Third Prize

An Internet-Based Smart Terminal

Institution: **Shanghai Jiao Tong University**

Participants: **Shanwen Zhang, Jie Zhang, Faheng Zang**

Instructor: **Zhigang Zhang**

Design Introduction

Households, schools, and administrative organizations are using more and more electrical equipment. With the increase in additional equipment, management becomes an issue. Because the equipment is usually maintained and managed in wired mode, extra investment will be needed specifically for equipment management. Our project considers using wireless mode to organize and manage equipment and centrally control it using the Internet. Internet access has become a standard interface for families and offices. Therefore, with a wireless control terminal and a cost-efficient receiver for each device, one monitor can connect to and display information about electrical appliances (such as electric lights, air conditioners, etc.), saving enterprise administration costs. Our product is applicable to large office buildings, high-end residential buildings, administrative enterprises, smart villas, schools, hospitals, etc. that have a lot of electrical equipment to be managed.

The system is very scalable. Users can continuously add new equipment and functions, as long as the terminal firmware and control software is updated. For example, this system can deliver video and broadcast to each unit using a high-bandwidth LAN, making it a powerful method for ad broadcasting in office buildings. Additionally, a touch screen can provide users with more public information.

If electronic equipment vendors accept this scheme as the standard for equipment intercommunication, preset communication interface, information technology will increase user satisfaction.

This system uses the Altera® Development and Education (DE2) development board as the terminal control core. Embedded system-on-a-programmable-chip (SOPC) technology conveniently integrates Ethernet, SDRAM, a programmable I/O (PIO) interface, digital-to-analog (D/A) switch interface, and a universal serial bus (USB) controller into the system, which greatly contributes to the system implementation. With the µClinux cross-compilation development environment, we can immediately begin work, which greatly enhances efficiency. By migrating a clipped µClinux system, managing the development board resources is very easy and convenient, which reduces development problems and shortens the development cycle.

Function Description

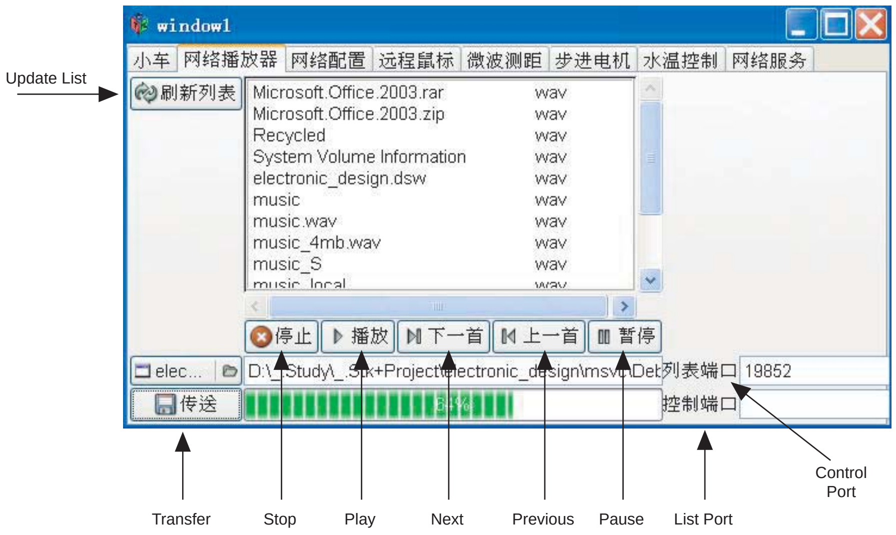
Our project is an experiment to provide proof of concept for smart network equipment. To this end, we completed two functions in this design:

- Network **.wav** music playback
- Remote mouse

Network .wav Music Playback

This functional module sends files from a PC to the DE2 board. The DE2 board stores the files into a mobile hard disk. The PC can obtain music files from the music folder and select any file to play on the DE2 board. See Figure 1.

Figure 1. Network .wav Music Player



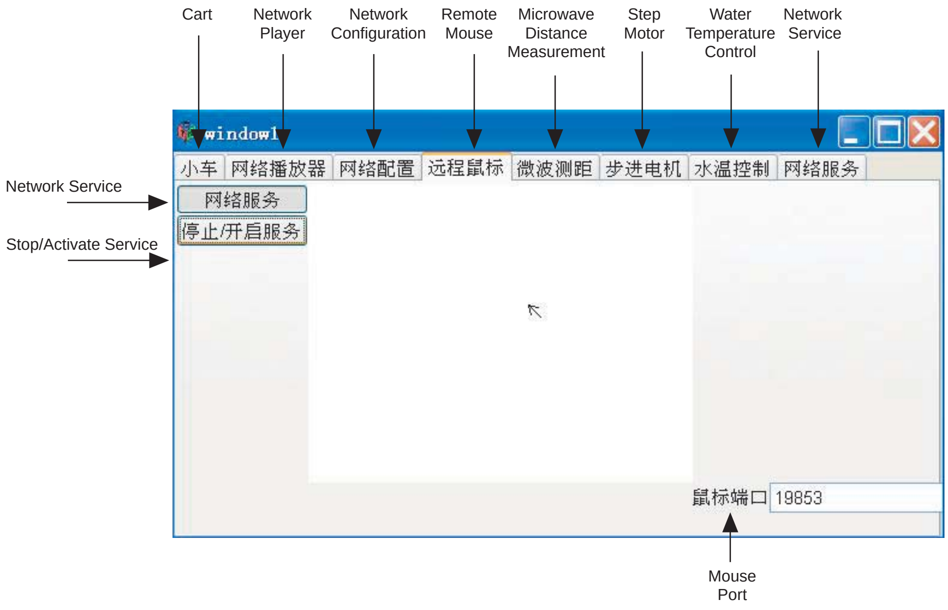
```
Microsoft.Office.2003.rar        wav
Microsoft.Office.2003.zip        wav
Recycled                         wav
System Volume Information        wav
electronic_design.dsw            wav
music                            wav
music.wav                        wav
music_4mb.wav                    wav
music_S                          wav
music_local                      wav
```

```
elec...   D:\_.Study\_.Stk+Project\electronic_design\msvc\Det   列表端口 19852
```

Remote Mouse

We can connect a USB mouse to the DE2 board's USB controller. The mouse pointer then appears on the PC and is controlled through the DE2 board. See Figure 2.

