Guide d'utilisation de l'infrastructure des salles A07/A09

Institut REDS, HEIG-VD

v1.12/Septembre 2015

Cette documentation donne de brèves explications sur l'utilisation des différents outils, environnements, protocoles utilisés dans le laboratoire de systèmes embarqués. Ce document est un résumé, il ne remplace en aucun cas les différentes documentations fournies au début des laboratoires.

1 Introduction

Les **salles A07 et A09** sont équipées machines *DELL Optiflex*, installées durant l'été 2014. Le système d'exploitation installé sur ces machines est *Linux Xubuntu 14.04 LTS (Trusty Tahr)*

Afin d'utiliser l'environnement système, les étudiants doivent être administrateurs de leurs machines. Linux peut être démarré avec trois noyaux différents : un des noyaux est *patché* avec *Xenomai*, l'autre est un noyau *patché* avec le patch temps-réel *CONFIG_PREEMPT_RT*, et le dernier, celui bootant par défaut, est le noyau générique.

Les machines ont été configurées afin de disposer de tous les outils, environnements, services dont vous avez besoin pour le laboratoire. Elles possèdent deux cartes réseau Ethernet : une carte pour la connexion au réseau local (identifiant : em1, intégrée à la carte mère), la deuxième pour la connexion à votre cible (identifiant : *p1p1*, en bas du boitier, sur bus *PCI*).

2 Utilisation de la machine de laboratoire

2.1 Information de login

Sur les machines *DELL*, il faut utiliser le compte utilisateur suivant:

user:	**redsuser**
pass:	**reds**

Pour obtenir les privilèges *root*, il faut utiliser la commande ***sudo su*** dans une fenêtre de commande (le mot de passe est également ***reds***).

2.2 Sauvegarde des données

!!! ATTENTION : les machines de laboratoires sont gérées de manière centralisée. Elles sont réinitialisées de manière globale, potentiellement plusieurs fois par semaine. Le disque est alors formaté automatiquement avant la réinstallation. **!!!**

Il est donc obligatoire de faire une sauvegarde de vos données après chaque séance de laboratoire.

Outre un périphérique de stockage (clé USB, disque dur externe, etc.), les étudiants ont la possibilité de sauvegarder leur *workspace* sur le serveur public ***eistore0***. Les informations pour se connecter se trouvent au chapitre suivant.

2.3 Connexion d'un lecteur réseau

Sur le bureau il y a un raccourcis mappant directement le serveur *eistore0*, on peut également y accéder avec l'URL suivante: ***smb://eistore0.einet.ad.eivd.ch***

Les différentes sauvegardes sont à placer dans le dossier *public*. Attention ce serveur est **public** et donc la garantie que les données ne soient pas effacées/altérées n'est pas assurées.

Dans le cadre de certains laboratoires, il peut vous être demandé de récupérer des fichiers (ou de déposer vos solutions) dans ***smb://eistore1/cours/reds/Cours*** ou ***smb://eisore1/cours/reds/Labo***.

Une fenêtre demandera de vous identifier avec votre login et mot de passe réseau. Ne préfixez pas votre nom d'utilisateur avec le nom de domaine, celui-ci est déjà spécifié. Il est possible que vous deviez entrer ces informations deux fois consécutives avant de pouvoir accéder au partage. Prenez garde de ne pas choisir de conserver vos informations de login au-delà de la fermeture de session.

2.4 Répertoire de travail

Afin d'éviter une prolifération sauvage de dossiers et de fichiers dans le répertoire utilisateur, nous vous invitons à déposer vos fichiers/répertoires dans le **sous-répertoire** "***~/cours_REDS/***<*cours*>*/*" (il faudra créer le sous-répertoire correspondant au cours, par exemple ***ifs*** ou ***sye***).

Attention ! Le répertoire *~/Cours_REDS* peut être viré occasionnellement. **Vous êtes responsables de sauvegarder vos données** (cf. 2.2).

3 Environments privés (ordinateurs portables)

Pour utiliser votre propre ordinateur portable, vous pouvez vous procurer une copie allégée d'une machine de labo sous forme d'une machine virtuelle. Elle est normalement fonctionnelle avec tous les environnements de virtualisation standard supportant le format *OVF*[1] mais **seul *VirtualBox* a été testé et validé**.

Pour des raisons de taille, certaines applications ne sont pas disponibles dans cette VM. Référez-vous au responsable de votre cours pour savoir si la VM répond à vos besoins.

3.1 Installation pas à pas de la machine virtuelle

1. Installer la dernière version de *VirtualBox* en suivant la documentation officielle
   URL de téléchargement: https://www.virtualbox.org/wiki/Downloads
   Documentation : https://www.virtualbox.org/manual/ch02.html

   Installer *Virtualbox* et *l'extension pack*. L'installation de *l'extension pack* se fait simplement en double-cliquant sur le fichier *.vbox-extpack* téléchargeable à partir de la page *Dowloads* donnée ci-avant.

   Si *VirtualBox* est installé sur un hôte *Linux*, ne pas oublier de rajouter l'utilisateur au groupe *vboxusers* pour permettre l'accès aux périphériques USB depuis la machine virtuelle. Voir la page de documentation (section 2.3.4 *The vboxusers group*) pour ce point.

