microtronix

Linux Development Kit

Getting Started Guide

Version 1.7

Table of Contents

System Requirements

The Linux Development Kit requires the following before beginning the installation:

1. Workstation (PC) with:
    -two serial COM ports available
    -Windows NT™ v4 or Windows 2000™

2. Altera Excalibur™ Development Kit featuring the Nios™ Embedded Processor (Version 2.0), with the GNUPro Tools and Cygwin™ installed.

3. QUARTUS II™ Programmable Logic Development Tools
    -This is only required if new devices are to be supported on the Excalibur Board that will require the Nios processor core to be rebuilt.

This guide assumes that the reader has basic knowledge of the Linux operating system and its conventions.

If you are new to the Linux operating system, it is recommended that one of the many available beginner or intermediate guides be purchased to give the necessary background information.

Introduction to the Linux Development Kit

The Linux Development Kit (LDK) includes an embedded version of Linux intended for processors without a Memory Management Unit (also referred to as MMU-less processors). The version of Linux distributed with the LDK is known as uClinux™, which Microtronix was responsible for porting to the Nios™ soft-core embedded processor. Also included are selected hardware components that have been developed to work with the Altera™ Excalibur™ Development Kit featuring the Nios processor. All of the tools and libraries necessary to supplement the Altera Software Development Kit (SDK) are also included to provide a complete development environment for building, loading, and debugging the Linux kernel. A group of user applications and examples completes the set.

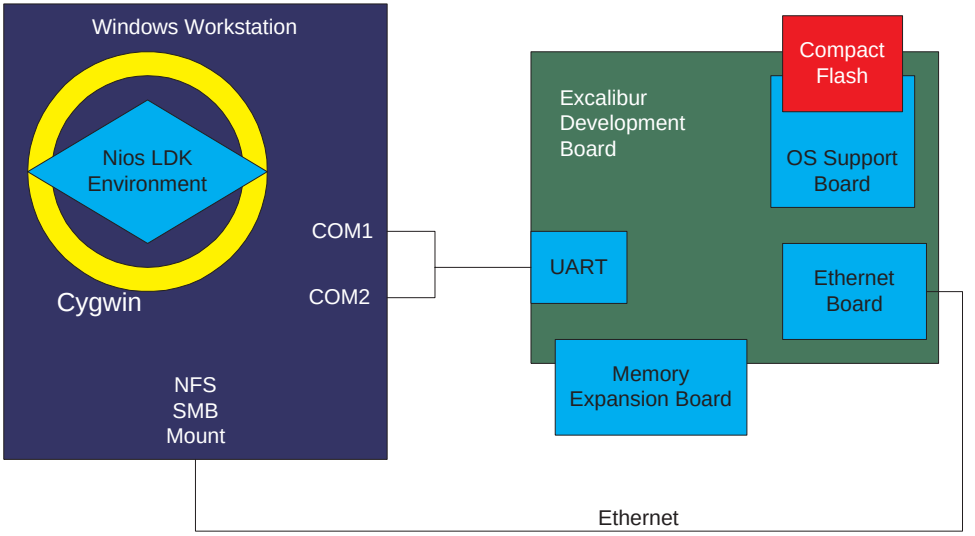
uClinux Overview

uClinux is an open source version of Linux modified to run on embedded processors lacking, or having limited, memory management capabilities -- such as the Altera Nios soft-core processor. Its memory usage is flexible enough to provide a small footprint with limited features, or it can be built as a feature-rich model.

The default configuration included in the LDK is medium size, providing TCP/IP networking, a DHCP client, a web server, telnet access, and both NFS and SMB clients for accessing Linux and Windows NT servers. These clients allow the development workstation to host executable application images, which makes the task of testing and debugging applications less cumbersome.

NOTE: Please see the README.TXT file located on the Software Support Package CDROM supplied with the Linux Development Kit for up to date information, including application notes.