Figure 2. Remote Mouse



Performance Parameters

This section describes the performance parameters of our design.

Music Playing Performance

We broadcast a **.wav** file in our design. To play the music, the design writes 16-digit pulse-code modulation (PCM) codes into an audio FIFO buffer. This method is simple and direct, but has the following weaknesses:

- Large **.wav** files are inconvenient for network transmission.
- The files use excessive memory resources, causing low memory utilization.

Because of these limitations, we plan to use MP3 files in future designs. However, our current project is an experiment to demonstrate and materialize the concept of smart network equipment. Therefore, **.wav** files are fine for this project.

During our experiments, we found inferior sound quality and a noisy background. Through analysis, we determined that these issues are caused by the **.wav** file playback algorithm, i.e., cyclic query. Due to assimilated processing capability of **nios2kernel**, if we used the CPU query algorithm the design would overutilize the CPU and affect system processing. We will need to optimize this function in the future.

When we use an FTP server to upload files, the DE2 board's fastest receiving speed is 300 k/second, which is also the DE2 board's limit for processing data. The **.wav** file speed is 1,411 bps (or 176 k/second) and the design consumes some system processes and network services, making playback difficult to complete in software query mode. Therefore, this mode occupies too many processor resources, which is a problem we should address first in our future design work.

Remote Mouse Performance

The system can successfully collect real-time mouse movements and clearly the simulate mouse movement on a PC.

Design Architecture

The goal of our project is to convert equipment without information processing capability into an Internet node using a small communication module. Using powerful Internet interactivity, we can use and control the equipment in centralized management mode, simplifying office communication lines. Figures 3 and 4 show the block diagrams for our design.

Figure 3. Smart Terminal Block Diagram

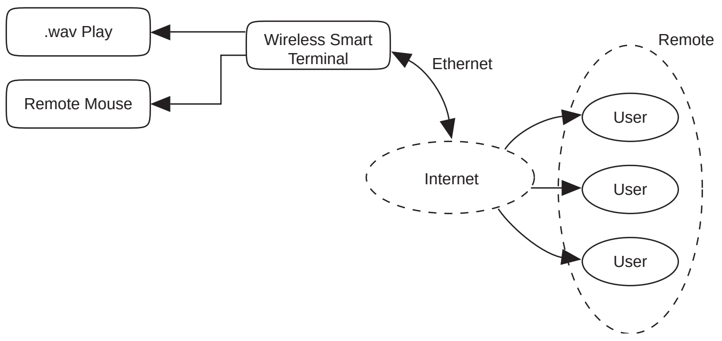
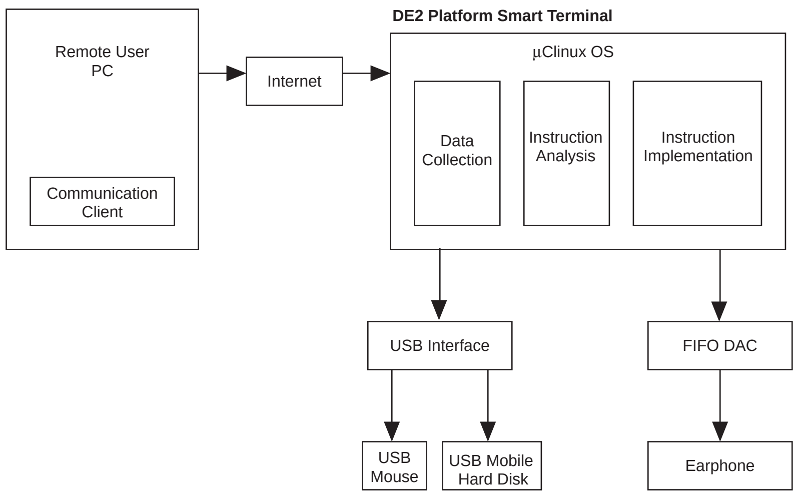


Figure 4. Functional System Block Diagram



The remote module uses special communication software to communicate with the DE2 board via the Internet and sends control instructions. The DE2 board obtains remote messages via Ethernet, interprets the message packet into specific instructions, and sends them out using a wireless transmission device. Equipment within the system's coverage analyzes the instruction upon receiving it and confirms whether it is the destination device for the instruction. If yes, it carries out the instruction, implementing remote control and achieving our goal. The entire system has two core modules:

- An intelligent, integrated, personalized remote control (i.e., the remote equipment management software)
- Networking capability, functional expandability, reliability, and low power consumption from the DE2 board at the smart terminal

The DE2 board has abundant resources to meet the design demands, and we expect them to be expanded.

Design Methodology

The following sections describe our design methodology.

