

μ Clinux on the Altera DE2

David Lariviere and Stephen A. Edwards
Columbia University, 2008.

Abstract:

This technical report provides an introduction on how to compile and run uClinux and third-party programs to be run on a Nios II CPU core instantiated within the FPGA on the Altera DE2. It is based on experiences working with the OS and development board while teaching the Embedded Systems course during the springs of 2007 and 2008.

INTRODUCTION.....	3
UCLINUX.....	3
BUILDROOT.....	3
uCLIBC.....	3
HARDWARE.....	4
ALTERA DE2.....	4
FPGA.....	4
SOPC.....	4
Nios II.....	4
DEVELOPMENT ENVIRONMENT.....	4
QUARTUS II.....	4
VMWARE.....	4
WALKTHROUGH.....	6
BUILDING uCLINUX.....	6
<i>Running uClinux</i>	6
<i>Directory Structure of uClinux-dist</i>	7
GETTING STARTED.....	7
CONFIGURING HARDWARE.....	7
<i>Minimum Tools</i>	7
PROGRAMMING THE DE2 WITH uCLINUX.....	8
<i>Hello World</i>	8
DEMO APPLICATIONS.....	8
<i>LED</i>	9
<i>Audio</i>	9
<i>Frame Buffer</i>	9
CONFIGURING UCLINUX FOR FPGA HARDWARE.....	9
CUSTOMIZING THE KERNEL.....	10
PORTING SOFTWARE.....	11
CONFIGURING THE DEVELOPMENT ENVIRONMENT.....	11
<i>Installing VMware Tools</i>	11
INSTALL GCC CROSS-COMPILER.....	12
<i>Extract Archive</i>	12
<i>Add to Path</i>	12
INSTALL REQUIRED UTILITIES.....	13
EXTRACTING UCLINUX DISTRIBUTION.....	14
EXTRACT GIT REPOSITORY.....	14
CONFIGURING uCLINUX-DIST FOR THE FIRST TIME.....	14
CUSTOMIZING THE KERNEL.....	17
<i>Ethernet/Network Support</i>	17
<i>USB Key (File system) Support</i>	17
CONFIGURING THE HARDWARE.....	20
VGA AND PS2 KEYBOARD SUPPORT.....	20

Introduction

uClinux

uClinux, pronounced “you-see-linux”, is a port of the Linux operating system targeting platforms that lack memory management units. uClinux was first written for the Motorola DragonBall processor, and has since been ported to a large number of embedded architectures: Altera Nios II, Xilinx Microblaze, ARM.

The lack of an MMU imposes additional requirements. The current Nios II set of patches do not provide support for shared libraries. Thus all libraries must be statically linked in to the final binary, making compiled binary size, especially for embedded systems, a critical concern. Several open source projects centered around uClinux exist, tailored to the specific requirements of embedded development.

uClinux was originally a set of patches to the existing Linux kernel. It has since been incorporated into the main Linux kernel.

Buildroot

As stated on the project’s website, Buildroot “is a set of Makefiles and patches that allow to easily generate both a cross-compilation toolchain and a root filesystem for your target.”¹ In order to compile programs that can be executed by the Nios II, the Buildroot is used to create a version of the GCC compiler and associated tools specifically for the Nios II and uClinux.

A preconfigured binary is hosted by Altera on their ftp site at <ftp://ftp.altera.com/outgoing/nios2gcc-20080203.tar.bz2> .

uClibc

uClibc is a ported implementation of the GNU C library, rewritten specifically to reduce compiled binary sizes.

The C standard library contains the implementation of the most common and required C functions (printf, fopen, rand, etc).

Note that while uClibc *can* be controlled via the standard uClinux-dist build configuration menus, with the current Nios Buildroot config, it *is not*. Rather, uClibc is preconfigured, compiled, and stored (located under /opt/nios2/nios2-linux-uclibc) along with the GCC crosscompiler tools for the Nios. Whenever the compiler is used to compile an application, it will automatically use uClibc.

What do we want here? Level of detail, if any, or skip entirely?