Linux Development Kit Overview



Component Locations

Software Components

CompactFlash Card
Default root file system

Memory Expansion Card
Standard Kernel (in flash)
ROMDisk images (in flash)

Cygwin
GNUPro Tools
Nios LDK Environment

On Board Flash
Default Nios Processor Core
Development Nios Processor Core

Hardware Components

OS Support Board
Compact Flash Interface
IDE Header
SPI Interface:
  Real Time Clock
  Temperature Sensor

Memory Expansion Board
16MB SDRAM
8MB Flash

Ethernet Board
10BaseT Ethernet

The LDK is based on a Windows NT or 2000 user environment using the Cygwin tool chain. It is possible to set up a development environment in native Linux; please contact Microtronix for further details.

Linux Development Kit Overview

Working With Linux

If you have worked with a desktop version of Linux, you will be familiar with many of the conventions and utilities that you will find are a part of the Linux Development Kit environment. It is outside of the scope of this user guide to provide the background necessary to begin doing embedded development with Linux. Therefore, we recommend the following resources:

For more information on the uClinux kernel, visit:
www.uclinux.org

For more information on embedded Linux in general, visit:
www.linuxdevices.com
www.alllinuxdevices.com

For written resources on getting started with Linux, visit:
www.oreilly.com

For a free subscription to the Embedded Linux Journal (if you qualify), visit:
www.embeddedlinuxjournal.com

For updates to the Linux Development Kit documentation, visit:
www.microtronix.com

Linux Development Kit Overview

The Linux Filesystem in General

There are standard directories in any Linux filesystem that hold certain files. Not all of these directories need to be present (certainly not in your embedded system), and there may be others. Listed are the well known directories and a brief description of their purpose.

/bin: user commands are in this directory. This directory is part of the PATH statement that allows you (or your script files) to execute commands in this directory from any location.

/sbin: location of statically linked binaries (generally 'non-user' system tools).

/opt: this is the standard directory space for applications.

/mnt: this directory refers to temporarily mounted file systems. On a desktop system this might refer to a floppy or CDROM drive. In the LDK it is used for SMB and NFS mount points, described later on in this guide under the section entitled "Running Applications From A Server".

/lib: Runtime libraries are located in this directory.

/dev: the references in this directory represent physical devices, such as the serial port.

/usr/src/linux: the Linux kernel sources are located in this directory.

/etc: applications that use configuration files will usually place the files here. In addition, uClinux start up scripts also reside here.

Linux Development Kit Overview

File System Structure and the LDK

It is important to remember that there are actually two file systems you will need to keep track of. One of these is the file system that resides under your Cygwin environment. This is the environment in which you will do your development. When you install the LDK for example, a subdirectory under the root Cygwin directory will be created called opt. This subdirectory will house all of the specific LDK files on your host machine.

The other file system is known as the target file system. This is the file system that uClinux will use when running on the Nios Excalibur Board. This file system can be loaded into the flash memory of the Memory Expansion Board or onto the Compact Flash card which is used with the Operating System Support Board.

Getting Started - Hardware Components

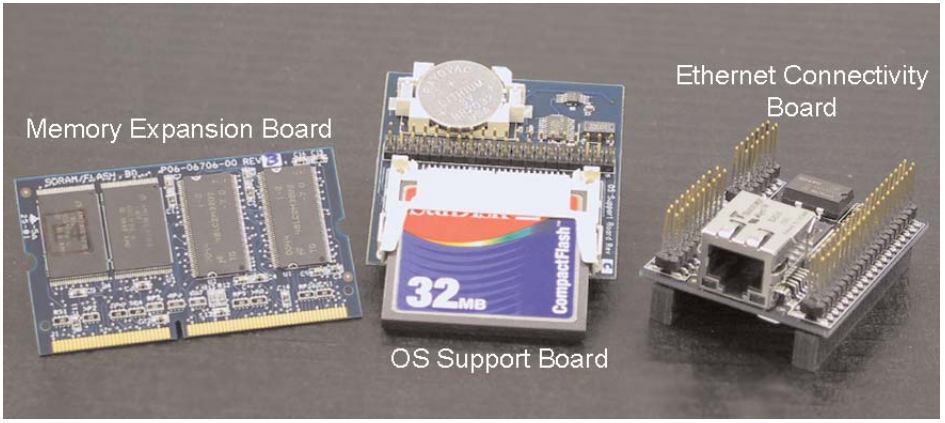
The Linux Development Kit contains:

- a Memory Expansion Board
- an Operating System Support Board with Compact Flash Card
- an Ethernet Connectivity Board.