Migrating µClinux to the DE2 Board

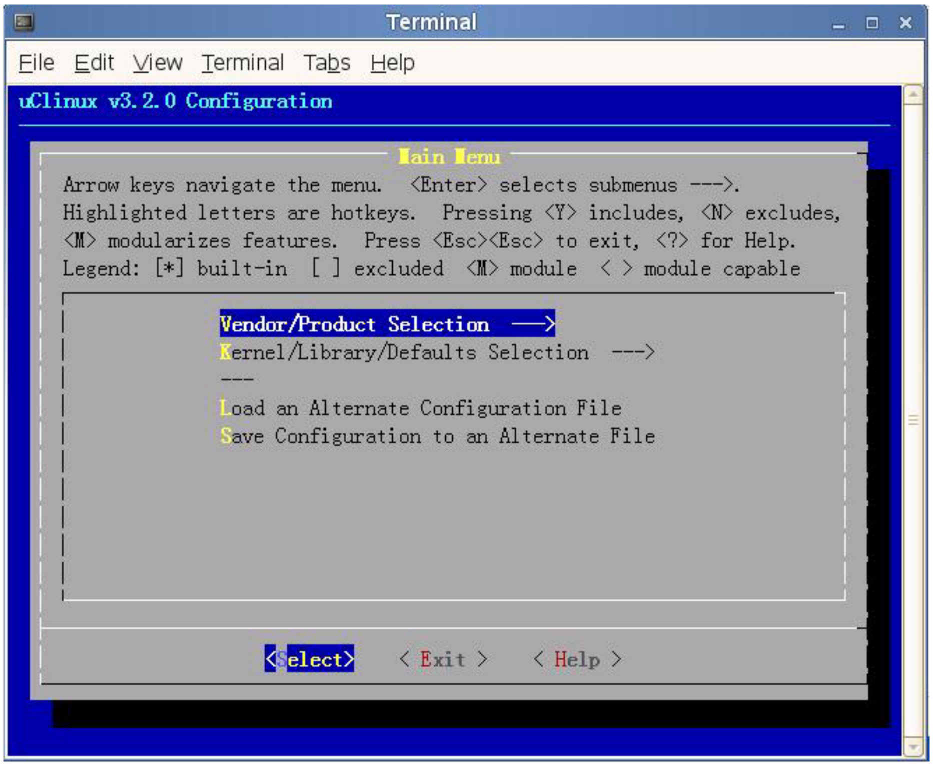
The µClinux system is a reduced Linux kernel without a memory management unit (MMU); we used a version 2.6 kernel. First we downloaded and installed the kernel:

1. We downloaded the µClinux kernel source code from the Internet at **http://www.uclinux.org/dls.uclinux.org/uClinux-dist-20070130.tar.bz2**.
2. We downloaded a µClinux kernel modification file from the Internet at **http://nioswiki.jot.com/WikiHomeOperatingSystems/µclinux/UClinuxDist/uClinux-dist-20070130-nios2-02.diff.gz?cacheTime=1170915312300**.
3. We downloaded a cross-compilation environment file from the Internet at **http://nioswiki.jot.com/WikiHome/OperatingSystem/µClinux/BinaryToolchain/nios2gcc.tar.bz2**.
4. We installed the cross-compilation environment, after which we could compile the kernel.

We used the following steps to compile the kernel:

1. At a command prompt, we typed `/make menuconfig` in the **UClinuxDist** directory, which opened the compilation interface shown in Figure 5.

Figure 5. Compilation Interface



```
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 >
```

2. We selected the **Vendor/Product Selection** option.

3. For the **Select the Vendor you wish to target** option, we chose **(Altera) Vendor**.

4. For the **Select the Product you wish to target** option, we chose **(nios2nommu) Altera Products**.

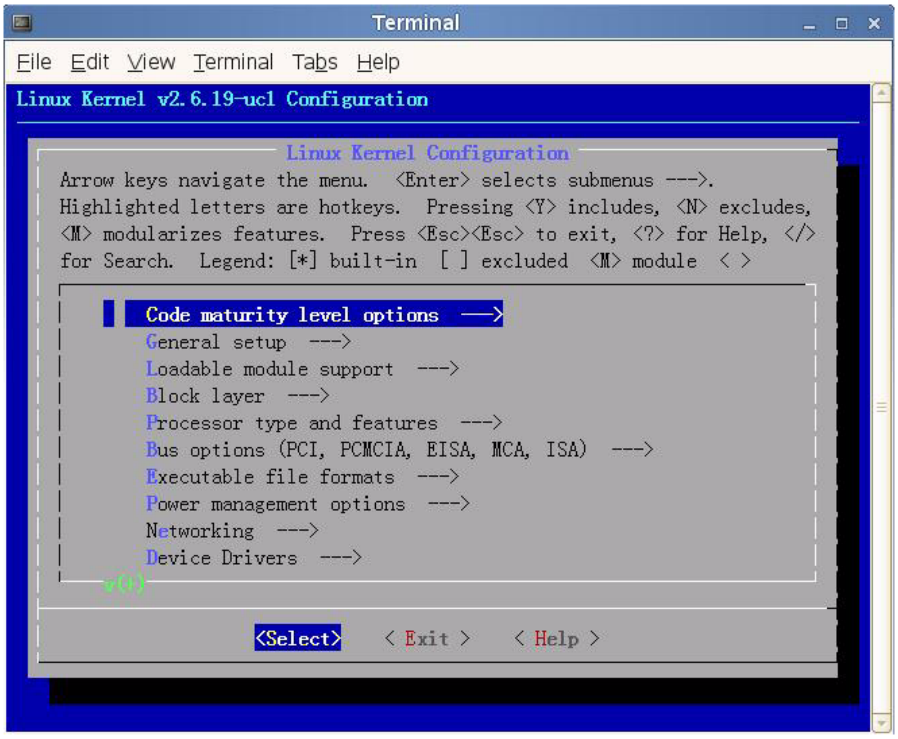
5. Then we went back to the **Main Menu** and choose the **Kernel/Library/Defaults Selection**, and made the following selections:

   - (Linux-2.6x) Kernel Version
   - (None) Libc Version
   - [ ] Default all settings (lose changes)
   - [*] Customize Kernel Settings
   - [*] Customize Vendor/User Settings
   - [ ] Update default Vendor Settings

6. We chose **Select** to exit. The kernel driver and service configuration program displayed next (see Figure 6).

   We needed to configure options so that we could use the USB equipment and the network modules.

