projet, on peut aussi aller dans la fenêtre de Project Navigator, sélectionner le fichier, puis Remove File from Project.



<mark>«, Quartus II - C:/Users/jea</mark>n-marie/Docume

A noter qu'un fichier éliminé du projet n'est nullement effacé de l'ordinateur et qu'on pourra le ré-inclure dans ce projet ou dans un autre ultérieurement.

5 Compiler le projet

La compilation va générer la configuration du composant programmable qui permettra la réalisation concrète du projet. Pour cela :

-	sélectionner	l'icône idoine	(la flèche	violette)	dans l	le menu	

📙 🗅 🚅 📕 🎒 🎒 🐰 🖻 🛍 🗠 🗠 projet1	- 😵 🧐 🖉 🚬	1 🕨 🔁 💌	🕚 ≿ 🗶 🍉 👱 🎯
· · · · · · · · · · · · · · · · · · ·			·

- ou dans la barre de menu, choisir *Processing* > Start Compition,
- ou encore en tapant Ctrl L.

On peut ne pas lire le rapport de compilation dans la fenêtre de Messages (en bas de la fenêtre **Quartus**) lors des premières programmations. Si la compilation n'est pas couronnée de succès, il va falloir trouver et corriger les erreurs dans les fichiers sources. Sinon, on peut passer àăla programmation du composant, paragraphe 6.

Remarques sur la fenêtre de *messages* et celle de *Flow Summary* : de façon générale, un grand nombre d'informations vous sont données sur le bon (ou mauvais déroulement de la compilation), mais dans un 1er temps leur lecture n'est pas indispensable. Vous y verrez un nombre important de warningsă : ils ne sont pas forcément critiques. Enfin la fenêtre de *Flow Summary* vous donne un résumé sur les ressources occupées par votre fonction sur votre composant.

6 Configurer le composant

Pour configurer le composant afin d'obtenir le fonctionnement décrit par le projet, il faut impérativement :

– que la maquette DEO soit reliée à l'ordinateur par câble USB,

- qu'elle soit sous tension (bouton rouge),

- que le commutateur RUN / PROG soit positionné sur RUN

Il est de plus préférable d'avoir testé la validité de la chaîne de développement ainsi que celle de la connexion USB, à l'aide du programme de test fourni, *DEO Control Panel* ou par la programmation de la carte à l'aide d'un programme déjà réalisé et présent sur DEO pour TPELEC\New Files, de0_debounce_cnt.

Pour configurer le composant de la maquette DEO :

→ sélectionner l'icône de programmation,

] 🗅 😅 🖬 🎒 / 🖧 🗈 🛍 🛤 🗠 🖓 [projet1	💽 💢 🖉 🥙 🗞	🖻 🕨 😽 Խ	🏷 🕐 📩	📎 👱 📀

 \rightsquigarrow ou bien Sélectionner Tools > Programmer

 \rightsquigarrow La fenêtre de "programmation" s'ouvre alors.

🖞 Quartus II - c:/altera/90/quartus/projet1 - projet1 - [projet1.cdf]									- 🗆 ×				
File Edit Processin	File Edit Processing Tools Window												
La Hardware Setup No Hardware Node: JTAG Progress: 0%													
🔲 Enable real-time IS	Enable real-time ISP to allow background programming (for MAX II devices)												
Mart Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP		
🖬 Stop	projet1.sof	EP3C16F484	000C78D6	FFFFFFF	•								

 \rightsquigarrow Cliquer sur Hardware Setup

- Hardware Setup
 Hardware Setup
 Hardware Settings
 JTAG Settings
 Select a programming hardware setup to use when programming devices. This programming
 hardware setup applies only to the current programmer window.
 Currently selected hardware:
 USB-Blaster (USB-0)

 Available hardware items:
 Hardware
 Server Port
 Add Hardware...
- \rightsquigarrow Enfin, cliquer sur \bowtie Start pour lancer la configuration du circuit de la maquette DE0.

7 Tester le fonctionnement du composant configuré

Pour valider la conception, il faut tester le comportement électronique et le comparer au cahier des charges. En cas de fonctionnement non-conforme, il faudra (sûrement) revenir à la conception des fonctions du projet.