Also included with the Kit is:

- a Software Support Package CD
- two ethernet patch cables (one straight-through, one crossover cable)
- a serial Y cable.

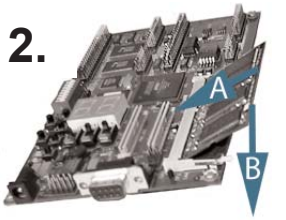
In the event that you have not received any of these components, please contact the Altera Applications Support Hotline at the number listed on the back of this Getting Started Guide.



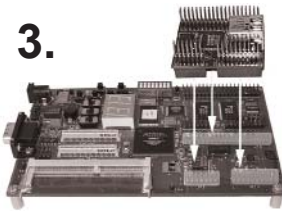
Getting Started - Hardware Installation

1. Power down the Excalibur Board, after verifying that the board is configured with the default factory core flash image.

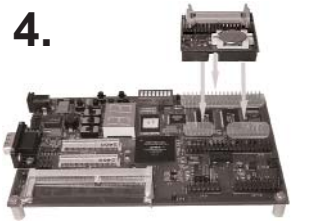
2. Plug the Memory Expansion Board into the SODIMM socket J2 by placing the card in the slot at the angle demonstrated in the figure to the right (A). When the card is inserted at this angle, press down lightly to lock it in place (B). You should hear a small click once the board has been secured.



3. Plug the Ethernet Connectivity Board onto the 3.3V Expansion Prototype Connector by lining up the sockets and pressing the board vertically downward. This connector consists of headers JP8, JP9, and JP10.



4. Plug the Operating System Support Board onto the 5V Expansion Prototype Connector by lining up the sockets and pressing the board vertically downward. This connector consists of headers JP11, JP12, and JP13. Be sure to plug the Compact Flash card into the Operating System Support Board's socket if this hasn't been already done.



5. If the Ethernet Connectivity Board is to be connected to a hub then the Ethernet Patch Cable should be used. If the Ethernet Board is to be attached directly to another ethernet card, the Ethernet Cross-Over Patch Cable should be used.

6. The Serial Y-Cable is used to connect two UARTs built into the core to two seperate serial COM ports on a PC. The male end should be connected to the serial port connector J3 on the Excalibur Board. The female end, labelled COM1, is the main console serial port, and it must be connected to the PC's COM1 port before any communication with the board can occur. The other female end (labelled COM2) is meant to be used for debugging with GDB. You only need to connect this port if you plan to perform debugging tasks.

7. Power on the Excalibur Board.

Getting Started - Software Installation

Installing the Linux Development Kit

1. Ensure that the "Software Support Package" CD is placed in an available CD-ROM drive. You should have Adobe Acrobat Reader installed to view all documentation files.

2. Using Windows Explorer, navigate to the CD-ROM drive in which the "Software Support Package" CD was placed. From there, run "setup.exe" by double-clicking it (please ensure that there are no other programs or windows open before you do this). This will launch the LDK Installer. Simply follow the on screen directions until the Installer completes its task. (Note: the LDK should be installed in the Cygwin directory. The Installer will ask for you to confirm the directory into which Cygwin was previously installed).

The Installer will copy a shortcut, **Nios_2_0_LDK**, onto your desktop that will allow you to invoke the LDK bash environment.

3. The Nios LDK environment is now set up. When developing using the LDK you should run bash configured with this new environment using the desktop shortcut, **Nios_2_0_LDK**.

Getting Started - Software Installation

In order to support the devices now plugged into the Excalibur Board, the Nios core needs to be updated with the one supplied in the LDK. The following steps must be taken to load this new core into the flash memory located on the Excalibur board.

1. Ensure that the serial Y-Cable is connected between the Excalibur port and COM1 of the PC.

2. Invoke the LDK bash window by double-clicking the **Nios_2_0_LDK** icon on the desktop.

3. Make sure GERMSmon is running on the board and load the new core into flash by entering the command (all on one line):

```
[Nios2.0 LDK Bash]..uClinux/: nios-run
                    LDKcore2.hexout.flash
```

4. After **nios-run** indicates that the download has completed successfully, restart the board by powering the board off and on again.