2. Télécharger le fichier correspondant à la machine virtuelle du laboratoire depuis le *SWITCHdrive* de l'institut (http://tiny.cc/s0dc3x). Le fichier en question porte l'extension .ova.

   !!! Il est préférable de se connecter au réseau de l'école via le réseau câblé. Celui-ci offre en effet des débits nettement supérieurs à ceux du réseau wifi pour ce gros téléchargement. Des prises RJ45 libres sont disponibles aux potelets des salles A07 et A09. !!!

3. Démarrer *VirtualBox Manager* et choisir "***Fichier > Importer une application virtuelle…***" et sélectionner le fichier téléchargé au point 2.
4. Valider l'importation de la machine virtuelle en cliquant sur ***Importer***.

   La mise en place de la machine virtuelle démarre. Celle-ci peut durer plusieurs minutes en fonction de la performance du PC hôte.
5. La machine virtuelle est créée et apparaît comme machine disponible dans *VirtualBox Manager*. Elle porte le nom *VM-REDS64*.

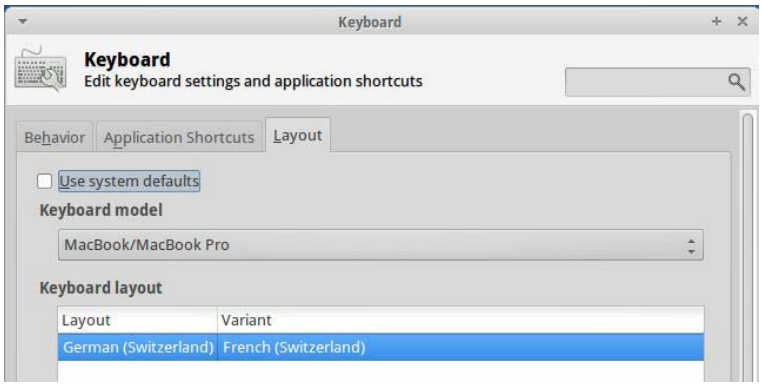
[1] http://en.wikipedia.org/wiki/Open_Virtualization_Format

La VM peut maintenant être démarrée.

6. Vous pouvez supprimer le fichier *.ova* du point 2.

Les options système par défaut pour la machine virtuelle, comme la quantité de mémoire vive (*RAM*) et le nombre de processeurs virtuels, sont configurées pour utiliser le minimum de ressources. Ces configurations peuvent être adaptées à votre machine pour obtenir les meilleures performances, ceci dans le but d'accélérer les compilations et les applications.

Les utilisateurs de machines *Apple (Mac OS)* doivent reconfigurer la disposition du clavier dans la machine virtuelle en allant dans le menu ***Applications*** > ***Settings Manager*** > ***Keyboard*** de la machine virtuelle *Linux* et choisir le *layout MacBook/MacBook Pro*. La nouvelle disposition du clavier ne prendra effet qu'au prochain *login*.

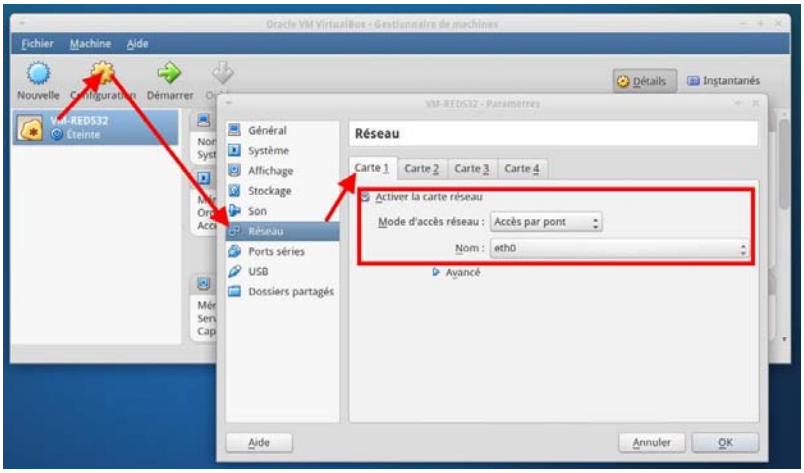


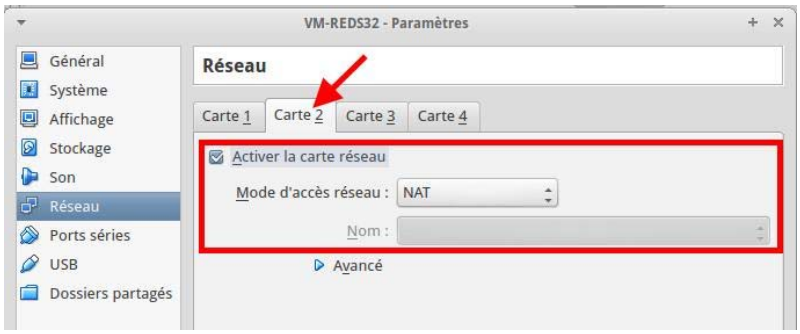
3.2 Configuration du réseau

Lors du premier démarrage, vous devrez modifier la configuration par défaut de la machine virtuelle. La machine virtuelle doit idéalement être configurée avec 2 interfaces réseau routées vers l'interface *WiFi* pour l'une et vers l'interface *Ethernet* (filaire) pour l'autre.