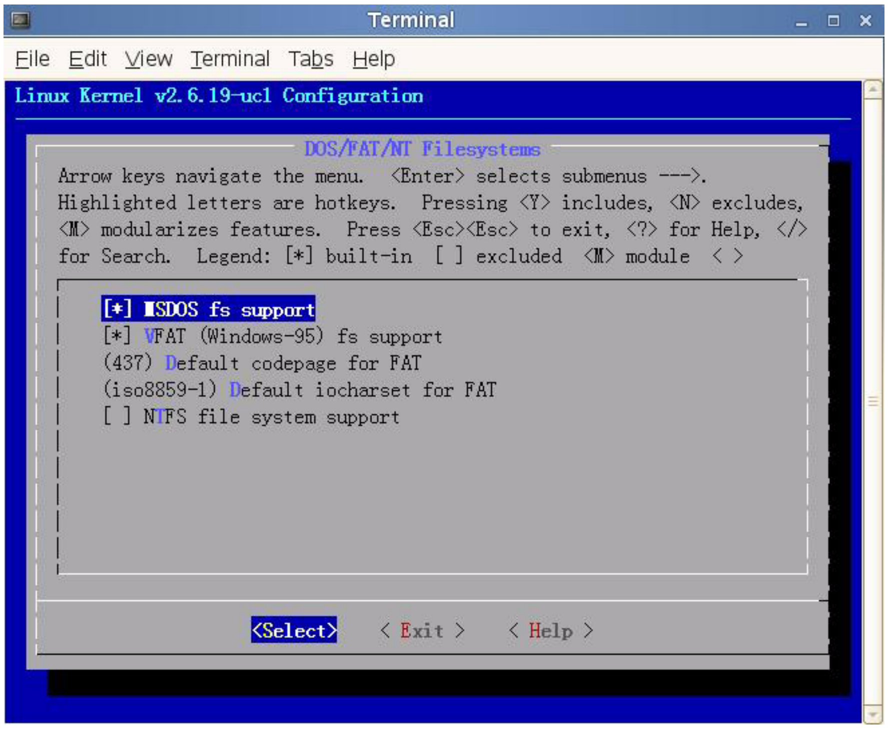
Figure 6. Linux Kernel Configuration



```
Linux Kernel v2.6.19-uc1 Configuration

                    Linux Kernel Configuration
  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  < >

          Code maturity level options  --->
          General setup  --->
          Loadable module support  --->
          Block layer  --->
          Processor type and features  --->
          Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->
          Executable file formats  --->
          Power management options  --->
          Networking  --->
          Device Drivers  --->
       v(+)

                <Select>    < Exit >    < Help >
```

7. We made the selections shown in Figure 7 in the **File systems > DOS/FAT/NT Filesystems** screen to configure the USB file system.

Figure 7. DOS/FAT/NT Filesystems Screen



```
Linux Kernel v2.6.19-uc1 Configuration

                    DOS/FAT/NT Filesystems
  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  < >

      [*] MSDOS fs support
      [*] VFAT (Windows-95) fs support
      (437) Default codepage for FAT
      (iso8859-1) Default iocharset for FAT
      [ ] NTFS file system support

               <Select>    < Exit >    < Help >
```

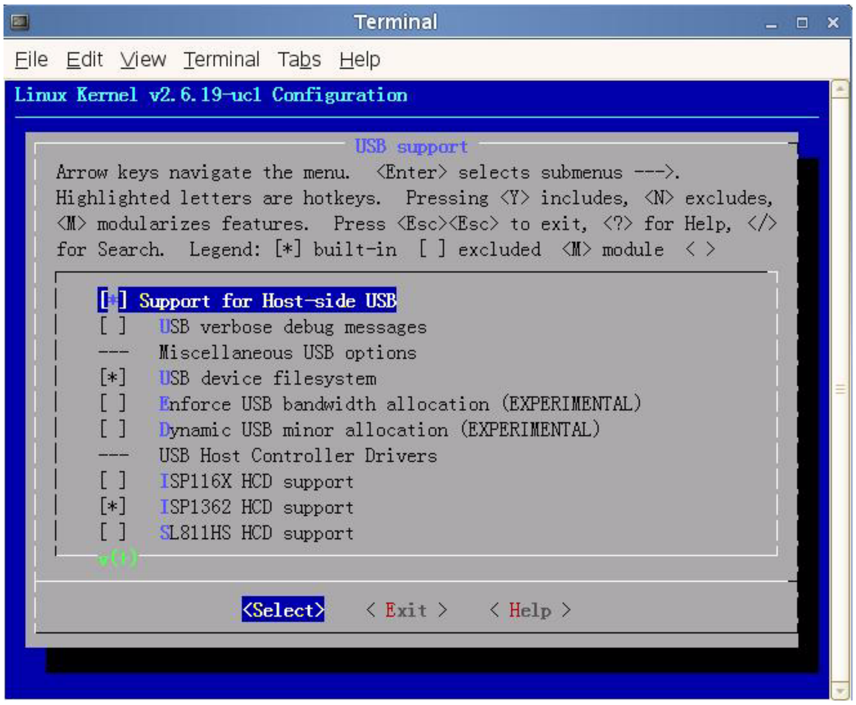
8. In the **Native Language Support** screen, we made the following selections:

   - (iso8859-1) Default NLS Option
   - [*]Codepage 437 (United States, Canada)
   - [*] NLS ISO 8859-1 (Latin 1; Western European Languages)

9. To configure the USB equipment driver, we made the following selections in the **Device Drivers > USB support** screen (see Figure 8).