The prebuilt image of uClinux and a copy of the root file system (the ROMDISK image) are pre-loaded onto the Memory Expansion Board's flash memory. The new GERMSmon will look for an executable image in this flash when it is restarted. **nios-run** (which has remained in terminal mode) will now display the messages from the uClinux startup. The default root file system is pre-loaded onto the CompactFlash card.

5. In order to log into uClinux, at the login prompt any user name can be used. However, the password that must be entered is **uClinux**

If you need to find out the IP address of the board, use the **`ifconfig`** *command at the Nios shell prompt.*

When uClinux is initialized it is set up to look for a DHCP Server by default. However, if a DHCP Server is not available, an error message will be displayed. To manually set the IP address and other variables, you must use the **`ifattach`** *command. A description of how to use this command is under the "Setting an IP Address" heading in the "Running Applications From a Server" section in this guide.*

Creating New Applications

Creating a new application is a relatively straightforward task, as most of the build process has been automated. The steps are as follows:

1. Create a directory under the **root** of the application tree (**/opt/uClinux/niosuserland**). Example:

```
[Nios2.0 LDK Bash]..//: cd /opt/uClinux/niosuserland
[Nios2.0 LDK Bash]../niosuserland/: mkdir myNewApp
```

3. Peruse the **niosuserland** tree and find a Makefile that most closely matches your requirements. For a simple application, consider the makefile in **hello/Makefile**. A more complex example can be found in **route/Makefile**. This makefile builds a library and an executable using that library. Edit the makefile to reflect your application source files. Note that the default target for executables is **/opt/uClinux/romdisk/bin**.

NOTE: ***Make sure your new application's Makefile contains the line:***

```
include ../Rules.mak
```

This is very important as the file Rules.mak contains many of the rules and variable definitions required for applications to be built properly for use with the Nios.

If your needs differ (for instance, if you wish to place your executable onto an SMB server), edit the **FTARGET** variable in your makefile to reflect the proper path, for example: **/opt/uClinux/niosuserland/bin** as in **hello/Makefile.** Refer to the "Running Applications From a Server" section for more details.

4. To build your application, you can either run make from its subdirectory or use the command "**make all -C *ApplicationDirectoryName***" from the **niosuserland** directory. Alternatively, you can follow the steps defined in this guide under the section "Rebuilding the niosuserland Applications."

Points to Keep in Mind

1. The top-level Makefile and Rules.mak do most of the work for you, allowing your makefiles to be relatively simple.

2. Try to reuse an existing makefile whenever possible.

Rebuilding the niosuserland Applications

The source code for the included applications reside in the directory **/opt/uClinux/niosuserland/**. In this directory, there is a top level makefile that builds all of the applications which reside in the various subdirectories.

To force a rebuild of all applications, use the following command:

```
[Nios2.0 LDK Bash]..niosuserland/: make clean
```

This is only necessary if the uC-libc library or the Linux header files have changed.

To build a new application or rebuild applications whose source files have changed, use the following command:

```
[Nios2.0 LDK Bash]..niosuserland/: make all
```

Alternatively, individual applications can be built using the -C argument to make. For example, if you wish to do a make clean and make on only the ping utility, you can issue the following commands:

```
[Nios2.0 LDK Bash]..niosuserland/: make clean -C ping
[Nios2.0 LDK Bash]..niosuserland/: make all -C ping
```

These commands have the same effect as changing the current working directory to the application's subdirectory and doing a make clean and make from there.

The **make** command copies the resulting executables to:

```
/opt/uClinux/romdisk/bin/
/opt/uClinux/romdisk/sbin/
```

This is done so that the applications will be included in the ROMDISK image when it is rebuilt (refer to the section "Rebuilding the ROMDISK" for information about this procedure).

Rebuilding the niosuserland Applications

Executables may also be copied to a subdirectory outside of the ROMDISK to be run from the target via SMB or NFS. Refer to the section “Running Applications From a Server” for details. The usual directory is:

```
/opt/uClinux/niosuserland/bin/
```

The individual application makefiles dictate to which directory executables are copied.