8 Compléments

8.1 Simulation

La simulation est l'étape qui vous permet de valider un design avant de le programmer sur la carte. Il est en effet souvent bien plus facile de trouver une erreur sur des résultats de simulation plutôt que sur le circuit. De plus, programmer une configuration de circuit qui ne correspond pas à ce que vous deviez concevoir est une perte de temps. En dehors des cas les plus simples, il faut donc simuler après avoir compilé et avant de programmer.

Pour simuler, vous allez devoir définir précisément le type de simulation voulu, la durée de cette simulation, les entrées et les sorties que vous voulez observer.

Il faudra aussi choisir les signaux qu'on veut observer, et imposer une configuration aux entrées. Cela consiste à définir les "vecteurs de tests", ce que nous ferons en configurant un fichier *Vector Wave File* (extension .vwf). Enfin se pose la question de savoir si les tests sont exhaustifsă : c'est un problème qui peut être (très) compliqué et qui n'est pas abordé dans ce document.

Configurer le simulateur

- → Lancer le simulateur par le menu Processing > Simulator Tool,
- → Compléter les champs Simulation mode et End Simulation. Au niveau des modes de simulation, Functionnal correspond à une simulation avec des portes idéales, et Timing à une simulation en tenant compte des retards technologiques. On vous conseille de choisir ce dernier mode.
- \rightsquigarrow Il va falloir créer un fichier de vecteur de tests. On va pour cela, aller dans File > New (ou Ctrl N) et sélectionner Vector Wave File.

🗐 Simulator Tool	
Simulation mode: Timing	Generate Functional Simulation Netlist
Simulation input:	Add Multiple Files
Simulation period C Run simulation until all vector stimuli are used C End simulation at: 5000.0 Ins	



→ Dans la nouvelle fenêtre, clic droit sur Name pour ajouter des signaux, puis sélectionner Insert > Insert Node or Bus et sélectionner le menu Node Finder.



Insert Node o	r Bus	X				
Name:		OK				
Туре:	INPUT 💌	Cancel				
Value type:	9-Level	Node Finder				
Radix:	Binary 💌					
Bus width:	1					
Start index:	0					
Display gray code count as binary count						

 \leadsto On ajoute les signaux qui nous intéressent en les faisant apparaître par double-clic (ou encore, par sélection puis >).

Node Finder					×
Named: x	•	Filter: Pins: assigned	Customize	List	0K
Look in: DiviPar8192bdf			Include subentities	Stop	Cancel
Nodes Found:			Selected Nodes:		
Name	Assignments	Туре	Name	Assignments T	
CLOCK_50	PIN_G21	Input	DiviPar8192bdllCLOCK_50	PIN_G21 Ir	
@ GPI01_D[0]	PIN_AA20	Output	DiviPar8192bdljGPI01_D[0]	PIN_AA20 0	
@ GPI01_D[1]	PIN_AB20	Output	DiviPar8192bdljGPI01_D[1]	PIN_AB20 0	
IIII SW[0]	PIN_J6	Input	DiviPar8192bdf/SW[0]	PIN_J6 Ir	

L'interface du simulateur s'ouvre alors :

n Wave	n Waveform1.vwf*								
Master Time Bar: 16.85 ns 💶 Pointer: 6.0 ns Interval: -10.85 ns Start: End:									
x x		Name	Val 16	0 ps 1	0.0 ns 16.8	20.0 ns 35 ns -1	30.0 ns		
	⊉ 0 ⊚1	CLOCK_50 GPIO1 DI01	B O B X		*****	*****	*****		
	2 2 3	GPIO1_D[1] SW[0]	B X B O	<u> </u>	*****	*****	******		
) C) C									

→ Il faut maintenant imposer une valeur pour chaque entrée. On sélectionne un signal, puis les icônes de la colonne de gauche passent en bleu, et on impose des valeurs, ou bien par Clic Droit, on choisit dans le menu contextuel.