Options:

- Description of FPGA in general.
- Description of Nios processor.
- Walkthrough of building minimal SOPC configuration to run uClinux on DE2?

Hardware

Altera DE2

FPGA

SOPC

Nios II

Development Environment

Quartus II

Quartus II is Altera's IDE for developing and compiling the hardware configuration that the FPGA will be programmed with. Quartus II comes in two versions: Web Edition, offered free, and Subscription edition, which either requires purchasing an annual subscription, or requesting a donation from Altera (for academic institutions that qualify). The free Web Edition version, however, only exists on Windows.

For this technical report and included development environment, we assume that the user's host machine has Quartus II pre-installed, and thus do not include a pre-installed version within the provided VM development environment.

VMware

This technical report is accompanied with a preconfigured ready-to-go VMware virtual machine image containing a development environment. Included are all tools needed to compile both uClinux and third party applications to be run on the Altera DE2 FPGA, if configured with the appropriate hardware configuration.

Do we want to go into details of configuring/optimizing VMware. VMware player vs. server, how to setup a configuration to allow VMware server to host images that students can remotely connect into?

Configuring Network settings (bridged vs. NAT).

Optimization Settings

- Disable encryption of VM session (where communication between VMware server and a client (either on the local machine or remote) is not SSL encrypted).
- Disable paging (if VMware image stored on removal drive, such as iPod or USB drive).
- Ensure image is on drive configured for performance (instead of quick removal).

Also, transferring files between VMware environment and host machine? Could describe SAMBA/NFS for windows/Linux hosts respectively, but also need to try latest version of VMware Server released last week to see if file sharing from VMware tools works.

The accompanied VM-based development environment consists of a minimal CentOS 5.1 version of Linux, the Nios II-specific port of GCC, and the uClinux Buildroot. CentOS is a distribution of Linux that is binary compatible with Redhat Enterprise Edition, but is free.

The included development environment is configured as follows:

- User Accounts (Description: login/password):
 - Root user: root/fpga
 - Main user: fpga/fpga
- uClinux-dist (located in /home/fpga/uClinux-dist): containing all source code and Makefiles used to compile the Linux kernel, user mode programs, and generate the Linux image.
- GCC toolchain for the Nios II (located in /opt/nios2): contains the GCC crosschain and uClibc. Note that /opt/nios2/bin has already been added to the path.

Walkthrough

The following details the basic steps, using the provided VM environment, to compile uClinux and then upload it to the DE2.

Building uClinux

First launch the virtual machine in VMware by opening “uClinuxOnNios.vmx”. Once the virtual machine starts, it will boot CentOS up. When it has finished booting, the login prompt will be displayed. The login and password are both: `fpga`

```
CentOS release 5 (Final)
Kernel 2.6.18-53.el5 on an i686

fpgadev login: fpga
Password:
Last login: Fri May 30 13:31:58 on tty1
Welcome to the uClinux Nios II Buildroot!
-----
To compile uClinux, applications, and generate a bootable image
that can be downloaded to the DE2, please type 'make'
-----
To configure the kernel or the applications to be included
in the image, type 'make menuconfig'.
-----
[fpga@fpgadev uClinux-dist]$_
```

Once logged in, you can immediately build uClinux. Simply type: `make`

Running `make` will invoke a series of build scripts that will:

1. Prepare the build environment for the specific target architecture (Nios II) and specific FPGA's programmed hardware configuration (using a provided PTF file generated by SOPC Builder).
2. Compile uClinux
3. Compile applications.
4. Generate a filesystem containing all apps and other miscellaneous files.
5. Combine the Linux kernel and file system into a single bootable image (located in `~/uClinux-dist/images/zImage`)

Running uClinux

When the entire build process is complete, `~/uClinux-dist/images/zImage` is the file to be executed on the Nios II's FPGA.

Configure the FPGA

First transfer the file from the virtual machine to a host machine connected to the DE2.

