

Installation de µClinux sur une carte Altera DE2-115 (3 h)

Objectifs

Ce TP vise à découvir comment configurer un système d'exploitation Linux pour un système embarqué. Vous aurez également à tester un pilote de périphérique opérant avec le système d'exploitation que vous avez configuré.

1 Prérequis

Pour réaliser ce TP, il vous faut au préalable terminer le TP de l'UCE « Concepts et développement », demandant de configurer votre carte Altera à l'aide du logiciel QSYS pour instancier un processeur SoftCore NiosII. Il vous faudra en particulier avoir à disposition un fichier .sopcinfo (nécessaire à la configuration du noyau μ Clinux) et un fichier .sof (nécessaire à la configuration du FPGA).

2 Installation et configuration de µClinux

Cette section explique comment installer et configurer la version μ Clinux utilisant la MMU à partir d'un environnement de développement Linux.

2.1 Clonage des dépôts git

L'ensemble des dépôts sont en principe déjà chargés sur votre PC dans le répertoire /home/administrateur/TP-uClinux.

Veuillez à ce que le répertoire contenant le noyau Linux 2.6 s'appelle linux-2.6 et qu'il soit situé au même niveau que le répertoire uClinux-dist, sous peine d'avoir ultérieurement une erreur lors de la configuration de Linux avec make menuconfig.

Vérifiez que la version disponible de make sur le PC soit bien 3.81 ou une version antérieure.

2.2 Préparation du fichier dts

Un *Device Tree* permet de décrire le matériel dans un système embarqué. Ses informations sont à fournir au noyau durant la configuration, c'est pouquoi il vous est demandé de le créer avant de démarrer la configuration proprement dite de μ Clinux. QSYS fournit un fichier .sopcinfo qui doit être converti vers un fichier .dts (*Device Tree*). Certaines informations ne pouvant pas être spécifiées dans le fichier .sopcinfo (comme les partitions flash, l'endroit depuis lequel le noyau doit être exécuté, les adresses MAC...), le fichier .dts peut prendre en compte des fichiers *boardinfo* additionels¹.

^{1.} Se reporter à http://www.alterawiki.com/wiki/Sopc2dts pour avoir de plus amples informations.

Lancez l'interface graphique de SOPC2DTS à l'aide de la commande suivante exécutée depuis le répertoire contenant le dépôt de SOPC2DTS :

cd <rep installation>/sopc-tools/sopc2dts²

java -jar sopc2dts.jar --gui

Sélectionnez « *Choose file* » puis ouvrez votre fichier .sopcinfo de manière à obtenir un écran qui ressemble à la figure 1. Vérifiez que votre composant LED soit présent à droite dans la liste des composants et ait une couleur verte en arrière plan.