Quelques remarques : – Pour imposer un 0 ou un 1 pendant juste une durée, sélectionner cette durée dans le panneau de droite puis cliquer sur l'icône associée dans le menu.

- Pour l'horloge, sélectionner le signal voulu puis cliquer sur l'icône de signal périodique.

Quelques raccourcis clavier du simulateur :
 Ctrl W : affiche tout l'intervalle de simulation
 Ctrl Espace : Zoom
 Ctrl Shift Espace : Dézoom

Lancer la simulation

- → Lancer la simulation par l'icône Start du Simulator Tool
- \rightsquigarrow Pour observer les résultats, cliquer sur Report
- → Pour modifier le fichier de vecteur de tests, cliquer sur les résultats, cliquer sur Open.



8.2 Quelques astuces

Projets

Lorsqu'on n'est pas directement au démarrage de Quartus, pour créer un nouveau projet, on peut sélectionner le menu File > New Project Wizard.

Pour ouvrir un projet existant, on peut faire File > Open Project, ou bien Ctrl J. Pour fermer un projet ouvert, faire File > Close Project.

Commandes de schéma

Pour déplacer un symbole, cliquer dessus et le déplacer. Pour déplacer l'ensemble du schéma dans la fenêtre active, faire *Ctrl A* et déplacer Pour copier un symbole, procéder par *Ctrl C* puis *Ctrl V*. Pour détruire un symbole, procéder sélection puis *Suppr*. Pour modifier le nom d'un signal, double cliquer sur le nom puis modifier.

Notion de bus

Les signaux numériques se présentent souvent sous forme de bus (associations de plusieurs fils par exemple lorsqu'il représentent un mot numérique. Il est aussi parfois judicieux d'associer plusieurs signaux booléens et de créer un bus.

Sous l'éditeur de schéma de Quartus, on trace les bus avec l'outil

Pour nommer ces bus on utilisera par exemple : $\texttt{HEX2_D[6..0]}$ pour nommer les 6 fils du bus d'entrée d'un des afficheurs 7 segments (donc de sortie du FPGA), ou encore SW[2..0] pour le bus associé à 3 switchs. On choisit la taille souhaitée pour le bus.

Du schéma au VHDL

A partir d'un schéma, Quartus peut générer un code VHDL. Pour cela, dans l'éditeur de schéma, sélectionner le menu Create/Update > Create HDL Design: Le code VHDL sera ensuite généré automatiquement.

Interfaçage carte d'acquisition - Maquette DEO

Bornier partagé

Le bornier permet de relier la carte d'acquisition NI USB 6009 au monde physique par l'intermédiaire de câbles BNC. Ce même bornier va aussi être utilisé pour relier la carte DEO au monde physique.

On a donc 3 modes de fonctionnement :

- 1. DEO reliée au bornier (carte d'acquisition déconnectée)
- 2. carte d'acquisition reliée au bornier (carte DEO déconnectée)
- 3. carte d'acquisition et DEO reliées aux bornier, donc reliées ensemble.

Tout dépend de la façon dont vous avez choisi de relier les connecteurs en nappe.

Au niveau de la DEO, 2 connecteurs d'extension sont disponibles, totalisant jusqu'à 72 entréessorties possibles :