First program the FPGA on the DE2 using the default compiled hardware SOF file:

```
quartus_pgm -mode=jtag -o p:C:/path/default_de2.sof
```

Upload the uClinux Image

Then convert the zImage to the appropriate format and download it to the board:

```
nios2-elf-objcopy C:/path/zImage -O srec zImage.srec  
nios2-gdb-server -g zImage.srec
```

The DE2's FPGA is now programmed with the proper hardware configuration expected by uClinux, and the uClinux image containing both the kernel and filesystem have been downloaded to the board. All that is left is to connect to the Nios over the JTAG UART:

```
nios2-terminal
```

Directory Structure of uClinux-dist

When working with uClinux-dist to compile uClinux systems for the Nios II, the most relevant directories are:

- `images`: after compilation the final compressed "zImage" file is placed here. This is the file
- `linux-2.6.x`: contains the source code for the v2.6 of the Linux Kernel, used to run Linux on the Nios II.
- `user`: contains source code for all possible applications that can be included in the Linux system. Note that uClinux-dist contains applications that are ported only to specific architectures, excluding the Nios II, or to specific boards, excluding the DE2. It is therefore important to check the README and other documentation/source code of the application.

Getting Started

Configuring Hardware

Minimum Tools

Note that the most common method of downloading software to the board relies on a set of scripts (`nios2-configure-sof` and `nios2-download`) for both programming the FPGA and downloading the Linux image to memory. These programs require the installation of the full Nios SDK, which is more than a gigabyte in size.

If one only intends to use the provided virtual machine and uClinux, without the use of nios2-ide, it is possible to program and download the linux image to the DE2 without installing the Nios SDK.

Given a Quartus-compiled hardware design (DE2_NIOS_HOST_MOUSE_VGA.SOF) file and uClinux-generated image (zImage), it is possible to load uClinux on the board by:

```
quartus_pgm --mode=jtag -o p;DE2_NIOS_HOST_MOUSE_VGA.sof
nios2-elf-objcopy zImage -O srec zImage.srec
nios2-gdb-server -g zImage.srec
nios2-terminal
```

Programming the DE2 with uClinux

Programming applications to run on uClinux is little different from programming any other applications targeting a Unix or generic ANSI C environment. The only restrictions are:

- no usage of “fork()” system call (use vfork instead).
- No shared libraries (all libraries must be statically linked in)
- All binaries must be compiled with the “-elf2flt” command, in order to generate binaries in the binary flat executable format, instead of the default ELF format.
- If a stack size greater than 4 KB is needed, add “-sXXX” to the linking command line, where XXX is the number of bytes to pre-allocate for the stack.

Hello World

For example, given the classic hello.c file (see below):

```
#include <stdio.h>
void main() {
    printf(“Hello World!\n”);
}
```

One need only use:

```
nios2-linux-gcc hello.c -o hello -elf2flt
```

Demo Applications

Several demo applications were developed in order to demonstrate interacting with several of the hardware components on the DE2 board.

Note that all user-mode applications are located within /home/fpga/uClinux-dist/user.

The demo examples created for this report are included in /home/fpga/uClinux-dist/user/de2examples.

LED

Audio

Frame Buffer

Configuring uClinux for FPGA Hardware

Note that the provided development VMware image comes with uClinux-dist pre-configured for the Altera Nios II, with default kernel and usermode configuration (as described in the appendix).

Note that in order for uClinux to run on the FPGA, the FPGA must contain a SOPC containing a Nios. Every SOPC is described by a PTF file.

It is possible to remotely Assuming that the DE2 System CD (version 1.4b) is mounted at /mnt/de2systemv14b, enter the following to configure uClinux-dist for the DE2_NET project:

```
make vendor_hwselect
SYSPTF=/mnt/de2systemv14b/DE2_demonstrations/DE2_NET/system_0.ptf.
```

When prompted for:

1. Please select which CPU you wish to build the kernel against: type "1". (Note that the system only has one Nios CPU).
2. Please select a device to upload the kernel to: type "1".
3. Please select a device to execute kernel from: type "2".