De manière à offrir la connectivité nécessaire pour atteindre Internet et les serveurs de l'école d'une part et les cartes de développement d'autre part (*REPTAR*, *OlinuXino*) les interfaces réseau doivent être configurées de la manière suivante. Ne changez pas les adresses MAC assignées par défaut.

Interface de la machine virtuelle	**Interface physique rattachée**	**Mode**
eth0	interface Ethernet (eth0 sous Linux)	Bridge (pont)
eth1	Aucune (réseau virtuel NAT virtualbox)	NAT





Dans l'environnement de la machine virtuelle, la première carte virtuelle eth0 sera configurée avec l'IP fixe 192.168.1.1, et la deuxième carte virtuelle eth1 sera configurée dynamiquement via le DHCP de Virtualbox. Lorsque votre VM sera démarrée, vous pourrez activer les interfaces réseau en cliquant sur « Carte de Labo » et « Réseau externe » via le gestionnaire de réseau en haut à droite. A ce point vous devriez avoir accès à Internet et à la carte LAN de votre machine hôte.

3.3 Activation du port série sous *Virtualbox*

Si vous désirez accéder au terminal de la carte *REPTAR ou OlinuXino*, celui-ci est accessible via un convertisseur USB-série. Pour y accéder depuis la machine virtuelle, il faut rediriger le périphérique USB correspondant vers la machine virtuelle.

La manipulation suivante donne accès au convertisseur USB-série depuis la machine virtuelle.

1. Brancher le câble mini-USB entre la carte REPTAR et le PC et démarrer la carte REPTAR en suivant les instructions du chapitre suivant
2. Dans la fenêtre Virtualbox, en bas à droite, faire un clic droit sur l'icône "USB" et choisir le périphérique "***Silicon Labs CP2103 USB to UART Bridge Controller***" (pour REPTAR) ou **"Prolific Technology Inc. USB Serial Converter"** (pour OlinuXino).

Le périphérique USB est maintenant géré par la VM et une nouvelle entrée est automatiquement créée dans */dev* (typiquement */dev/ttyUSB0*).

3. Se connecter à la carte via un terminal série en suivant les indications du chapitre consacré à *picocom*.

3.4 Redirection du lecteur de carte SD

Pour accéder la carte SD ou micro-SD connectée au PC hôte depuis la machine vituelle, le lecteur de carte doit être redirigé vers la machine virtuelle. Cette opération s'effectue de la même manière que pour le port série (voir point précédent) mais en choisissant le périphérique ***"Generic Flash Card Reader/Writer"***

4 Carte de laboratoire REPTAR

REPTAR associe un processeur de type *OMAP* (*TI DaVinci DM3730*), lui-même constitué d'un core *ARM Cortex-A8* et d'un DSP, avec un composant programmable (FPGA) de type *Xilinx Spartan 6.*

La plate-forme comporte également un grand nombre de périphériques de contrôle, d'affichage et de communication. Grâce à sa conception modulaire, elle offre de nombreuses possibilités d'extension.

L'adresse IP préconfigurée de la carte *REPTAR* est ***192.168.1.<xxx>*** Le dernier octet varie selon le numéro de la carte *REPTAR* utilisée.

Connexion à la carte *REPTAR* depuis une machine de laboratoire :

1. **Alimenter la carte en 12V** avec des fiches bananes, depuis une alimentation du laboratoire
2. **Connexion du port USB** de la machine hôte à la cible. Le connecteur à utiliser est celui libellé USB *debug* situé sous l'écran à l'arrière de la carte processeur.
3. **Connexion Ethernet** de la machine hôte à la cible (carte réseau **2**, câble *Ethernet* croisée rouge)

4.1 Emulateur de terminal *picocom*

L'application *picocom* est l'un des émulateurs de terminal série qui peut être utilisé afin d'accéder au moniteur de la carte depuis la machine-hôte. L'application peut se lancer de la manière suivante :

```
$ sudo picocom –b 115200 /dev/ttyUSB0
```

/dev/ttyUSB0 représente le contrôleur série-USB N°0. Il arrive parfois lors d'une déconnexion du câble suivie d'une reconnexion rapide qu'un nouveau périphérique *ttyUSB1* apparaisse. Si la commande ci-dessus échoue alors que le câble est branché, essayez alors de remplacer *ttyUSB0* par *ttyUSB1*.

Pour quitter *picocom*, appuyez *Ctrl+a* suivi de *Ctrl+x*.

4.2 Programmation de la FPGA

Pour pouvoir programmer la Spartan 6 avec le câble micro USB et le contrôleur JTAG intégré, il faut mettre le switch N°2 du DIP Switch SP3, à OFF (en bas).

DIP Switch SP3

Dans le cas d'une programmation via une sonde JTAG série, le switch N°2 du DIP Switch SP3, doit être ON

DIP Switch SP3

Se référer au guide d'utilisation de la REPTAR pour plus d'informations.