Input Boardinfo Output Choose file /home/philgood/uClinux_DE2:115_GPA/uClinux4Nios2.sopcinfo Load file Input source Component list SOPC-Class altera_avalon_o <fi>-/mol version = "1.0" encoding="UTF-8"?> SOPC-Class altera_avalon_o sdram <fi><fi> SOPC-Class altera_avalon_o sdram <fi><fi><fi> SOPC-Class altera_avalon_o sdram <fi><fi><fi><fi><fi><fi><fi><fi><fi><fi< th=""><th>Sobest Sobest So</th><th></th><th></th></fi<></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi></fi>	Sobest So		
Close The Inditisy India Indi	Input Boardinfo Output Change file themetablicated (Clinux, DE2,115, CRA/(/Clinux/Mice2, consistent		Load file
input source		Component list	Load me
<pre>chain version= 1.0 encounters ''' encounters''''''''''''''''''''''''''''''''''''</pre>	apput source	Component list	
cl-restinue relation and commentations and contain additional information.) -> cl-restinue relation of modules and connections -> cyarameter name="AUTO_GENERATION_ID"> value>1407180151 value>14071	<pre>CEnsembleReport pame_"refigury/Nice2" kind="refigury/Nice2" version="1.0" fabric="0SVS"></pre>	SOPC-Class	altera_avalon_o 🚔
<pre><!-- 2014.08.04.21:22:32 ---> <!-- A collection of modules and connections--> <pre>cyarameter name="AUTO_GENERATION_JD"> <type>java.lang.integer</type> <ubr></ubr> <ubr></ubr></pre></pre>	Characteristic and the second seco	sdram	
<pre></pre>	2014.08.04.21:22:32	Туре	sdram
<pre>charameter name="AUTO_GENERATION_ID"></pre>	A collection of modules and connections	Crown	
<pre><value>1407180151</value></pre> <pre>value>1407180151</pre> <pre>setial soPC-Class altera_avalon_n it ag Type juart Group setial SOPC-Class altera_avalon_t it ag Croup timer Croup timer SOPC-Class altera_avalon_t it ag Croup timer Croup timer Croup timer SOPC-Class altera_avalon_t it ag Croup timer Croup tim</pre>	<pre><pre>cparameter name="AUTO_GENERATION_ID"></pre></pre>	Group	memory
<pre><value>140/180151</value> </pre> <pre>(derived>false </pre> <pre>(derived>false</pre> <pre>(jtag Type juart Type juart Group serial SOPC-Class altera_avalon_jt Type/avalang.String</pre> <pre>(timer Type timer Group timer SOPC-Class altera_avalon_ti Uart SOPC-Class altera_avalon_ti Uart Type uart Croup serial SOPC-Class altera_avalon_ti Uart Type uart SOPC-Class altera_avalon_ti Uart Type uart Group serial SOPC-Class altera_avalon_ti Uart Type uart Group serial SOPC-Class altera_avalon_ti Uart Croup serial SOPC-Class Croup serial SOPC-Class C</pre>	<type>java.lang.Integer</type>	SOPC-Class	altera_avalon_n
<pre></pre> pre> pre> pre> pre> <td><value>1407180151</value></td> <td>jtag</td> <td></td>	<value>1407180151</value>	jtag	
<pre></pre> <pre>chabled>false<!--/sible--> false</pre> <pre>false</pre> <pre>false<td><pre><gerved>false </gerved></pre></td><td>Туре</td><td>juart</td></pre>	<pre><gerved>false </gerved></pre>	Туре	juart
<pre> valid>true valid>true valid>true valid>true soPC-Class altera_avalon_it timer Type timer Group timer Group timer SoPC-Class altera_avalon_ti uart Type uart crue valid>true valid>tru</pre>	 visible>flue Aisible> 	Group	serial
<pre></pre> <pre> </pre> <td><pre><value< pre=""></value<></pre></td> <td>SOPC-Class</td> <td>altera avalon it</td>	<pre><value< pre=""></value<></pre>	SOPC-Class	altera avalon it
<pre><pre>charameter name="AUTO_UNIQUE_ID"> <type> ivalues-K/alue? <type>ivalues-K/alue? <derived>false</derived> <enabled> <value>visible>false <enabled> <visible>false <enabled> <visible>false</visible></enabled> <visible>false</visible></visible></enabled> </visible></enabled> </value></enabled> <td></td></type></type></pre>/parameter></pre>		SUP C-Class	altera_avalon_t