Rebuilding the ROMDISK

The Nios LDK comes with a pre-built romfs image (ROMDISK) programmed into the flash memory of the Memory Expansion Board. The loadable image file is located at:

```
/opt/uClinux/romdisk.flash
```

The ROMDISK can be used as the root file system (although by default, the ext2 file system on the Compact Flash Card is used as the root), and includes various device files (nodes) and application executables. The ROMDISK is an image made from an existing directory tree which resides under:

```
/opt/uClinux/romdisk/
```

Therefore, any new files that are to be included in the ROMDISK file system should first be placed under this directory. The following command should then be run to rebuild the ROMDISK image file from that tree:

```
[Nios2.0 LDK Bash]../uClinux/: ./mkflashromfs
```

The mkflashromfs script will create the new romdisk.flash, which contains the necessary commands for erasing the flash reserved for this purpose, and the romdisk image itself. The image can then be loaded onto the Excalibur Board's flash memory. The procedure for accomplishing this is:

```
[Nios2.0 LDK Bash]../uClinux/: nios-run romdisk.flash
```

Note: If there is already a Linux image running on the board, you must press and hold the SW4 button and press the RESET button (SW2) before using this command.

This will program the image into flash. **nios-run** will remain in terminal mode to allow testing of the new romfs. If the Linux kernel is already loaded into flash, it should start up automatically; otherwise press the CLEAR (SW3) button to restart the kernel.

Building the IDE File System

This section will guide you through the process of creating a new file system on the CompactFlash Card device. It should be noted that by default, the file system on the CompactFlash Card is used as the default root file system. The information presented in this section assumes that you have changed this setting (you are using the ROMDISK as the default root file system; see the "**readme**" file in the **/opt/uClinux/tools/cmdline_edit/** directory for more details).
More information regarding the file system utilities can be found elsewhere, as they are standard, well documented Linux applications.

Before making any modifcations to the Compact Flash Card, make sure that it is not mounted. You can unmount the device using the following command:

```
# umount /dev/hda1
```

To create a new file system on the Compact Flash Card, you should first invoke **fdisk**, which is a utility that allows you to manage partitions on a specified device:

```
# fdisk /dev/hda
```

This will run **fdisk** on the **hda** device (the Compact Flash Card) and bring you to fdisk's main menu from where a number of operations can be performed. It may be necessary to delete an existing partition from the device to make room for the new partiton.

- From the main menu, use the **d** command to delete a partition.
- Next, the **n** command can be used to add a new partition.
    - Use **p** to select the primary partition.
    - fdisk will then prompt you to pick a partition number. If you are creating a new root file system, you should pick **1**, then use the appropriate starting and last cylinder number (the starting and last cylinder numbers will determine the size of the resulting partition).
- You make the changes permanent, use the **w** command. This will write the changes you made to the partition table and exit fdisk. If you wish to discard the changes you made, use the **q** command at the main menu.

The CompactFlash Card should now contain a Linux ext2 primary parition.

Building the IDE File System

Next, the newly created partition should be formatted. This can be accomplished using the following:

```
# mke2fs -s0 -Onone -LuClinux /dev/hda1
```

NOTE: ***the "1" of /dev/hda1 specifies the primary partition, the one you just created.***

After the partition is formatted, it can then be mounted using the following command:

```
# mount /dev/hda1 /mnt/ide0
```

NOTE: ***the second argument to the mount command (in this case, /mnt/ide0) is the point at which to mount the partition. This can be changed to any mount point as necessary.***

The final step to take is to populate the file system with the necessary files. If you are creating a root file system, the shell script **mkidedisk** can be used. This script will create the necessary directory tree, and device node files. You can then copy any files you wish to include in the root file system into the tree. If you are not created a root file system, the desired files can simply be copied onto the device.

Rebuilding the Linux Kernel

The Nios LDK comes with a pre-built Linux kernel, named **linux.srec**, located in the **/opt/uClinux/linux** directory. This is a compiled, ready to run kernel that can be loaded onto the Excalibur Board's RAM.

However, if changes are to be made to the Linux kernel code, this srec file must be rebuilt and reloaded onto the board. Rebuilding the kernel can be accomplished by using the following commands:

```
[Nios2.0 LDK Bash]..linux/: make clean
[Nios2.0 LDK Bash]..linux/: make
```

Where:

make clean deletes all executable and object files.This ensures that existing object files are not reused.

make will actually compile the source and build the final srec file. This is the only step required if changes have been made to existing source files and the dependencies and configuration have not been modified.