5 Emulateur *Qemu*

Sur les machines de laboratoire, *Qemu* a été compilé pour émuler les systèmes ARM, et il s'exécute avec la commande suivante:

```
$ qemu-system-arm [suivi de divers paramètres
```

L'option *serial* permet de rediriger la console série de *Qemu*, ainsi nous pouvons la paramétrer pour rediriger sur la console *stdio* :

```
$ qemu-system-arm ... -serial stdio
```

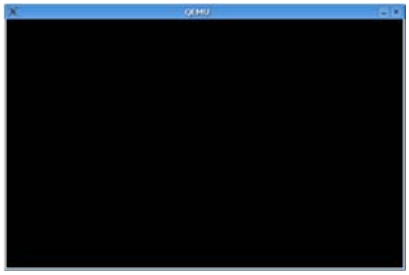
Pour lister les paramètres:

```
$ qemu-system-arm -help
```

ou simplement

```
$ qemu-system-arm
```

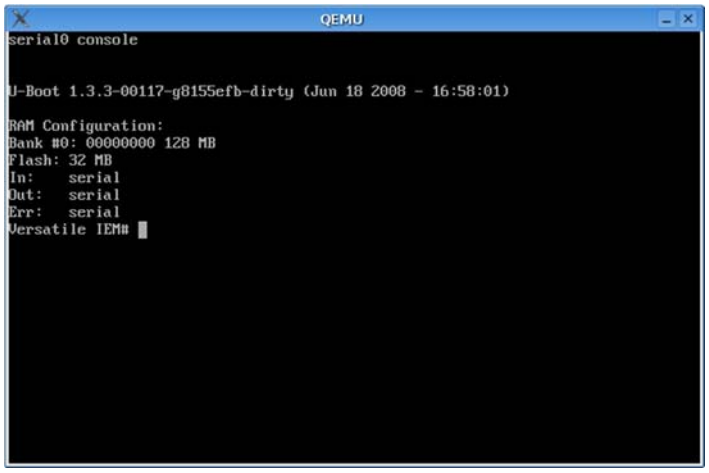
Qemu s'ouvre dans une nouvelle fenêtre:



Qemu fonctionne sur quatre fenêtres différentes:

La fenêtre d'accueil (celle ci dessus) contient l'environement émulé (**Ctrl-Alt 1**). Pour changer de mode, il faut utiliser les touches Ctrl-Alt + touche 1 à 4:

- **Ctrl-Alt 2**: Affiche les informations du "*monitor*"
- **Ctrl-Alt 3**: Commute sur la console serie (voir ci dessous)



- **Ctrl-Alt 4**: Commute sur la console parallèle
- **Ctrl-Alt**: Sort du system émulé et redonne la mains a l'OS principal (*Linux-Xubuntu*) ainsi que les périphériques ci-rapportant (par ex: la souris).

5.1 API Qemu et GUI (Graphical User Interface)

Ce document décrit les fonctions de communication entre l'émulateur *qemu* et l'interface graphique d'émulation utilisée dans le cadre de quelques labos avancés de cours Master.

Architecture générale

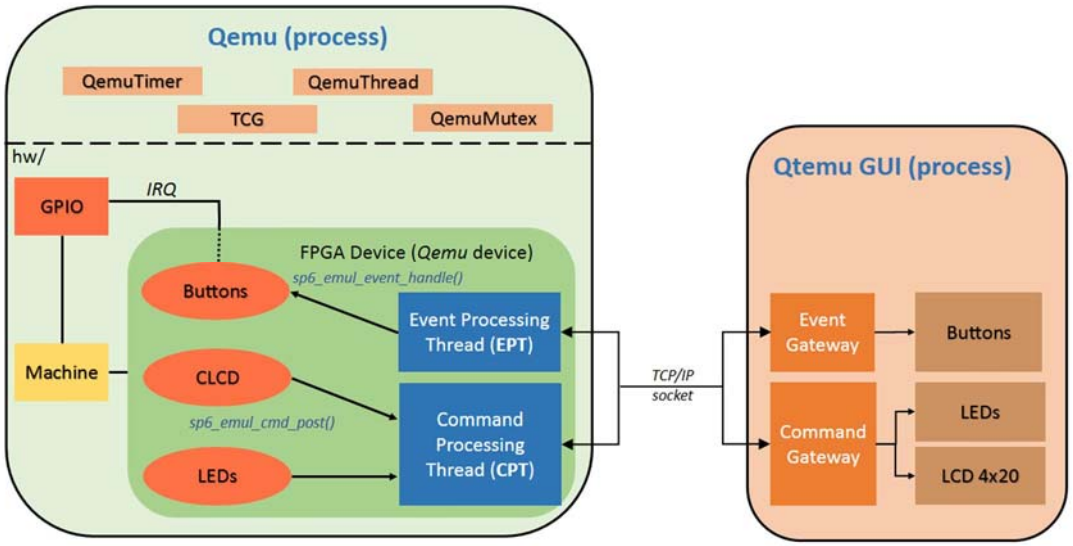


Figure 1. Architecture générale de l'interface de communication *qemu* avec le *GUI*

La communication entre l'émulateur et le *frontend* graphique est assurée par deux *threads* s'exécutant dans le processus *qemu* et deux autres *threads* s'exécutant dans le processus de l'application graphique.

Dans les deux cas, un *thread* gère l'émission de messages tandis que l'autre réagit à la réception de messages. Les messages sont échangés sur une socket TCP en local sur la machine exécutant le *framework* d'émulation.