<pre>type>java.lang.String<ftype> timer value> derwed>false cenabled>true valid>true valid>true<!--</td--><td><pre><pre>arameter name="AUTO UNIQUE ID"></pre></pre></td><td>timer</td><td></td></ftype></pre>	<pre><pre>arameter name="AUTO UNIQUE ID"></pre></pre>	timer	
<pre></pre> <pre> Group timer SOPC-Class altera_avalon_ti uart Type uart Croup serial SOPC-Class altera_avalon_ti Uart Type uart Group serial SOPC-Class altera_avalon_ti Uart Type led Group led SOPC-Class led_component Type led Group led SOPC-Class led_component Type led Group led SOPC-Class led_component Type led SOPC-Class led SOPC-Class l</pre>	<type>java.lang.String</type>	Туре	timer
<pre><derwed>talse</derwed> <erwed>talse <erwed>talse <erwed>talse</erwed> <erwed>talse</erwed></erwed></erwed></pre> SOPC-Class altera_avalon_ti uart Type uart Type uart Type uart Group serial SOPC-Class altera_avalon_u Ied_component Type led Group led SOPC-Class led_component Type led Group led SOPC-Class led_component Type led Class Clear Patients to load. It does not aviet	<value></value>	Group	timer
<pre></pre>	<derived>false</derived>	SOPC-Class	altera avalon ti
<pre></pre> <pre></pre> <pre></pre> <pre></pre>	<enabled>true</enabled>	unet	
<pre>cyparameter> <parameter> <parameter> <parameter> <parameter> <parameter> <value>CYCLONEIVE</value></parameter></parameter></parameter></parameter></parameter></pre> //PAILUE> <pre>cyparameter</pre> <pre>//PAILUE</pre> <pre></pre>	<vsbieztruezt alise<="" td=""></vsbieztruezt>	uart	=
oparameter name="AUTO_DEVICE_FAMILY"> <tparameter name="AUTO_DEVICE_FAMILY"> <tparameter name="AUTO_DEVICE_FAMILY"> <tparameter name="AUTO_DEVICE_FAMILY"> <tp> SOPC-Class altera_avalon_u <derwed>false</derwed> led_component <derwed>false</derwed> rype led <value>CYCLONEIVE soPC-class altera_avalon_u <derwed>false wisible>false soPC-class led_component <</derwed></value></tp></tparameter></tparameter></tparameter>		Туре	uart
<type>java.lang.String</type> SOPC-Classaltera_avalon_uled_component <td< td=""><td><pre>>parameter name="AUTO DEVICE FAMILY"></pre></td><td>Group</td><td>serial</td></td<>	<pre>>parameter name="AUTO DEVICE FAMILY"></pre>	Group	serial
<pre></pre> <der <="" ce="" control="" er="" p=""> Ied_component Type Ied_component Type Ied_component SOPC-Class Ied_component Clear Performance Performance</der></der></der></der>	<type>java.lang.String</type>	SOPC-Class	altera_avalon_u
<pre><derived>false</derived> <pre><derived>false</derived></pre>// Type led Group led SOPC-Class led_component </pre>	<value>CYCLONEIVE</value>	led component	
<pre><cabled>true</cabled></pre> // If the component of the compone	<derived>false</derived>	Type	lod
Group Ted SOPC-Class Led_component	<enabled>true</enabled>	Type	led
Clear	<vrsible>false</vrsible>	Group	led
clear		SOPC-Class	led_component
Refusing to load. It does not exist	clear	L	
Relasing to load. It does not exist	Refusing to load . It does not exist		
Loading uClinux4Nios2.sopcinfo using SopcInfoSystemLoader	Loading uClinux4Nios2.sopcinfo using SopcInfoSystemLoader		
Loaded 44 components from sopc_components_altera.xml	Loaded 44 components from sopc_components_altera.xml		
Loaded 1 components from sope_components_tps.xml	Loaded 1 components from sope components fps.xml		
Loaded 3 components from sope components_ixp.xmi	Loaded 3 components from sope_components_ixp.xml		
social cloading done	soncinfo: Loading done		
No point of view specified. Trying to find one.	No point of view specified. Trying to find one.		
Found a cpu of type altera nios2 gsys named nios2 cpu	Found a cpu of type altera nios2 gsys named nios2 cpu		
dts memory section: No memory nodes specified. Blindly adding them all	dts memory section: No memory nodes specified. Blindly adding them all		