   - [*] Support for Host-side USB
   - [*] USB device filesystem
   - [*] ISP1362 HCD support
   - [*] USB Mass Storage support
   - [*] USB Human Interface Device (full HID) support
   - [*] HID input layer support

Figure 8. USB Support Screen



```
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
    ---   Miscellaneous USB options
    [*]   USB device filesystem
    [ ]   Enforce USB bandwidth allocation (EXPERIMENTAL)
    [ ]   Dynamic USB minor allocation (EXPERIMENTAL)
    ---   USB Host Controller Drivers
    [ ]   ISP116X HCD support
    [*]   ISP1362 HCD support
    [ ]   SL811HS HCD support
    v(+)

               <Select>    < Exit >    < Help >
```

10. We made the following selections to configure support for the DM9000A device.

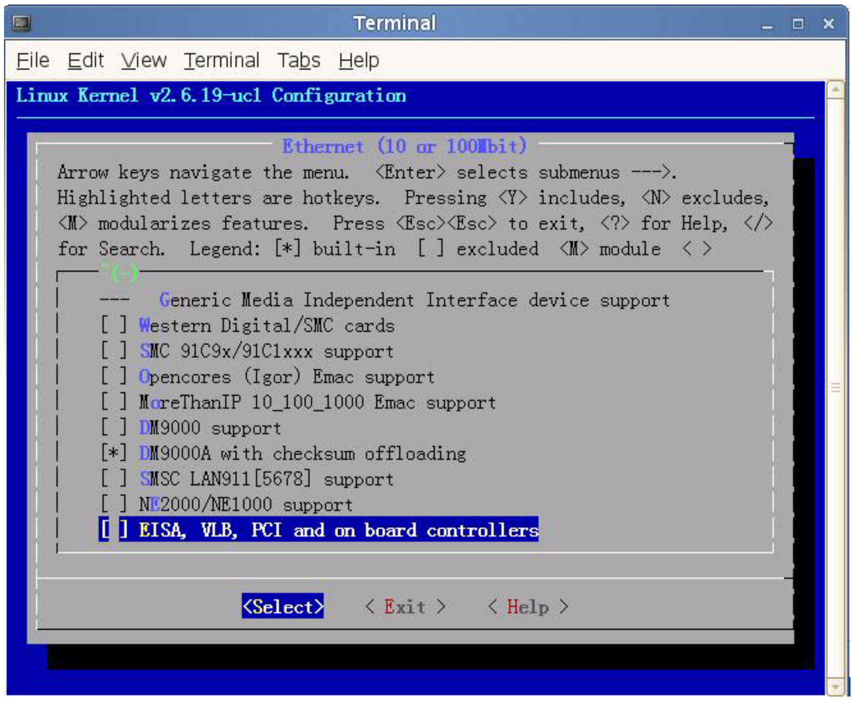
- In the **Enter Network device support** screen, we selected:

  [*] Network device support

- In the **Ethernet (10 or 100Mbit)** screen (see Figure 9), we selected:

  [*] DM9000A with checksum offloading

Figure 9. Ethernet (10 or 100 Mbit) Screen



```
Linux Kernel v2.6.19-uc1 Configuration

Ethernet (10 or 100Mbit)
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  < >

    ---   Generic Media Independent Interface device support
    [ ] Western Digital/SMC cards
    [ ] SMC 91C9x/91C1xxx support
    [ ] Opencores (Igor) Emac support
    [ ] MoreThanIP 10_100_1000 Emac support
    [ ] DM9000 support
    [*] DM9000A with checksum offloading
    [ ] SMSC LAN911[5678] support
    [ ] NE2000/NE1000 support
    [ ] EISA, VLB, PCI and on board controllers

          <Select>    < Exit >    < Help >
```

After we finished the configuration we exited the kernel compilation program. We could compile the kernel using the following commands at the command line:

1. `: /make romfs` ↵

2. `: /make` ↵

3. `: /make linux image` ↵

The system compiles the kernel and creates a zImage file in the image contents directory. The next step was to download the kernel to DE2 board using the Nios® II Integrated Development Environment (IDE) terminal and activate the µClinux operating system (OS).

Operating µClinux on the DE2 Board

To run µClinux on the DE2 board, we entered the following commands in the Nios II 6.1 Command Shell:

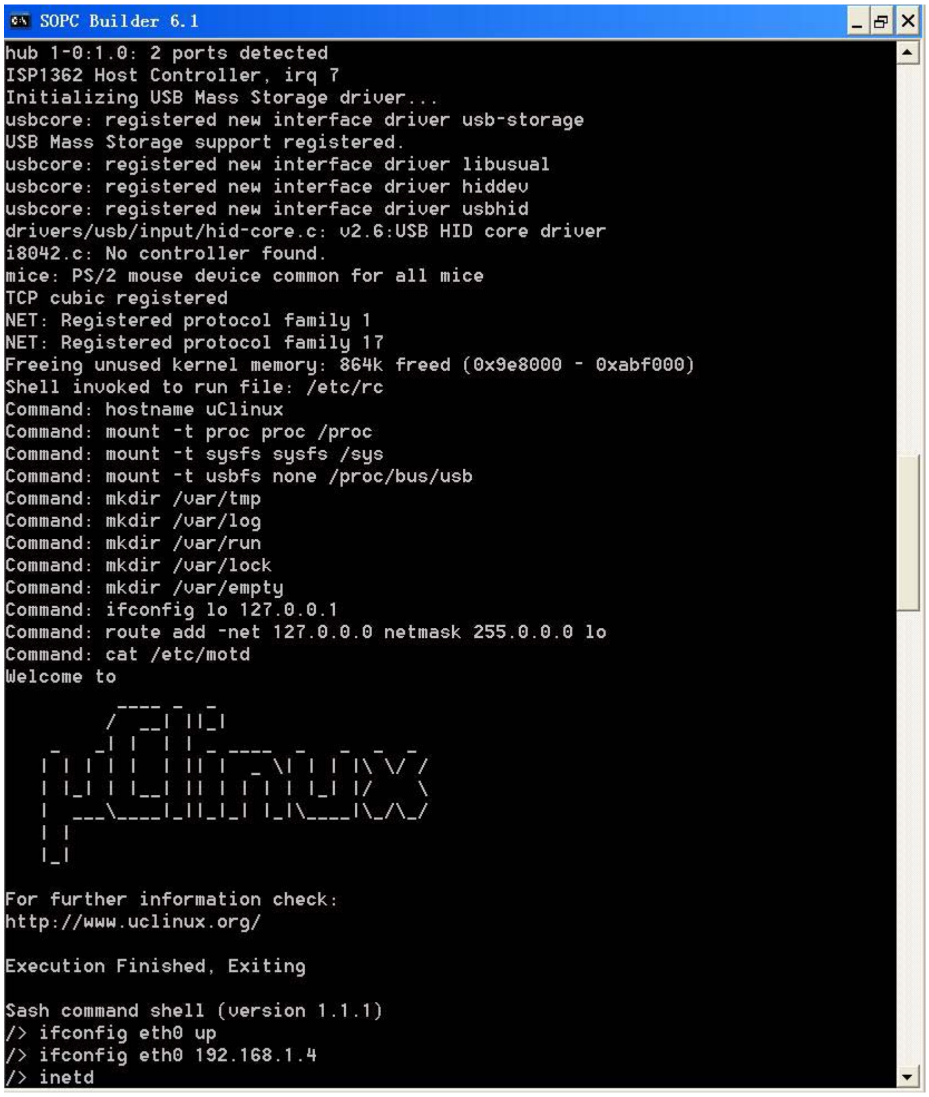
`: /nios2-configure-sof de2-net.sof` ↵ (this command configures the hardware circuit)

`: /nios2-download-g zImage` ↵ (this command downloads the system)

`: /nios2-terminal` ↵ (this command activates the µClinux system)

Figure 10 shows the interface after activation.

Figure 10. Activating μClinux



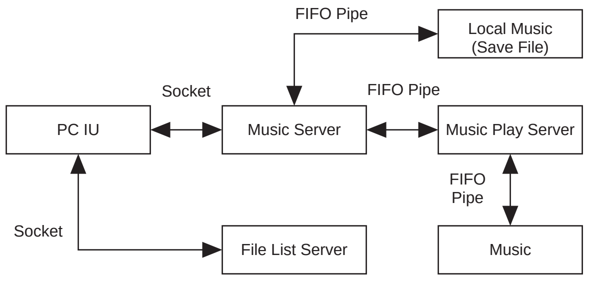
SOPC Builder 6.1

```
hub 1-0:1.0: 2 ports detected
ISP1362 Host Controller, irq 7
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
usbcore: registered new interface driver libusual
usbcore: registered new interface driver hiddev
usbcore: registered new interface driver usbhid
drivers/usb/input/hid-core.c: v2.6:USB HID core driver
i8042.c: No controller found.
mice: PS/2 mouse device common for all mice
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
Freeing unused kernel memory: 864k freed (0x9e8000 - 0xabf000)
Shell invoked to run file: /etc/rc
Command: hostname uClinux
Command: mount -t proc proc /proc
Command: mount -t sysfs sysfs /sys
Command: mount -t usbfs none /proc/bus/usb
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: mkdir /var/empty
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
Command: cat /etc/motd
Welcome to
          ____ _  _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
/> ifconfig eth0 up
/> ifconfig eth0 192.168.1.4
/> inetd
```

.wav File Network Uploading and Playing

Figure 11 shows the block diagram for uploading and playing **.wav** files on the network.

Figure 11. .wav File Uploading and Playing Block Diagram



The system functions are:

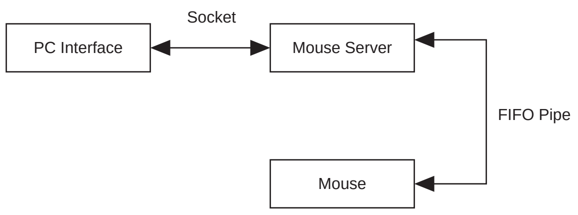
- *User interface*—The PC's user interface (UI) sends control information to the DE2 board, such as obtaining the music file list, playing specified music, suspending music, and stopping playback. The UI invokes network socket ports on the DE2 board for control:
  - 19851—Playing port
  - 19852—File list port
  - 19853—Remote mouse port
- *Music server process*—By transmitting control information, the music server process on the DE2 board automatically recognizes and controls the data. The music server process receives the control information from the PC via the network. It transmits the information to different energy-changing processes, and instructs them to finish various functions. When sending files, the music server process transfers the file name and file data to the music local process, which saves the files according to the designated file name. To control playback, the music server process sends control information to the music play server process along with the name of the file to play and when to pause or stop.
- *Music process*—When the music process finishes a task, it keeps reading data from the pipe and writes it into the audio FIFO buffer. If there is no data, it stops, which is equivalent to the function of a stop or pause.
- *Music play server process*—The music play server process obtains information required for playback, such as the file to play and control information. When a file must be played, the music play server process reads it and write its content into the playback pipe. After receiving the content, the music process begins playing immediately. Meanwhile, if the music play server process receives a pause or stop command, it stops writing audio information into the playback pipe and finishes the pause function.
- *List server process*—When a list request occurs, the list server process sends all file names in the music folder to the PC. The PC displays the received file names on the interface for the user to choose.
- *Music local process*—The music local process receives and saves files to the mobile hard disk. Files sent can be seen with the PC's list function.