Other make options are:

```
make mrproper
make config
make dep
```

make mrproper deletes all executable, object, configuration, and dependency files. This ensures a fresh start for building the kernel.

make config will allow the desired kernel options to be selected and used. This creates the configuration files used to control the build process.

make dep will create the dependencies. This step is mandatory if a source or header file has been added to the kernel tree or if include directories have been added or removed. This step also needs to be run if a **make mrproper** has just been run, since **make mrproper** deletes dependencies.

Rebuilding the Linux Kernel

After building a new kernel, the **linux/linux.srec** file is suitable for downloading into RAM memory using **nios-run**, for example:

```
[Nios2.0 LDK Bash]../linux/: nios-run linux.srec
```

To make a more permanent image of the kernel that will start up automatically when power is applied or the CLEAR button (SW3) is pressed, the image can be loaded into flash. The procedure for accomplishing this is:

1. Prepare a loadable file in the uClinux directory:

```
[Nios2.0 LDK Bash]../linux/: cd ..
[Nios2.0 LDK Bash]../uClinux/: ./mkflashlinux
```

This creates **linux.flash**, which contains the necessary commands for erasing the flash reserved for this purpose, and the kernel load itself.

2. Program the flash:

Note: If there is already an existing Linux image running, you must press and hold the SW4 button, then press and release the reset button (SW2) before using this command.

```
[Nios2.0 LDK Bash] ..uClinux/: nios-run linux.flash
```

This will put the image into flash and begin execution. **nios-run** will remain in terminal mode, and the Linux startup messages will be displayed.

Running a Custom Core

If you wish to use a core other than the one provided with the Nios LDK, you will need to update some of the header files and then rebuild the uClinux kernel, uC-libc, and niosuserland applications.

The Nios LDK includes a script file that automates the procedure of updating the necessary header files. This file is located in the /opt/uClinux/ directory. To use it, type the following command:

```
[Nios2.0 LDK Bash] ..uClinux/: ./core_update <path to
    Quartus project directory>
```

NOTE: ***Be sure* not *to use a trailing "/" after the path of the Quartus project directory.***

For example, you could run the core_update script passing the project directory provided on the Nios LDK CD-ROM as an argument:

```
[Nios2.0 LDK Bash] ..uClinux/: ./core_update
    /cygdrive/d/Quartus Project/Hardwarev_3_0_base/cpu_sdk
```

After running this script, you should rebuild the uClinux kernel, being sure to use the

```
make mrproper
make config
make dep
make
```

commands. You should also rebuild both uC-libc and niosuserland and then reload the kernel and applications onto the Excalibur Board.

Running Applications From a Server

The default kernel configuration shipped with the LDK includes both SMB and NFS support. This will allow you to mount a remote SMB share or NFS mount point. If you remove either of these and rebuild the kernel you will not be able to mount the respective remote system. Because of the convenience of being able to load and execute an application from a remote system where the applications are being built it is recommended that SMB and NFS be left in the kernel until application debugging is complete.

SMB

For SMB you will need to share the drive or folder that you want to access from the Linux Development Kit. For information on doing this follow the Windows Operating System instructions.

NFS

For NFS you will need to install the NFS server software. You will also need to specify which file system to export using NFS. For information on doing this, follow the NFS HOWTO document, that can be found at **http://www.linuxdoc.org/HOWTO/NFS-HOWTO**.

Once you have the remote file system mounted the applications can be built and tested without having to rebuild the romdisk. This provides a significant time benefit.