Note that directly before hitting enter, after typing "2" for the third option, the screen should appear as below:

```
Selection: 1
--- Please select a device to upload the kernel to:
(1) cfi_flash_0
    Class: altera_avalon_cfi_flash
    Size: 4194304 bytes
Selection: 1
--- Please select a device to execute kernel from:
(1) sram_0
    Class: sram_16bit_512k
    Size: 524288 bytes
(2) sdram_0
    Class: altera_avalon_new_sdram_controller
    Size: 8388608 bytes
(3) epcs_controller
    Class: altera_avalon_epcs_flash_controller
    Size: 2048 bytes
Selection: 2_
```

Congratulations, uClinux is now configured to run on the DE2, when programmed with DE2_NET.

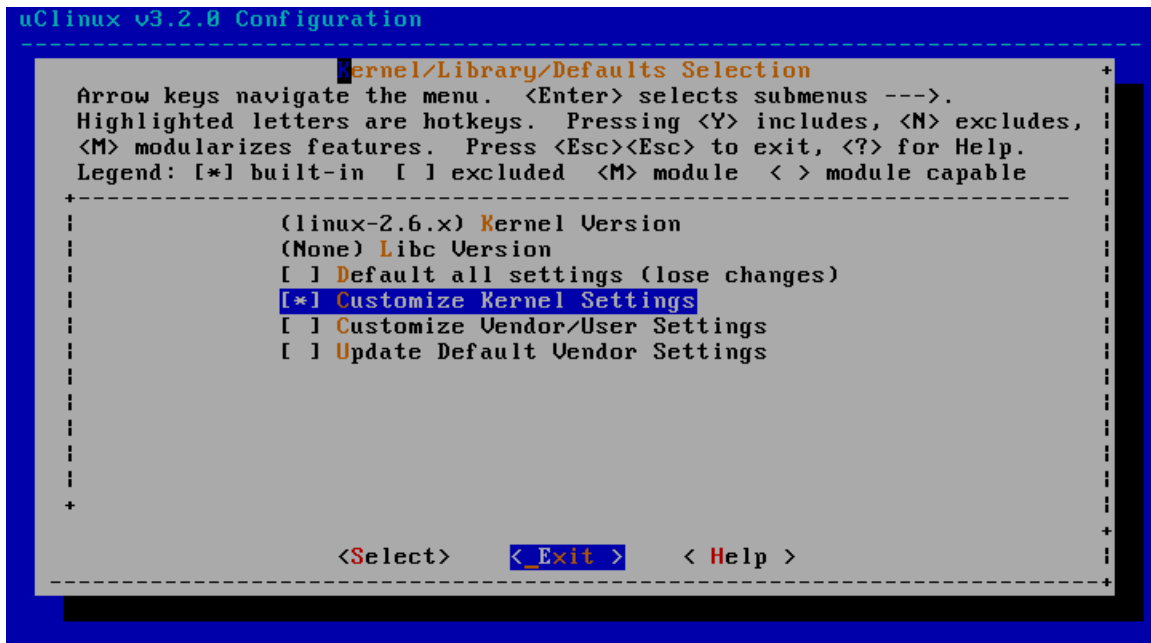
Lastly, to compile uClinux, simply type:

```
make
```

Customizing the Kernel

From within ~/uClinux-dist, run “make menuconfig”

Then choose “Kernel/Library/Defaults Selection”, and lastly check “Customize Kernel Settings”:



Now exit out of both menus, and choose “Yes” to save. This will now launch the configuration for the kernel.

Porting Software

Note: can use `-Dfork=vfork` to rename

Configuring the Development Environment

This section contains a description of all of the steps performed in order to configure the provided Virtual Machine development environment that accompanies this technical report. It should not be necessary to perform them again within the provided VM, but is described for reference to configure additional environments.

Installing VMware Tools

VMware tools is a set of tools that help the VMware image to better interact with the host computer, including accelerated graphics and file hosting. If upgrading VMware tools in the future, then first remove the current installation of VMware tools by running `/usr/bin/vmware-uninstall-tools`, and deleting all previous installation directories in `/tmp/vmware*`

Before installing VMware tools, first install the necessary third party tools.

```
yum install gcc gcc-c++ kernel-devel
```

may need to run `/usr/bin/vmware-config-tools.pl`

commented out “if (not (\$header_version_uts eq...”