FIGURE 1 – Interface graphique de SOPC2DTS à l'ouverture du fichier .sopcinfo.

Il reste maintenant à configurer la manière dont est chargé le noyau au démarrage. Allez dans l'onglet « *Boardinfo* ». Sélectionnez « cpu : nios2_cpu » dans le champ « *Select master* ». Cliquez sur « *Apply* », ce qui devrait fixer le champ par défaut « *Kernel Bootargs* » à debug console=ttyAL0,115200 (voir Fig. 2). Cette option indique que l'OS est chargé depuis la SDRAM et définit l'UART comme port de communication par une console série.

Enfin, sélectionnez l'onglet « *Output* » et faites une sauvegarde dans un fichier .dts. La fin de ce fichier devrait avoir un contenu identique à celui de la figure 3.

^{2.} Le répertoire d'installation <rep installation > est en principe /home/administrateur/TP-uClinux.

Master informatique

⊗⊜ Sopc2DTS		
Input Boardinfo Output		
Choose file		Load file
General Ethernet Flash 12C SPI		
Point of view		
Select master	cpu: nios2_cpu	•
Memory nodes		
Memory nodes You can specify what nodes should be displayed in the memory section of the DTS file. You normally only select the memories you wish to run code from. I you don't select any node, all nodes will be added to the memory section.	Tam_tib Type Group SOPC-Class sdram Type Group SOPC-Class	onchipmem memory altera_avalon_onchip_memory2 sdram memory altera_avalon_new_sdram_con
Chosen Kernel Bootards	debug console-ttyAL0.115200	
Apply Re	vert Save	

FIGURE 2 – Affichage de l'onglet « Boardinfo » de SOPC2DTS.



FIGURE 3 – Contenu du fichier .dts généré.

2.3 Configuration de µClinux

Avant de configurer le noyau, modifiez la variable d'environnement **PATH** pour que le système de configuration trouve la chaîne de compilation :

export PATH=\$PATH:<rep installation>/toolchain-mmu/x86-linux2/nios2-linux-gnu/bin

Positionnez-vous dans le répertoire uClinux-dist et assurez-vous d'effacer les fichiers générés pour une configuration précédente avec les commandes make mrproper et make clean.

Lancez l'interface graphique de configuration de μ Clinux en exécutant la commande : make menuconfig.

Si vous avez une erreur d'exécution, vérifiez que la bibliothèque NCURSES est bien installée : sudo apt-get install libncurses5-dev

Le premier menu de make menuconfig permet de chosir la plateforme et le vendeur. Positionnez ces options comme indiqué à la figure 4.

Le deuxième menu permet de préciser ce qui doit être configuré. Sélectionnez uniquement l'item « *Customize Kernel Settings* » (voir Fig. 5) de manière à ne paramétrer que le noyau et à conserver

uClinux Distribution v4.0 Configuration
Vendor/Product Selection Arrow keys navigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc><esc> to exit, <? > for Help, for Search. Legend: [*] built-in [] excluded</esc></esc></m></n></y></enter>
Select the Vendor you wish to target Vendor (Altera)> Select the Product you wish to target Altera Products (nios2)>
<pre></pre>

FIGURE 4 – Sélection du vendeur et de la plateforme.

la configuration par défaut des librairies et des applications installées.

uClinux Distribution v4.0 Configuration
Kernel/Library/Defaults Selection Arrow keys navigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc> to exit, <? > for Help, for Search. Legend: [*] built-in [] excluded</esc></m></n></y></enter>
<pre> Kernel is linux-2.6.x Libc is None [] Default all settings (lose changes) [*] Customize Kernel Settings [] Customize Application/Library Settings [] Update Default Vendor Settings</pre>



Quittez le menu. Lors de la première configuration, le terminal réapparait en proposant de régler manuellement différents paramètres. Validez avec entrée tous les paramètres par défaut.

Vous aurez ensuite une nouvelle fenêtre de configuration affichée (voir Fig. 6). Parcourez les différents menus en effectuant les configurations suivantes :

- Fixez l'adresse de base pour la mémoire SDRAM en fonction de l'adresse configurée auparavant avec QSYS, qui est ici 0x0 (menu « *Platform options* », variable CONFIG_MEM_BASE);
- Indiquez la compilation et l'édition de liens du *Device Tree* (DTB) dans l'image du noyau (menu « *Platform options* », variable CONFIG_DTB_SOURCE_BOOL);
- Précisez la localisation du fichier *Device Tree* à prendre en compte (menu « *Platform options* », variable CONFIG_DTB_SOURCE);
- Activez le support de la MMU (menu « Processor type and features », variable CONFIG_MMU);
- Modifiez la fréquence du noyau à 1000 Hz (menu « Kernel features », CONFIG_HZ_1000);
- Vérifiez que les communications séries UART Altera sont opérationnelles, avec et sans JTAG comme montré sur la figure 7 (menu « Device drivers » → « Character devices » → « Serial drivers », variables CONFIG_SERIAL_ALTERA_UART et
 CONFIG_SERIAL_ALTERA_ITAGUART): cette option sera utile pour ouvrir un terminal