Setting Up an IP Address

Although the ethernet controller is automatically assigned an IP address when uClinux starts up, you can manually assign it an arbitrary address using the **ifattach** command:

```
ifattach  --addr <local_IP_address> --mask <subnet_mask>
          --net <network_id> --gw <gateway_address>
          --if <device>
```

Example:

```
# ifattach --addr 10.1.2.9 --mask 255.255.255.0
      --net 10.1.2.0 --if eth0
```

Running Applications From a Server

Mounting an SMB Remote File System From uClinux

To SMB mount a remote shared drive or folder you will need to know the server name, share name, server IP address and the password (if one was assigned). You will also need a mount point which was either created on the rom file system or in the ram disk. For example:

To mount the following system:

Server Name:	**WINDOWS**
Share Name:	**c**
Password:	**MyPass**
IP Address:	**10.1.1.50**

at mount point: **/mnt/smb**

You would type (on one line):

```
# smbmount //WINDOWS/c /mnt/smb -I 10.1.1.50
      -U john -C -P MyPass
```

Mounting an NFS Remote File System from uClinux

To NFS mount a remote NFS server mount point you will need to know the exported file system and the NFS server IP address. For example:

To mount the following system:

Server Name:	**LinuxBox**
Exported File System:	**/opt**
IP Address:	**10.1.1.52**

at mount point: **/mnt/nfs**

You would type:

```
# mount -t nfs 10.1.1.52:/opt /mnt/nfs
```

The Default Root File System

Directories and Files in the Default Root File System

By default, the LDK is configured to use the CompactFlash Card as the default root file system.

To view the default root file system's structure, examine the /opt/uClinux/romdisk subdirectory. Everything in and below this directory is present by default in the default file system (on the CompactFlash Card).

Directories in the Default Root File System

`/bin /dev /etc /home /mnt /proc /sbin /usr /var`

These directories serve essentially the same purpose as in any standard Linux distribution. However, because this is an embedded system, there are some differences.

First, there are no user profiles established. This is why it doesn't matter which user name you use when you login into the uClinux environment. Every time a login is made, it is at the system adminstrator level (or "root" as it is referred to in Linux).

Secondly, although this is a "default" file system, when you start to work on specific applications for your board, you will likely wish to customize this file system to reflect the most efficient use of space. In the prebuilt system we've provided, all of the niosuserland/ applications are included by default, however there will probably be a lot of applications, files, and directories that can be removed to tailor the environment to your specific requirements.

ROMDISK

The ROMDISK is a pre-built file system (the pre-built image is included in the LDK as /opt/uClinux/romdisk.flash) that can be loaded into the flash memory and used as the default file system (however, by default the CompactFlash Card is used as the default) The LDK comes with a sample ROMDISK already pre-loaded into the flash memory of the Memory Expansion board. If you wish to use this image as the default root file system, please see the "**readme**" file in the /opt/uClinux/tools/cmdline_edit/ directory.
This file system has read only access, however you can build a custom ROMDISK

by modifying the files and directories in the /opt/uClinux/romdisk tree and then building a flash image of this tree. The new image would then need to be loaded onto the Memory Expansion Board. For more information on building a custom ROMDISK, please refer to the "Rebuilding the ROMDISK" section in this guide.

IDE

The kernel that comes preloaded into flash memory is set up to use the IDE File System (CompactFlash Card) as the default root file system. However, this setting can be changed when rebuilding the kernel or by using the **cmdline_edit** utility (see the "**readme**" file in the /opt/uClinux/tools/cmdline_edit/ directory).
For more information on the IDE file system, please refer to "Rebuilding the IDE File System" section in this guide.

Navigating the Default Root File System

Listed here are some important files and their locations:

bin/
Binaries for most of the utility applications ported to the Nios Processor reside here.

etc/
Any configuration files used in your uClinux environment are located here. Make sure that you understand any changes being made before attempting to modify or delete any files in this directory. For example, in the /config subdirectory you will find the configuration file for the Boa Web Server (boa.conf). Usually by referring to the text of a configuration file you will find which application the file is related to, and instructions on the proper method of modifying it.

sbin/
Applications used by the system (such as the Boa Web Server) are here.

dev/
All devices accessed by Linux are represented by a file. In this directory you will find references to specific devices that are available to your uClinux environment. Before attempting to modify or delete any files, it is important that you understand what is being changed!

Debugging Applications

The LDK only supports remote debugging; there is no self-hosted debugger available. Remote debugging consists of a debugger stub on the target (which is built into uClinux), and a host debugger on your workstation.