Install GCC Cross-Compiler

Extract Archive

Provided with the installation files for the technical report is a copy of the gcc cross compiler targeting the Nios II. It was installed from archive file `nios2gcc-20080203.tar.bz2`,

```
tar jxf nios2gcc-20080203.tar.bz2 -C /
```

This will extract the archive to `/opt/nios2`. `/opt/nios2/bin` contains all of the common gcc toolchain binaries.

Add to Path

Modify the file “`~/.bash_profile`”, changing the line

```
PATH=$PATH:$HOME/bin
```

to

```
PATH=$PATH:$HOME/bin:/opt/nios2/bin
```

This will add the installation directory of the nios II gcc cross compiler to the set of directories that the shell will automatically search executables for, so that one can run the tools without entering the complete installation path in addition to the binary name.

To verify everything is installed correctly, log out and then log back in, then type:

```
nios2-linux-gcc --version
```

which should then print out the following:

```
[fpga@fpgadev ~]# nios2-linux-gcc --version
nios2-linux-gcc (GCC) 3.4.6
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[fpga@fpgadev ~]# _
```

Install Required Utilities

Note that the below is specific to CentOS (along with other Redhat compatible Linux distributions, including Fedora and RHEL).

As root, run:

```
yum install gcc make ncurses-devel bison flex gawk lynx
```

This will install all of the tools required for building uClinux.

Next, it is necessary to add an additional repository location (where the “yum” update program searches for possible installations), by entering (all on one line):

```
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-2.noarch.rpm
```

For a detailed explanation, see ².

Then, install git:

```
yum install git-core
```

Extracting uClinux Distribution

The maintainers of the Nios II-specific codebase now use git, instead of SVN or CVS, to host the code base and updated patches. Git is a new file revision management system written by Linus Torvalds.

Extract git repository

The below commands assume that the user has mounted an external final system at /mnt/uclinuxdownloads that contains the file uClinux-dist-20080131.tar, hosted by Altera @ While in the home directory (“cd ~”), run the following:

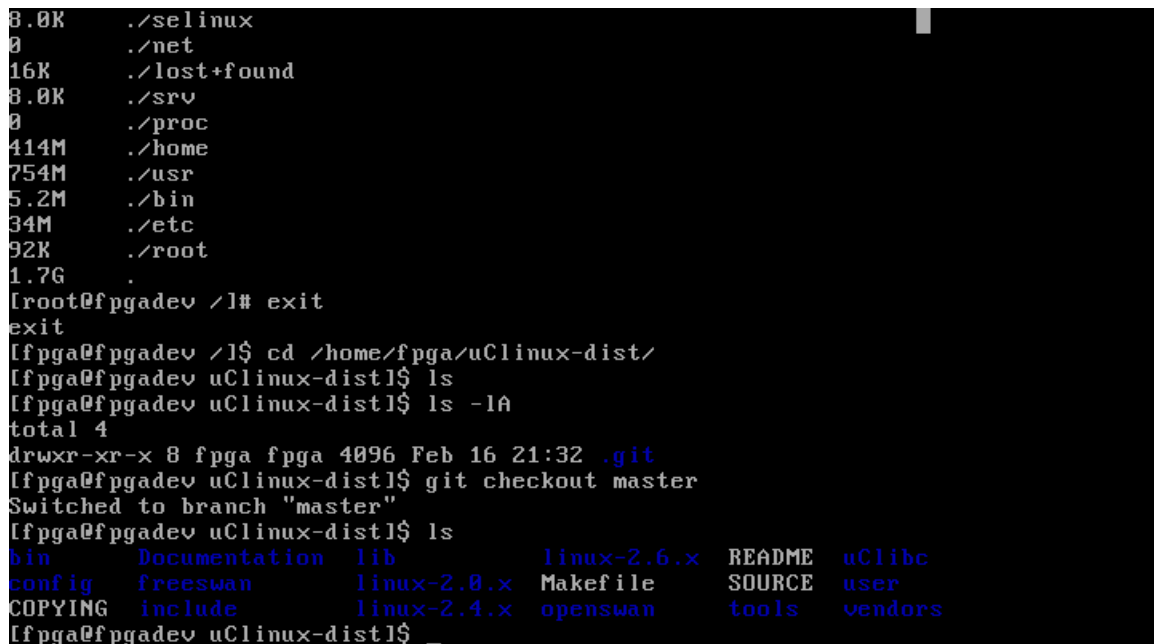
```
tar xf /mnt/uclinuxdownloads/uClinux-dist-20080131.tar
```