Format des messages

Les messages échangés entre *qemu* et la *GUI* sont formatés de manière générique sous forme d'objets JSON séparés par des retours à la ligne. Les données dans ses objets sont stockées sous forme de paires **clé/valeur,** les clés étant sous forme de texte.

Chacun de ses objets JSON doit posséder une clé « *perif* » indiquant le périphérique correspondant au message, sous forme d'une chaine de caractères. Les autres champs présents dans l'objet JSON sont spécifiques à chaque périphérique concerné

QEMU n'inclut pas nativement des fonctionnalités de support JSON. **cJSON**, une bibliothèque externe, est donc utilisée pour générer et parser le JSON dans QEMU.

Représentation graphique d'un message

L'avantage d'utiliser JSON est que les messages échangés sont facilement lisibles par un humain. Voici une capture du flux TCP échangé en local (en rouge, les messages envoyés depuis QEMU, en bleu ceux envoyés depuis l'interface graphique QtEMU)

{"perif":"led","value":240}
{"perif":"7seg","digit":1,"value":9}
{"perif":"lcd","command":"update","ddram":"@ "}

{"perif":"btn","status":4}
{"perif":"btn","status":0}

Envoi de commandes depuis qemu

L'envoi de messages depuis du code d'émulation s'effectue en trois temps. L'exemple ci-dessous (pour un périphérique imaginaire *mon_perif*) illuste cette séquence.

Premièrement, il faut créer un Objet JSON.

```
cJSON *root = cJSON_CreateObject();
```

Ensuite lui assigner un périphérique valable et des paires de clé/valeur :

```
cJSON_AddStringToObject(root,"perif","mon_perif");


cJSON_AddStringToObject(root,"status","run");        // Pour une valeur de type string
cJSON_AddNumberToObject(root,"speed",10);            // Pour une valeur de type entier
```

Pour terminer, utiliser la fonction *sp6_emul_post* pour mettre en queue l'envoi du paquet.

```
void *sp6_emul_cmd_post(cJSON *root)
```

Le paquet sera ensuite mis en queue et envoyé automatiquement, et l'espace mémoire qui lui était réservé sera automatiquement nettoyé. Les noms des périphériques dans l'interface QtEMU, ainsi que les clés/valeurs attendues par les périphériques implémentés sont visibles dans le tableau ci-dessous.

Leds		**7 segments**		**CLCD**	
perif	**leds**	*perif*	**7seg**	*perif*	**clcd**
Clé	*Valeur*	*Clé*	*Valeur*	*Clé*	*Valeur*
value	Entier dont les 8 bits LSB définissent l'état des LEDS	**digit**	Entier de 1 à 3 permettant d'identifier le digit à modifier	**command**	commande à exécuter (*update* ou *clear*) de type string
		value	Entier de 0 à 9 représentants la valeur à afficher	**ddram**	Chaine de 4x20 chars indiquant la nouvelle valeur à afficher dans le cas d'une commande *update*.

Réception d'événements (events) dans qemu

Les *messages* envoyés par l'interface graphique sont récupérés dans la fonction *sp6_emul_event_handle()*. Le paquet JSON parsé se retrouve dans la variable locale *root*.

La valeur correspondant à chacune des clés se récupère en utilisant la fonction *cJSON_GetObjectItem* dont voici le prototype.

```
cJSON *cJSON_GetObjectItem(cJSON *root, const char *key) ;
```

Si la valeur attendue pour cette clé-là est une chaine de caractères, elle peut être récupérée via son attribut *valuestring*. Si l'on attend une valeur numérique, celle-ci peut être récupérée via son attribut *valueint*.

La seule clé obligatoirement présente dans le message JSON en provenance de QEMU est *perid*. La valeur textuelle correspondant à cette clé est inspectée afin de déterminer à quel périphérique est destiné le paquet.

```
char *perid = cJSON_GetObjectItem(root,"perid")->valuestring;
```

En fonction de l'identifiant de périphérique, il est ensuite possible d'associer une fonction de type *callback* associé au message, qui aura pour rôle d'exploiter les autres données présentes dans la structure cJSON.

```
if(strcmp(perid,"mon perif") == 0)
       callback_mon_perif(root);
```

Le paires clé/valeurs émises par chacun des périphériques d'entrée de Qtemu (pour le moment, juste les boutons) sont affichées dans le tableau ci-dessous.

Boutons	
perif	**btn**
Clé	*Valeur*
status	Entier dont les 8 bits LSB indiquent l'état des boutons

La fonction ci-dessous est un callback d'exemple pour les messages en provenance des boutons.

```
void callback_boutons(cJSON *root)
{
        int btnStatus = cJSON_GetObjectItem(root,"status")->valueint;
        printf("0x%x\n", btnStatus);
        cJSON_Delete(root);             // Ne pas oublier de libérer la mémoire
}
```