Remote Mouse

Figure 12 shows the remote mouse modules, including:

- *PC interface*—Because of performance considerations, the PC interface inquires about mouse movements every 10 ms, obtains 32-byte movement information, and shows it on the interface.
- *Mouse server process*—The process continuously reads the piped data, saves it in 32-byte buffers, and waits for the PC acquisition commands. When there is a PC inquiry, null information is sent out to ensure that the program performs even if there is no data in the buffer.

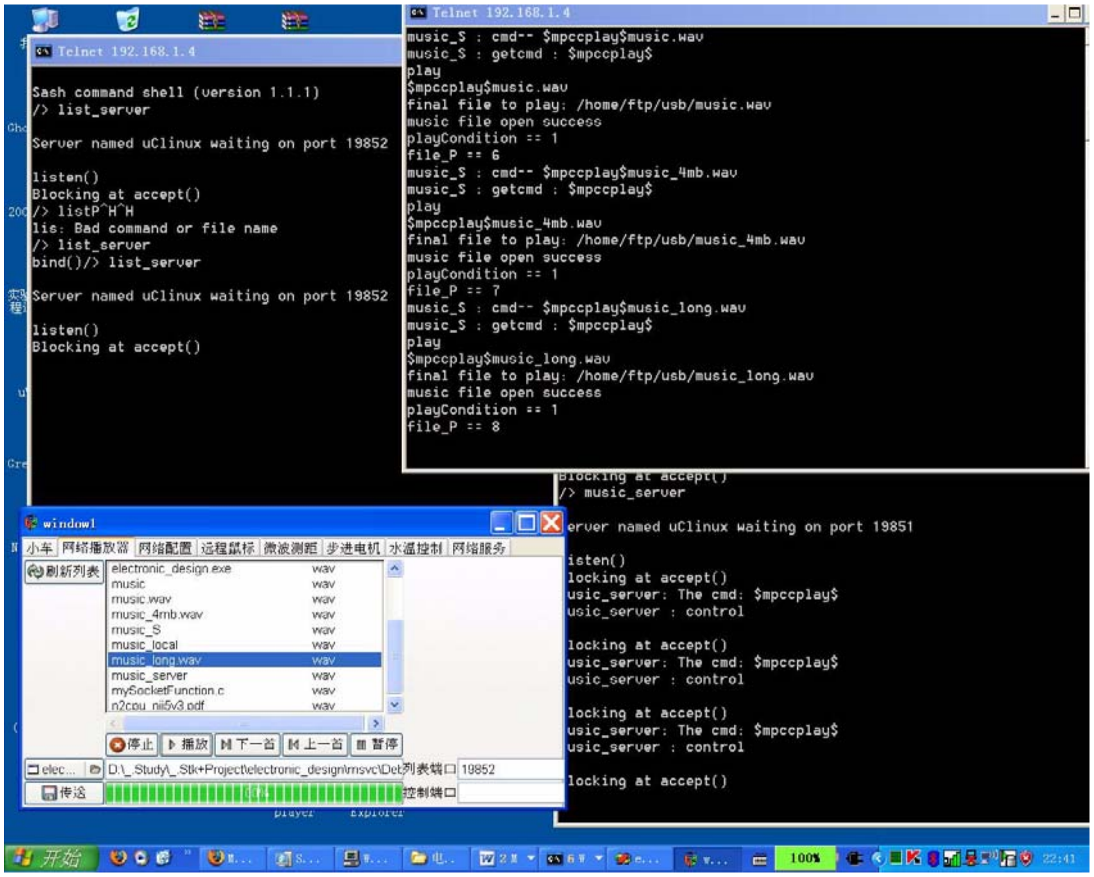
Figure 12. Remote Mouse Modules



Implementing On-Line Debugging

The obvious difference between embedded design and ordinary PC program debugging is the debugging difficulty because every debugging change requires the user to download the entire system file. With μClinux, however, the network interface simplifies the debugging process. To perform debugging using telnet, we entered μClinux, configured the network, activated the **inetd** server, logged onto the DE2 board using telnet, uploaded the program to debug using ftp, and debugged the program online. See Figure 13.

Figure 13. Telnet Debugging Interface



```
Telnet 192.168.1.4

Sash command shell (version 1.1.1)
/> list_server

Server named uClinux waiting on port 19852

listen()
Blocking at accept()
/> listP^H^H
lis: Bad command or file name
/> list_server
bind()/> list_server

Server named uClinux waiting on port 19852

listen()
Blocking at accept()
```

```
Telnet 192.168.1.4
music_S : cmd-- $mpccplay$music.wav
music_S : getcmd : $mpccplay$
play
$mpccplay$music.wav
final file to play: /home/ftp/usb/music.wav
music file open success
playCondition == 1
file_P == 6
music_S : cmd-- $mpccplay$music_4mb.wav
music_S : getcmd : $mpccplay$
play
$mpccplay$music_4mb.wav
final file to play: /home/ftp/usb/music_4mb.wav
music file open success
playCondition == 1
file_P == 7
music_S : cmd-- $mpccplay$music_long.wav
music_S : getcmd : $mpccplay$
play
$mpccplay$music_long.wav
final file to play: /home/ftp/usb/music_long.wav
music file open success
playCondition == 1
file_P == 8
```

```
Blocking at accept()
/> music_server

erver named uClinux waiting on port 19851

isten()
locking at accept()
usic_server: The cmd: $mpccplay$
usic_server : control

locking at accept()
usic_server: The cmd: $mpccplay$
usic_server : control

locking at accept()
usic_server: The cmd: $mpccplay$
usic_server : control

locking at accept()
```