This will extract the uClinux build root into ~/uClinux-dist. Assuming you are logged in using the default “fpga” user, it will be located in /home/fpga/uClinux-dist.

Extract Main Branch

```
cd /home/fpga/uClinux-dist
git checkout master
ls
```

which result in roughly the below screenshot:



```
8.0K  ./selinux
0     ./net
16K   ./lost+found
8.0K  ./srv
0     ./proc
414M  ./home
754M  ./usr
5.2M  ./bin
34M   ./etc
92K   ./root
1.7G  .
[root@fpgadev ~]# exit
exit
[fpga@fpgadev ~]# cd /home/fpga/uClinux-dist/
[fpga@fpgadev uClinux-dist]# ls
[fpga@fpgadev uClinux-dist]# ls -lA
total 4
drwxr-xr-x 8 fpga fpga 4096 Feb 16 21:32 .git
[fpga@fpgadev uClinux-dist]# git checkout master
Switched to branch "master"
[fpga@fpgadev uClinux-dist]# ls
bin      Documentation  lib          linux-2.6.x  README      uClibc
config   freeswan       linux-2.0.x  Makefile     SOURCE      user
COPYING  include        linux-2.4.x  openswan     tools       vendors
[fpga@fpgadev uClinux-dist]# _
```

Configuring uClinux-dist for the first time:

After checking out the main branch, run the following from /home/fpga/uClinux-dist:

make menuconfig

```
uClinux v3.2.0 Configuration
-----+
                                Main Menu
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.
Legend: [*] built-in [ ] excluded <M> module < > module capable
-----+
Vendor/Product Selection --->
Kernel/Library/Defaults Selection --->
-----
Load an Alternate Configuration File
Save Configuration to an Alternate File
-----+
<Select> < Exit > < Help >
-----+
```

With “Vendor/Product Selection --->” selected, hit the Enter key, to load the menu:

```
uClinux v3.2.0 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.
Legend: [*] built-in [ ] excluded <M> module < > module capable
-----+
--- Select the Vendor you wish to target
(SnapGear) Vendor
--- Select the Product you wish to target
(LITE300) SnapGear Products
-----+
<Select> < Exit > < Help >
-----+
```

Now change the target Vendor to “Altera”. Note that there is only one target product, nios2nommu”, for Altera. The Vendor/Prodcut Selection screen should be:

```
uClinux v3.2.0 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.
Legend: [*] built-in [ ] excluded <M> module < > module capable
+-----+
--- Select the Vendor you wish to target
(A)ltera Vendor
--- Select the Product you wish to target
(nios2nommu) Altera Products
+-----+
<Select> < Exit > < Help >
```

After setting the vendor to Altera, select “Exit” and hit Enter to return to the Main Menu. Now from the main menu, enter the “Kernel/Library/Defaults Selection” menu, which initially is set to:

```
uClinux v3.2.0 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><Esc> to exit, <?> for Help.
Legend: [*] built-in [ ] excluded <M> module < > module capable
+-----+
(linux-2.6.x) Kernel Version
(uClibc) Libc Version
[ ] Default all settings (lose changes) (NEW)
[ ] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings (NEW)
[ ] Update Default Vendor Settings (NEW)
+-----+
<Select> < Exit > < Help >
```