(GPIO 0) J4	(GPIO 1) J5
[AB12] GPIO0_CLKIN0 1 2 - GPIO0_D0 [AB16]	[AB11] GPI01_CLKIN0 - 0 1 2 - GPI01_D0 [AA20]
[AA12] GPIO0_CLKIN1 - 3 4 - GPIO0_D1 [AA16]	[AA11] GPIO1_CLKIN1 - 3 4 - GPIO1_D1 [AB20]
[AA15] GPIO0_D2 - 5 6 - GPIO0_D3 [AB15]	[AA19] GPIO1_D2 5 6 GPIO1_D3 [AB19]
[AA14] GPIO0_D4 — 7 8 — GPIO0_D5 [AB14]	[AB18] GPIO1_D4 🔶 7 8 🕞 GPIO1_D5 [AA18]
[AB13] GPIO0_D6 9 10 GPIO0_D7 [AA13]	[AA17] GPIO1_D6 9 10 GPIO1_D7 [AB17]
5V 🛑 11 12 🛑 GND	5V 🛑 11 12 🌰 GND
[AB10] GPIO0_D8 - 13 14 - GPIO0_D9 [AA10]	[Y17] GPIO1_D8 - 13 14 - GPIO1_D9 [W17]
[AB8] GPIO0_D10 - 15 16 - GPIO0_D11 [AA8]	[U15] GPIO1_D10 - 15 16 - GPIO1_D11 [T15]
[AB5] GPIO0_D12 💛 17 18 🔂 GPIO0_D13 [AA5]	[W15] GPIO1_D12 💛 17 18 🖵 GPIO1_D13 [V15]
[AB3] GPIO0_CLKOUT0 - 19 20 - GPIO0_D14 [AB4]	[R16] GPIO1_CLKOUT0 9 20 G GPIO1_D14 [AB9]
[AA3] GPIO0_CLKOUT1 - 21 22 - GPIO0_D15 [AA4]	[T16] GPIO1_CLKOUT1 — 21 22 - GPIO1_D15 [AA9]
[V14] GPIO0_D16 - 23 24 - GPIO0_D17 [U14]	[AA7] GPIO1_D16-C 23 24 - GPIO1_D17 [AB7]
[Y13] GPIO0_D18 - 25 26 - GPIO0_D19 [W13]	[T14] GPIO1_D18 - 25 26 - GPIO1_D19 [R14]
[U13] GPIO0_D20 - 27 28 - GPIO0_D21 [V12]	[U12] GPIO1_D20 - 27 28 - GPIO1_D21 [T12]
3.3V 🔴 29 30 🛑 GND	3.3V 🛑 29 30 🌰 GND
[R10] GPIO0_D22 - 31 32 - GPIO0_D23 [V11]	[R11] GPIO1_D22
[Y10] GPIO0_D24 - 33 34 - GPIO0_D25 [W10]	[U10] GPIO1_D24
[T8] GPIO0_D26 - 35 36 - GPIO0_D27 [V8]	[U9] GPIO1_D26 - 35 36 - GPIO1_D27 [T9]
[W7] GPIO0_D28 → 37 38 → GPIO0_D29 [W6]	[Y7] GPIO1_D28 - 37 38 - GPIO1_D29 [U8]
[V5] GPIO0_D30 - 39 40 - GPIO0_D31 [U7]	[V6] GPI01_D30 - 39 40 - GPI01_D31 [V7]

FIGURE 1 – Brochage des connecteurs d'extension de la carte DEO

En ce qui nous concerne, nous ne nous servirons que du connecteur GPIO1, et seulement d'un sous-ensemble de ce connecteur :



FIGURE 2 – Connecteurs d'extension de la carte ${\tt DEO}$

GPIO1 vers NI USB		NI USB vers GPIO1	
GPIO1	NI 6009	NI 6009	GPIO1
	port 0	port 0	
		P0.0	_
D[0]	P0.3	P0.1	CLK_OUT[0]
D[1]	P0.5	P0.2	CLK_OUT[1]
D[2]	P0.6	P0.3	D[0]
D[3]	P0.7	P0.4	CLK_IN[1]
CLK_OUT[0]	P0.1	P0.5	D[1]
CLK_OUT[1]	P0.2	P0.6	D[2]
CLK_IN[1]	P0.4	P0.7	D[3]
	port 1	port 1	
D[4]	P1.0	P1.0	D[4]
D[5]	P1.1	P1.1	D[5]
D[6]	P1.2	P1.2	D[6]
D[7]	P1.3	P1.3	D[7]

 $TABLE\ 1-Connexions\ entre\ la\ carte\ DEO\ et\ la\ carte\ d'acquisition\ NI\ USB\ 6009\ sur\ le\ bornier\ des\ TPs.$

Pour faciliter les connexions, le bornier à une face avant nommant explicitement les entrées/sorties, avec la convention :



FIGURE 3 – Bornier

Logiciel Measurement & Automation Ce logiciel permet d'acquérir ou de générer des signaux analogiques ou numériques sur la carte d'acquisition NI USB 6009