User Interface Compilation

We chose the GTK+ version 2.0 library to compile the UI for the following reasons:

- GTK+, a cross-platform library, can be run on the Windows, Linux, and Mac OS X operating systems. With this program, we can release software on current mainstream platforms without transforming it.
- It possesses a Glade item and can configure the UI quickly, saving a lot of mechanical work.

Our system has two program UIs:

- *Network player*—As shown in Figure 1 on page 22, the network player interface integrates two functions: a file list that displays the music files on the mobile hard disk and a network player that selects, plays, pauses, and stops the file listed previously.
- *Remote mouse*—When the user plugs a mouse into the DE2 board’s USB port and activates the network service, the mouse pointer on the monitor moves when the user moves the mouse (see Figure 2 on page 23). The user can pause or activate this function.

Development Environment

Our system has the following development environment:

- *Development board*—Altera DE2 board
- *FPGA synthesis tools*—Quartus®II version 6.1 development kit
- *Cross-compiling development environment*—Suse Linux 10.2 x86
- *Kernel compilation environment*—Suse Linux 10.2 x86
- *User interface*—GTK+ version 2.0

Design Features

Our design uses the network as information media. With highly integrated technology, the design saves system costs and shortens the system configuration cycle.

The design positions the PC as a network terminal, integrating the network into every day life. Just imagine, you can select goods you need at home using a touch-screen LCD on your refrigerator. Suppliers can sell their goods online, shortening shopping time and minimizing retailers' costs for real stores.

In schools, enterprises that manage the electronic equipment has become an important aspect of daily maintenance. With a smart network platform, administrators can sit in their offices and click a mouse to control buildings, electric lights, and air conditioners throughout the campus. Costing only a little, the platform can save huge costs in long-term maintenance.

Technologically, the DE2-based system integrates:

- Embedded μClinux OS
- SOPC hardware/software integration
- Wave music file playback on the DE2 board
- Loading and using USB equipment on the DE2 board
- USB mouse
- Mobile hard disk
- On-line μClinux debugging
- Embedded Ethernet
- Audio D/A converter

We developed a smart control design concept and have essentially finished it. The system integrates network technology and an SOPC embedded system, and implements a comprehensive technology application. With the development of additional technologies, integration will provide better applications.

Conclusion

We spent over three months in the Zhang Zhigang Innovation Lab where we learned advanced SOPC technology, learned how to use Altera's FPGAs and software, and gained a lot of experience. Theory is abstract. Undergraduates who get used to accepting theories lack training, hands-on experience, and innovation. This contest gave us the opportunity to put what we have learned into practice. From program design to implementation, it showed us the project stages and the gap between theory and practice.

Usually we are very confident about what we have learned in school, but we may be unable to deal with a real project. Because it is systematic engineering, this project covers almost all of our undergraduate curriculum. With this benefit, we gained a lot from participating in the contest. We had to solve many hard problems in the contest, and using three different Altera software design programs was the first one. During the project, a trivial problem could cost us several hours work, so accumulating knowledge we could use in daily life was very important. A solid foundation will help us solve problems smoothly and efficiently.

We transplanted μClinux to the DE2 board, which made it convenient and efficient to use the DE2 board resources. The process plays a particularly important role in multi-tasking development. Modular concepts are critical for the design of complex tasks. An embedded OS offers an ideal operating environment for modular programs. In our design, we can perform debugging on each module, which increases debugging flexibility.

By exposure to Altera's FPGA hardware/software system, we learned about embedded equipment trends. Altera's Nios II IDE supports the developers in hardware and software, saving a lot of time. Additionally, the products have considerable flexibility: the Nios II CPU can be clipped according to users' demands, the system bus interface supports Verilog HDL and VHDL for hardware interfaces, and users only need to load configured equipment onto the SOPC Builder bus.

By participating in the contest, we learned that system engineering requires comprehensive knowledge, and its performance is determined by the weakest segment. A faulty segment aborts the entire system.

Finally, we extend our gratitude to Instructor Zhang and the assistant instructors in the innovation lab for their support in equipment and technology. Without them, we would not have completed the program.

References

[1] Zhigang, Zhang, *DE2 Platform Application and DSP Builder Technology,* Training Material for 2007 Altera Cup Shanghai Jiao Tong University Electronic Design Contest, 2007.

[2] Zhigang, Zhang, *SOPC Technology,* Reference for 2007 Altera Cup Shanghai Jiao Tong University Electronic Design Contest, 2007.

[3] Thomas, Nathan, *Developing on Linux An Introduction to Development on Linux*, Red Hat, Inc., nthomas@redhat.com.

[4] Corbet, Jonathan, Rubini, Alessandro, and Kroah-Hartman, Greg, *Linux Device Drivers, Third Edition*, O'Reilly, Beijing: February 2005.

[5] Strahnen, Dr. Ing. M., *HOWTO: NIOS-CPU with additional hardware driven with μClinux*.

[6] *Linux Application Program Development Guide: to Use Gtk+Gnome Library*

[7] *GTK+ Reference Manual*, GNOME Documentation Library.

[8] **http://nioswiki.jot.com**