Change “Libc Version” to None, and select “Default all settings (lose changes).” The menu Kernel selection menu should now appear like:


```

uClinux v3.2.0 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><Esc> to exit, <?> for Help.
Legend: [*] built-in [ ] excluded <M> module < > module capable
+-----+
(linux-2.6.x) Kernel Version
(None) Libc Version
[*] Default all settings (lose changes) (NEW)
[ ] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings (NEW)
[ ] Update Default Vendor Settings (NEW)
+-----+
<Select> < Exit > < Help >
+-----+

```

Now Exit out of the Kernel Selection menu, and then exit out of the main Menu. When prompted “Do you wish to save your new kernel configuration?” choose “Yes.”

After “make menuconfig” exits, a set of scripts will automatically run configuring the build root. When it asks “debug (CONFIG_USER_DEBUG_DEBUG)” choose y (for yes).

Customizing the Kernel:

Ethernet/Network Support

Add Driver for DE2’s DM9000 NIC

Device Drivers → Network device support → Ethernet (10 or 100Mbit) → DM9000 support.

Modify startup bootscript

Note that at bootup the kernel will execute the file / to initialize, by editing ~/uClinux-dist/vendors/Altera/nios2nommu/rc

USB Key (File system) Support

SCSI

Device Drivers → SCSI device support → Select “SCSI device support” and “SCSI disk support”:

```
Linux Kernel v2.6.19-uc1 Configuration

SCSI device support
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 <M> module < >

[ ] RAID Transport Class
[*] SCSI device support
[*] legacy /proc/scsi/ support (NEW)
--- SCSI support type (disk, tape, CD-ROM)
[*] SCSI disk support
[ ] SCSI tape support (NEW)
[ ] SCSI OnStream SC-x0 tape support (NEW)
[ ] SCSI CDROM support (NEW)
[ ] SCSI generic support (NEW)
[ ] SCSI media changer support (NEW)
--- Some SCSI devices (e.g. CD jukebox) support multiple LUNs
■(+)
```

<Select> < Exit > < Help >

Exit back to main Kernel configuration menu.

USB

Device Drivers → USB support → Select “Support for Host-side USB”, “USB device filesystem”, “ISP1362 HCD Support”, “USB Mass Storage Support”:

```
Linux Kernel v2.6.19-uc1 Configuration

USB support
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 <M> module < >

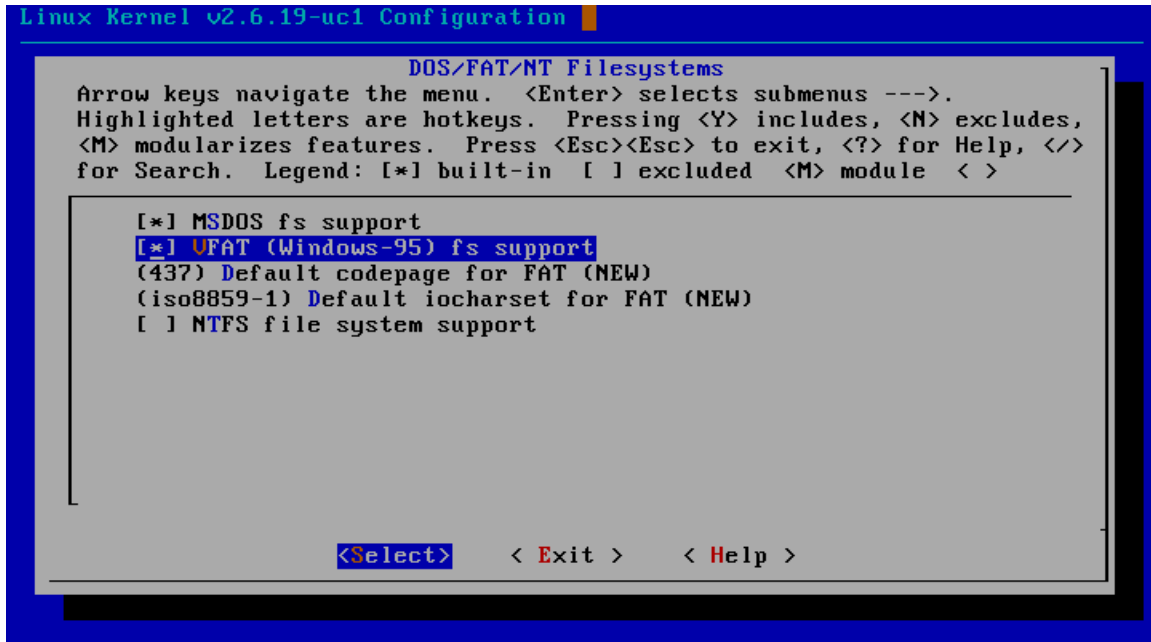
[*] Support for Host-side USB
[ ] USB verbose debug messages (NEW)
--- Miscellaneous USB options
[*] USB device filesystem
[ ] Enforce USB bandwidth allocation (EXPERIMENTAL) (NEW)
[ ] Dynamic USB minor allocation (EXPERIMENTAL) (NEW)
--- USB Host Controller Drivers
[ ] ISP116X HCD support (NEW)
[*] ISP1362 HCD support
[ ] SL811HS HCD support (NEW)
--- USB Device Class drivers
■(+)
```