To debug an application, you should first ensure that you have a gdb.ini file in the subdirectory from which nios-elf-gdb will be invoked (This should be the application's subdirectory). This file should contain information that will allow nios-elf-gdb to properly set up. The contents of the file should look like this:

```
set architecture nios32
file /opt/uClinux/linux/linux
set remotebaud 115200
target remote com2
```

Next, invoke the debugger stub from a console terminal by preceeding the application command line with a "debug". For example, if you wish to debug the "ping" program, you would type:

```
# debug ping 10.1.1.23
```

At this point, you should start up your nios-elf-gdb:

```
[Nios2.0 LDK Bash] ../: nios-elf-gdb
```

The stub will then display some information on the system console similar to the following:

```
GDB: Trap 5 at 0x????????
GDB: Enter the following command in the nios-elf-gdb
      console window:
GDB:add-symbol-file ping.abself 0x????? 0x????? 0x?????
```

Debugging Applications

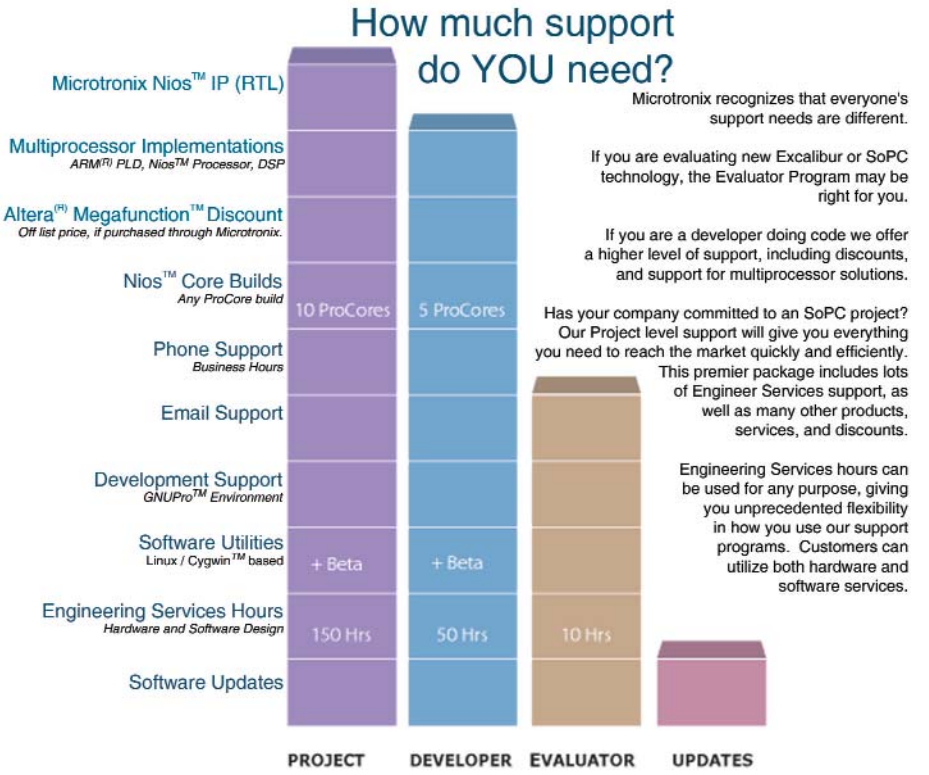
Open up a console window in GDB. If the terminal emulator you are using supports it, copy the "add-symbol-file ping.abself ..." command string from the console terminal into the console window of the GDB (you will need to substitute the actual values shown for the ?'s). If your terminal emulator does not support copying and pasting from the terminal window, you will have to type this command line manually into the GDB console window.

NOTE: ***you may have to change the path to the abself file as appropriate.***

You can now use GDB to step from `crt0` into `__uClibc_main()` and then into your applications `main()`.

Annual Support Contracts

Microtronix offers many options for technical support according to your needs. Please contact sales@microtronix.com for more information.



If you require assistance installing your LDK, please email Altera Technical Support:

support@altera.com

If you would like to purchase an annual support contract, or if you are interested in any of the following services:

Device Driver Development
Hardware Design, Layout, and Manufacturing
Linux Kernel Development
Programmable Logic Design and/or Prototyping
IP Integration
Embedded Systems Services

Please email ldk@microtronix.com or visit our website (www.microtronix.com) for more information.

Microtronix Datacom Ltd.
120 Bessemer Road
London, ON, CA
N6E1R2

www.microtronix.com

951-6000-01-B