CONFIG_SERIAL_ALTERA_JTAGUART) ; cette option sera utile pour ouvrir un terminal à distance sur la carte en utilisant une liaison série.

.config - Linux/nios2 3.7.0 Kernel Configuration	
Linux/nlos2 3.7.0 Kernel Configuration Arrow keys navigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc> to exit, <? > for Help, for Search. Legend: [*] built-in [] excluded</esc></m></n></y></enter>	
General setup > [*] Enable loadable module support > [*] Enable the block layer > Kernel features > Platform options > Processor type and features > Advanced setup > Executable file formats > Executable file formats > Device Drivers > File systems > Kernel hacking > Security options > -*- Cryptographic API > Library routines >	
<pre><select> < Exit > < Help ></select></pre>	

FIGURE 6 – Menu principal de configuration du noyau.

Vérifiez que l'ensemble des configurations ont été réalisées en saisissant / dans la fenêtre de configuration pour visualiser chacune des variables CONFIG_* mentionnées ci-dessus.

5



FIGURE 7 – Menu de configuration des consoles séries.

2.4 Compilation de µClinux

Maintenant que la configuration est effectuée, la compilation du système peut avoir lieu. Plusieurs bibliothèques sont nécessaires pour cette compilation. Voici une liste de paquets nécessaires sous Ubuntu :

```
sudo apt-get install uuid-dev git-core git-gui gcc bison \
flex gawk gettext ccache zlib1g-dev libx11-dev texinfo lib1zo2-dev pax-utils \
uboot-mkimage corkscrew libc6-i386
```

La compilation se fait avec les commandes :

export PATH=\$PATH:<rep installation>/toolchain-mmu/x86-linux2/bin
make

à saisir au niveau du répertoire uClinux-dist.

Si la compilation de $\mu Clinux$ s'est déroulée correctement, vous devriez voir apparaître la ligne suivante :

Kernel: arch/nios2/boot/zImage is ready

Ce fichier zImage est une image du système d'exploitation qui pourra être flashée directement dans la mémoire de la carte. Copiez ce fichier (situé dans le répertoire uClinux-dist/linux-2.6x/arch/nios2...) vers un répertoire personnel.

2.5 Chargement de µClinux sur la carte

Dans cette section, le chargement de μ Clinux est effectué dans la SDRAM de la carte. Le PC est à relier à la carte Altera au niveau de l'entrée USB-Blaster par un câble USB AB.

L'ensemble des scripts de chargement de fichiers sur la carte sont situés dans le répertoire ~/altera/13.0sp1/nios2eds. Positionnez-vous dans ce répertoire. Ajoutez également les commandes

```
suivantes dans le PATH :
```

```
export PATH=$PATH:~/altera/13.0sp1/nios2eds/bin/gnu/H-i686-pc-linux-gnu/bin/
export PATH=$PATH:~/altera/13.0sp1/nios2eds/bin/
```

export PATH=\$PATH: ~/altera/13.0sp1/quartus/linux

```
et mettez à jour la variable LD_LIBARY_PATH :
```

export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:~/altera/13.0sp1/quartus/linux.

Le fichier .sof généré avec QUARTUS doit tout d'abord être envoyé vers la carte.

nios2-configure-sof <répertoire>/<fichier>.sof

<fichier>.sof étant réalisé lors du TP de Mr GOZLAN.