<Select> < Exit > < Help >

(Note that “USB Mass Storage Support” is not shown but enabled further down in menu).

File System (including FS language) Support

File systems → DOS/FAT/NT Filesystems → Select “MSDOS fs support” and “VFAT (Windows-95) fs support):



Back out one level back to “File systems”, then → Native Language Support → Select “Codepage 437 (United States, Canada)” and “NLS ISO 8859-1 (Latin 1; Western European Languages)”.

VGA and Input (Keyboard, Mouse)

Kernel Configuration:

- Processor type and features
 - Platform drivers Options
 - Avalon VGA controller support
 - PS2 controller
- Device Drivers
 - Character Devices
 - Virtual Terminal
 - Disable “Support for console on virtual terminal”
 - Graphics Support
 - Console display driver support
 - Disable “VGA text console”
 - Disable “Framebuffer console support”
 - Input Device Support
 - Generic Input layer

- Mouse interface
- Keyboard
 - AT keyboard
- USB Support
 - USB Human Interface Device (full HID) support
 - HID input layer support

User Configuration:

- Libz/libjpeg
- Microwindows:
 - Optimize
 - Nanox
 - NanXdemo
 - NanoWM
 - Have file io/jpeg
 - Frame Buffer Display
 - Serial Mouse
 - Scan Keyboard
 - Nano-X
 - NanoWM
 - NXclock
 - NXterm
 - NXView

Configuring the Hardware

The example project was based on DE2_NIOS_HOST_MOUSE_VGA, an example project included with the DE2 System CD, located under DE2_System_v1.4b\

VGA and PS2 Keyboard Support

(**TODO CITE**), added following folders/files to main directory:

- Folders:
 - **altera_up_avalon_ps2**: folder containing PS2 component
 - **VGA_Controller_component**: folder containing VGA SOPC component
- Files:
 - **DE2_NIOS_HOST_MOUSE_VGA.v**: updated top level entity
 - **SDRAM_PLL.v**: updated PLL, changing SDRAM clock from 50MHz to 100MHz.
 - **system_0.ptf**: Updated SOPC description file, removing previous binary VGA controller, and adding new VGA controller and PS2 component.

Random to mention somewhere:

- names of components in SOPC must be exact to work, as drivers are expecting specific #define names.
- Different make methods, make user_only, make romfs
- Romfs vs. other filesystem options
- What actually happens with writing image to board at hardware level
-

Things to try:

- ccache
- mp3
-

¹ “About Buildroot.” Buildroot – Usage and documentation. <http://buildroot.uclibc.org/buildroot.html>

² “Using EPEL: How can I install the packages from the EPEL software repository?” EPEL/FAQ – Fedora Project Wiki. <http://fedoraproject.org/wiki/EPEL/FAQ#howtouse>