- La carte comprend comprend 4 blocs bien distincts :
- un bloc d'entrées analogiques : AI, au nombre de 8,
- un bloc de sorties analogiques : AO, au nombre de 2,
- un bloc d'entrées/sorties numériques : Port 0 (8 E/S) ou Port 1 (4 E/S) pouvant être configurées en entrée ou en sortie à la demande,
- un bloc E/S de compteur : PFI0

Une fois le logiciel lancé, cliquer sur « panneau de test », puis choisir dans un des 4 onglets (correspondant aux blocs cités ci-dessus, une (ou plusieurs) voie(s), et la (les) configurer.

Pour les entrées analogiques, noter le mode différentiel ou RSE (soit référencé à la masse), les valeurs maxi des entrées, et le mode (sur demande, en continu, ou fini). Pour le mode en continu fini, préciser la fréquence d'échantillonnage et le nombre de points à acquérir.

Pour les sorties analogiques, il n'y a que les limites à fixer.

Pour les entrées-sorties numériques, choisir le nom du port (0 ou 1), configurer bit à bit entrée ou sortie, sélectionner (pour les sorties) la valeur des différents bits.

15

.

Tableau des noms des broches sur la maquette DEO

Nom	Mode	Commentaires			
Broches du connecteur d'extension n°0 (GPIO0)					
(GPIO = General Purpose Input Output)					
GPIOO_CLKIN[1]	Entrée	Entrée n°1 d'un signal d'horloge			
GPIOO_CLKIN[0]	Entrée	Entrée n°0 d'un signal d'horloge			
GPIO0_CLKOUT[1]	Sortie	Sortie n°1 d'un signal d'horloge			
GPIOO_CLKOUT[0]	Sortie	Sortie n°0 d'un signal d'horloge			
GPIOO_D[i]	Bidirectionnelle	Données d'entrée ou de sortie n° i $(i$ va-			
		rie de 0 à 31).			
		Exemple : $GPIOO_D[0]$ pour le bit n°0.			
Broches du du connecteur d'extension n°1 (GPIO1)					
GPIO1_CLKIN[1]	Entrée	Entrée n°1 d'un signal d'horloge			
GPIO1_CLKIN[0]	Entrée	Entrée n°0 d'un signal d'horloge			
GPI01_CLKOUT[1]	Sortie	Sortie n°1 d'un signal d'horloge			
GPI01_CLKOUT[0]	Sortie	Sortie n°0 d'un signal d'horloge			
GPIO1_D[i]	Bidirectionnelle	Données d'entrée ou de sortie numéro i			
		(i varie de à 31).			
		Exemple : $GPIOO_D[0]$ pour le bit n°0.			
	Afficheurs 7 se	gments (de 0 à 3)			
7 LEDs (de 0)) à 6) par afficheur	r + 1 DP (dot point) par afficheur			
HEXj_D[i]	Sortie	LED n°i de l'afficheur 7 segments n°j			
		Exemples :			
		$\texttt{HEX0_D[6]} : \text{Led 6 de l'afficheur n°0},$			
		$HEX3_D[6]$: Led 6 de l'afficheur n°3			
HEXj_DP	Sortie	LED de Dot Point de l'afficheur 7 seg-			
		ments n°j			
		Exemple :			
		$HEX4_DP$: Led DP de l'afficheur n°4			
Switchs (de 0 à 9)					
SW[i]	Entrée	Switch n°i (i variant de 0 à 9)			
		Exemple :			
		SW[6] pour le switch n°6			
Boutons poussoirs (de $0 \ge 2$)					
ORG_BUTTON[i]	Entrée	Bouton poussoir n°i (i variant de 0 à 2)			
		Exemple :			
		ORG_BUTTON[2] pour le switch n°2			
Leds Vertes (de 0 à 9)					
LEDG[i]	Sortie	LED verte n'i (i variant de $0 \ge 2$)			
		Exemple :			
		LEDG[6] pour la LED n°6			

TABLE 2 – Noms des broches du circuit de la maquette ${\tt DE0}.$