6 U-boot

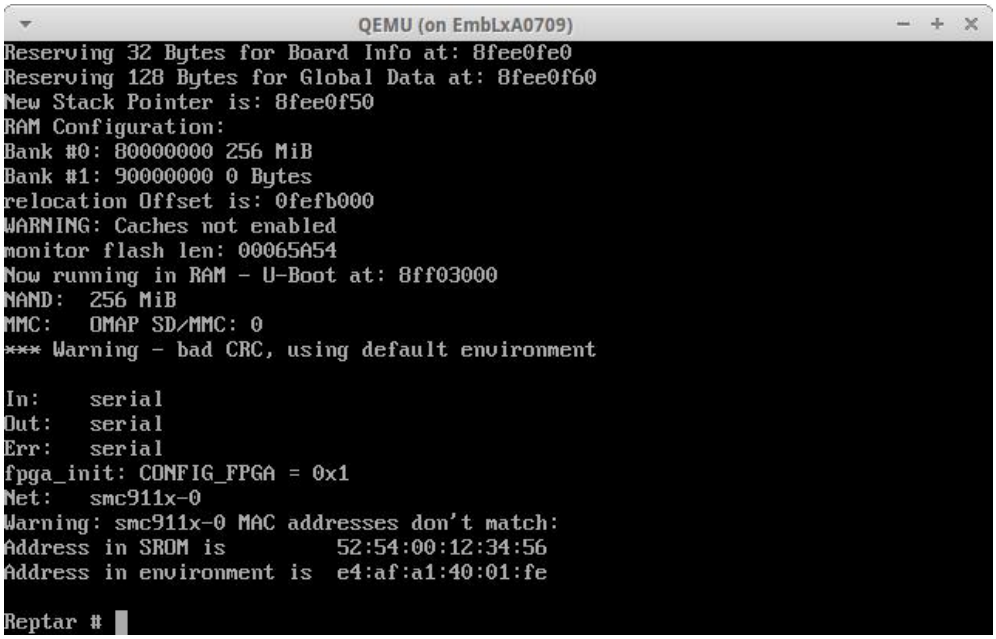
U-boot, *The Universal Bootloader*, est un moiteur multi-architecture utilisé dans de nombreux laboratoires. C'est le moniteur qui est installé sur la REPTAR, mais c'est également lui qui est exécuté dans l'émulateur Qemu dans le cadre de certains laboratoires.

Il est contenu dans le répertoire *u-boot-2009.11-reptar* du dépôt des sources de la REPTAR. Une fois compilé, il apparait sous forme de deux exécutables de type: *u-boot* (au format ELF, contient les symboles pour le *debug* notamment) et *u-boot.bin* (fichier aplati, déstructuré, qui représente l'image binaire telle qu'elle sera chargée dans l'espace mémoire). C'est ce dernier fichier qui sera chargé par l'émulateur, ou qui sera stocké dans la *sdcard* utilisée avec la plate-forme *REPTAR*.

Si U-boot est utilisé sous *Qemu*, le démarrage de l'environnement émulé s'effectue à l'aide des scripts fournis. Les scripts principaux sont les suivants:

- *stf* — Démarrage avec une image flash située dans *filesystem/flash*
- *stf-debug* — Démarrage comme *stf* avec *breakpoint* sur la première instruction
- *sts* — Démarrage avec la *sdcard*.
  Le noyau *Linux* et son *rootfs* se trouve dans celle-ci. La version de *u-boot* utilisée est celle de l'arborescence U-boot.
- *sts-debug* — Identique à *sts* avec *breakpoint* sur la première instruction

Chacun de ces scripts existe avec la forme "réseau" (suffixe "n"). Par exemple, *stfn* est équivalent à *stf* avec la configuration réseau permettant d'exploiter la carte réseau locale de la machine hôte.



Voici quelques commandes de base:

help (ou ?)

Liste toutes les commandes disponibles dans u-boot
Par exemple, pour obtenir de l'aide sur la commande *printenv* **:**

```
# help printenv
```

setenv <var> <value>

Définit une variable d'environnement <var> avec un contenu <value>

```
# setenv ipaddr 192.168.2.4
```

On peut aussi utiliser cette commande pour créer une commande (ou série de commandes séparées par une virgule) :

```
# setenv appExample tftp 0x81600000 /drivers/example.bin
```

run <var>

Exécute la commande définie dans la variable d'environnement <var>

```
# run appExample
```

tftp <addr> <file>

Transfère un fichier sur la cible via le protocole tftp à une certaine adresse

```
# tftp 0x81600000 try
```

- Si vous utilisez la plateforme REPTAR physique, il faut au préalable déposer le fichier à transférer dans **/home/redsuser/tftpboot** de la machine-hôte, et avoir exécuté les commandes suivantes sous REPTAR afin d'attribuer une IP à la carte.

```
# run setmac; run setip
```

- Si vous utilisez l'émulation sous QEMU, le fichier est à déposer dans le répertoire courant de QEMU, et l'adresse du serveur est 10.0.2.2. Il est donc nécessaire d'adapter la variable d'environnement serverip

```
# setenv serverip 10.0.2.2
```

go <addr>

Poursuit l'exécution à l'adrese <addr>

```
# go 0x81600000
```

md.(b|w|l) <addr> <count>

Affiche <count> données depuis l'adresse <addr>. Le type des éléments à afficher dépend du suffixe appondu à la commande md

```
md.b lecture d'un byte (8 bits)
md.w lecture d'un word (16 bits)
md.l lecture d'un long (32 bits)
```

Exemple 1 : lire la valeur du registre sw0 et sw1 sous forme de long (32 bits)

```
# md.l 0x49056038 1
```

Exemple 2 : lire la valeur de 10 éléments à l'adresse 0x18000000 sous forme de word ***(ATTENTION, un word = 16 bits sous U-boot)***

```
# md.w 0x18000000 10
```

L'espace adressé des FPGAs se situe depuis l'adresse 0x18000000 jusqu'à 0x19FFFFFF. Seuls des transferts mémoire 8 ou 16 bits sont autorisés vers/depuis les FPGAs.