Il faut ensuite charger le noyau en mémoire.

nios2-download -g <répertoire>/zImage

Pour vérifier ce qui se passe sur la carte et interagir avec elle, un câble DB9 mâle/femelle est à utiliser pour établir la communication entre le PC et la carte DE2-115. Vous pouvez ouvrir un terminal à distance sur la carte à l'aide par exemple du logiciel CUTECOM. Pour vous assurer du port utilisé (ttyS0 normalement), vous pouvez utiliser la commande

dmesg | grep tty.

Lancez le démarrage de $\mu Clinux$ sur la carte en exécutant la commande : nios2-terminal.

Si tout fonctionne, vous devriez voir apparaître dans le terminal de CUTECOM un résultat similaire à la figure 8 et témoignant de la décompression de l'image de l'OS puis de son initialisation.



FIGURE 8 – Lignes s'affichant au démarrage de $\mu Clinux.$

3 Configuration du noyau pour intégrer un pilote de périphérique

Maintenant que le système Linux est configuré et fonctionnel, il est possible de lui ajouter une prise en charge du matériel qui a été ajouté dans QSys (le composant LED). Dans cette partie du TP, vous aurez à modifier la configuration du noyau et à ajouter un pilote pour ce composant.

3.1 Configuration du noyau

Pour inclure le code du pilote lors de la compilation du noyau, il est néccessaire d'apporter quelques modifications.

1. Ajoutez l'option à Kconfig dans <rep installation>/linux-2.6/drivers/misc.

```
config LED
tristate "LED custom hardware"
help
LED custom hardware
```

2. Ajoutez un fichier objet au Makefile dans <rep installation>/linux-2.6/drivers/misc

obj-\$(CONFIG_LED) += led.o

3.2 Ajout de fichiers de périphériques au système de fichiers racine

Le pilote utilise un fichier de périphérique (device file) pour communiquer avec le matériel. Pour créer un tel fichier, ajoutez la ligne suivante dans

uClinux-dist/vendors/Altera/nios2/device_table.txt :

/dev/led c 666 0 0 250 0 - - -

3.3 Ajout du code source

Le code source du pilote de périphérique doit être écrit dans linux-2.6/drivers/misc et appelé led.c. Reportez-vous à l'annexe pour le code C du pilote. Ce code est également téléchargeable sur le site e-uapv.

Pour mettre en œuvre l'affichage des LED, complétez le code fourni aux endroits où il y a des commentaires :

- fonction write_led(),
- fonction drv_probe(),
- fin du pilote.

Vous pourrez vous inspirer de la documentation trouvée sur Internet et notamment du cours suivant : http://www.cs.columbia.edu/~sedwards/classes/2014/4840/device-drivers.pdf.

8

3.4 Compilation du noyau

Pour inclure le pilote, le noyau doit être reconfiguré puis recompilé. Positionniez-vous dans le répertoire uClinux-dist puis resaisissez les commandes suivantes :

make menuconfig make

Lors de cette première étape, ajoutez une étoile en regard de « *LED custom hardware* », accessible par les menus « *Device drivers* » puis « *Misc Devices* » (voir Fig. 9). Suivant le mode de configuration



FIGURE 9 – Menu de configuration du module LED.

choisi, il se peut également que des questions vous soient posées directement dans le terminal (Fig. 9). Confirmez alors en particulier l'utilisation du module LED.

3.5 Test du pilote

Rechargez la nouvelle image du système sur la carte en répétant les opérations décrites à la section 2.5.

Si la compilation du noyau a été faite avec succès, le pilote devrait être capable d'afficher des valeurs sur les LED à partir de l'espace utilisateur du système Linux. Pour tester cette fonctionnalité, entrez la commande suivante dans l'invite de commande de μ Clinux :

echo 1 > /dev/led

Les LED devraient afficher 1 en n'allumant que la première LED.

```
🔊 🗇 🗊 🛛 root@DELL-T3400: ~/git-uClinux/uClinux-dist
*** End of Linux kernel configuration.
*** Execute 'make' to build the kernel or try 'make help'.
make[1]: entrant dans le répertoire « /home/philgood/git-uClinux/uClinux-dist »
KCONFIG_NOTIMESTAMP=1 make ARCH=nios2 CROSS_COMPILE=nios2-linux-gnu- 0=/home/p
hilgood/git-uClinux/uClinux-dist/linux-2.6.x -C ../linux-2.6 oldconfig
make[2]: entrant dans le répertoire « /home/philgood/git-uClinux/linux-2.6 »
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC scripts/kconfig/zconf.tab.o
  HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --oldconfig Kconfig
  Restart config...
  Misc devices
Enclosure Services (ENCLOSURE_SERVICES) [N/m/y/?] n
Altera system ID Driver (ALTERA_SYSID) [N/m/y/?] n
LED custom hardware (LED) [N/m/y/?] (NEW)
```

FIGURE 10 - Extrait des sorties de make menuconfig avec le module LED.

3.6 Nettoyage du code

Remettez les fichiers suivants dans leur état initial, c-à-d avant les ajouts nécessaires à la prise en compte du module led:

- linux-2.6/drivers/misc/Kconfig
- linux-2.6/drivers/misc/Makefile
- uClinux-dist/vendor/altera/nios2/device_table.txt

4 Annexe : Code C du pilote de périphérique pour le composant LED

```
/* Driver du moduleLed pour noyau Linux*/
/* TP uCLinux sur DE2-115
*/
#include <linux/module.h>
#include <linux/mod_devicetable.h>
#include <linux/platform_device.h>
#include <linux/of_device.h>
#include <linux/of_address.h>
#include <linux/fs.h>
#include <linux/ioport.h>
#include <asm/uaccess.h>
#include <asm/uaccess.h>
#include <asm/io.h>
```

```
MODULE_DESCRIPTION("LEDG[7..0]udriveruforuAlterauDE2-115");
MODULE_SUPPORTED_DEVICE("none");
#define LED MAJOR 250
static ssize_t write_led(struct file *, const char *, size_t, loff_t *);
static char led_str[3];
int *led_mem=0;
static struct of_device_id of_match_led[] __devinitdata =
ſ
        { .compatible = "ALTR, led -1.0", },
        {}
};
MODULE_DEVICE_TABLE(of, of_match_led);
static struct file_operations fops_led =
ſ
        .write = write_led,
1:
static ssize_t write_led(struct file *fp, const char *buf, size_t len, loff_t *offset)
ſ
        int not_copied, led_value;
        // utilisation de copy_from_user pour remplir led_str
        // not_copied= ???
        led_str[len] ='/0';
        // conversion de led_str en led_value
        // ecriture de la variable led_value a l'adresse led_mem
        return len-not_copied;
}
static int __devinit drv_probe(struct platform_device *op)
{
        struct resource *res;
        if(!of_match_device(of_match_led, &op->dev))
                return - ENODEV;
        res = platform_get_resource(op, IORESOURCE_MEM, 0);
        if(!res)
                 return - ENODEV:
        if(!request_mem_region(res->start, resource_size(res), "led"))
                return -ENOMEM;
        // remplissage de la variable led_mem
        if(!led_mem)
                 return -ENOMEM;
        if(register_chrdev(LED_MAJOR, "led", &fops_led))
        {
                 \texttt{printk}(\texttt{"register_chrdev}: \_led_{\sqcup}failed \setminus \texttt{n"});
                 return -EIO;
        }
        return 0;
}
static int __devinit drv_remove(struct platform_device *op)
{
        struct resource *res;
        res = platform_get_resource(op, IORESOURCE_MEM, 0);
        if(!res)
```

```
return -ENODEV;
                release_mem_region(res->start, resource_size(res));
        return 0;
}
static struct platform_driver platform_driver_led =
ł
        .probe = drv_probe,
        .remove = drv_remove,
        .driver =
        {
                .name = "led",
.owner = THIS_MODULE,
                .of_match_table = of_match_led,
        },
};
static int __init mod_init(void)
{
        int ret;
        ret = platform_driver_register(&platform_driver_led);
        return ret;
}
static void __exit mod_exit(void)
{
        platform_driver_unregister(&platform_driver_led);
        unregister_chrdev(LED_MAJOR, "led");
}
// specification du module_init et du module _exit utilises
```