Exemple 2 : affiche les 64 premiers bytes situés à l'adresse 0x81600000

```
# md.b 0x81600000 64
```

mw.(b|w|l) <addr> <value>

Ecrit <value> à l'adresse <addr>. Le type des éléments à écrire dépend du suffixe de la commande mw (avec la même signification que pour la commande md).

Exemple 1 : Allumer le LED 0.

```
# mw.l 0x49056094 0x8000
```

7 *Linux* embarqué

Pour démarrer un Linux embarqué sur la carte *REPTAR*, il faut avoir en flash le noyau *Linux* embarqué (*uImage*), le bootloader (*MLO*), le moniteur (*u-boot.bin*) ainsi que l'arborescence racine sur la *sd-card*. Tous ces composants ont été transférés à l'avance sur les cartes de laboratoires à l'aide des commandes précédentes.

7.1 Démarrage du noyau

Pour démarrer Linux sur la carte *REPTAR* ou dans l'emulateur *Qemu*, il faut taper la commande

```
# boot
```

7.2 Transfert de fichiers avec scp

Pour transférer des fichiers sur *Linux* il est possible d'utiliser la **commande *scp***.

- Copie d'un fichier sur la cible depuis une machine serveur :

```
$ scp <path1>/<file1> root@192.168.1.XX:<path2>/<file2>
```

- Copie d'un fichier de la machine de laboratoire sur la cible depuis la cible :

```
$ scp redsuser@192.168.1.1:<path>/<file> .
```

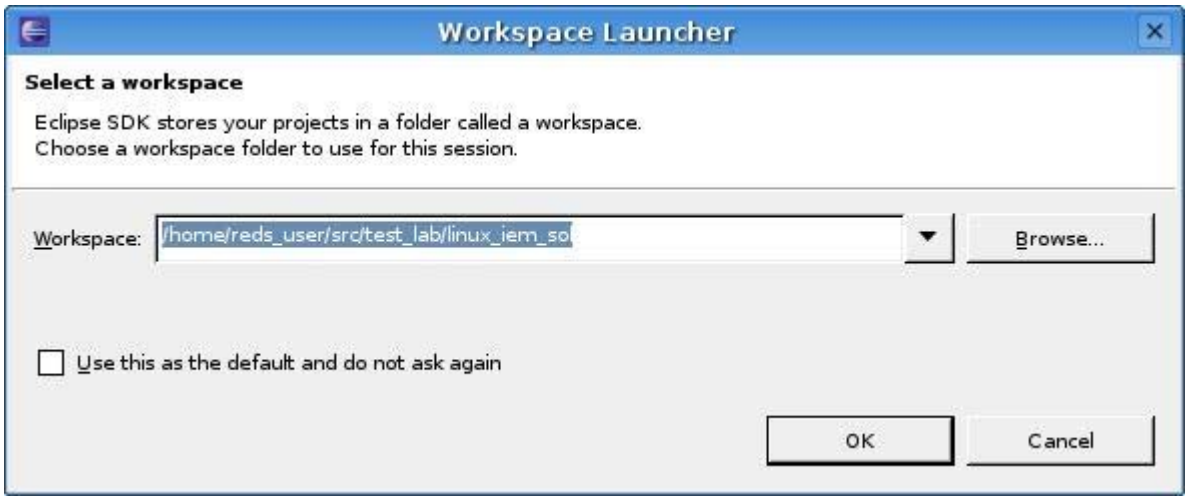
Indications

- Pour copier un répertoire, utiliser l'option « –r » dans les commandes ci-dessus.

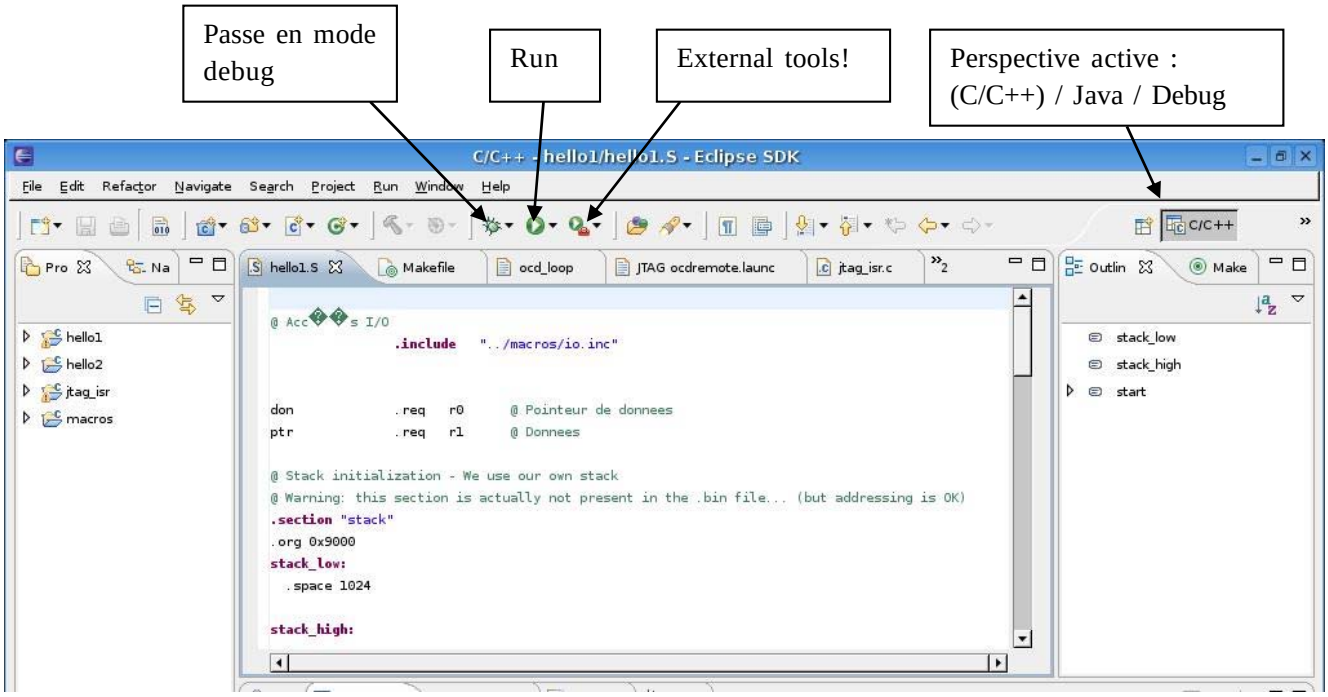
8 Utilisation du debugger

Dans certains laboratoires, vous serez amenés à debugger vos propres programmes, qu'ils soient exécutés sous environnement émulé, ou sur la REPTAR directement, via un adaptateur JTAG USB.

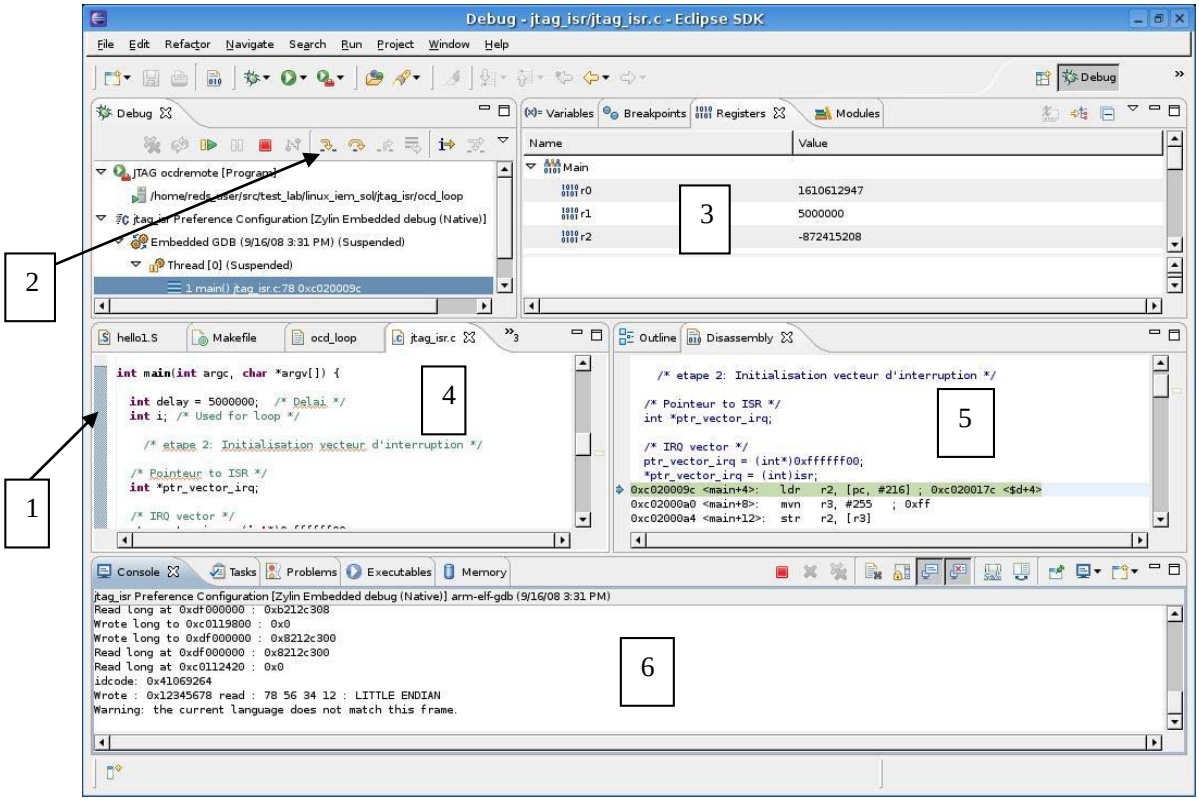
Dans tous les cas, l'environnement de debug *d'eclipse* sera utilisé.



Choisissez-votre espace de travail!



Fenêtre debug:



1. Possibilité de mettre des breakpoints
2. Mode de debug (pas à pas, …)
3. Fenêtre des registres
4. et 5. Fenêtres contenant le code
6. Affichage de la console