XILINX®

XAPP542 (v1.0) Sept. 27, 2004

Getting Started with U-Boot on the ML300

Author: Sean Chang and Peter Ryser

Summary

This application note covers the steps necessary to run the open source firmware, Universal Bootloader (U-Boot), and to use it to boot Linux on the embedded IBM PowerPC™ 405 (PPC405) processor available on Virtex-II Pro™ ML300 Evaluation Platforms.

Introduction

U-Boot is a universal bootloader that is easily extensible to accommodate various embedded devices based on popular CPU architectures. Being an open source project, it has received contributions from developers all over the world. One of the strengths of U-Boot is its flexibility to let users boot their systems from various sources such as Ethernet Trivial File Transfer Protocol (TFTP), flash memory, or serial port. The ML300 port of U-Boot includes the Embedded Development Kit (EDK) Microprocessor Library Definition (MLD) format that can automatically generate libraries matching the hardware project. This application note offers concrete steps to get U-Boot up and running on the PPC405 processor embedded in the Virtex-II Pro FPGA that is featured on the ML300 board.

The exercises in this application note are prepared on a standard x86 workstation running on Red Hat Linux. The workstation has the ISE and EDK development tools that came with the ML300 development board package installed. Also installed is the MontaVista Linux 3.1 Preview Kit for ML300. This application note assumes the knowledge covered in application note XAPP765, *Getting Started with EDK and MontaVista Linux*. The first section shows the steps necessary to get U-Boot up and running on the ML300 development board. The next section discusses stand-alone applications and provides two ready to run demonstrations. The third section demonstrates some useful U-Boot commands for ML300. Finally, a section covers additional features relating to EDK and U-Boot that might be useful.

Table 1 lists the tools and files required to get started.

Table 1: **Required Tools**

Item	Details
Workstation	Standard x86 system running Red Hat Linux with GNU compiler.
ML300 board	See http://www.xilinx.com/ml300.
Files associated with the application note	http://www.xilinx.com/bvdocs/appnotes/xapp542.zip
CompactFlash adapter or writer	For PCs, use a CompactFlash writer. Most of these writers are USB-based. For laptops, use the CF card adapter that ships with ML300.
Parallel Cable IV download cable	Shipped with ML300 board.

Table 1: **Required Tools *(Continued)***

Item	Details
EDK and ISE development tools	Shipped with ML300 board.
MontaVista Linux 3.1 Preview Kit for ML300	Fill out the web form at http://www.mvista.com/previewkit to access the MontaVista Linux Professional Edition 3.1 In the Platform field, select Xilinx Virtex-II Pro ML300. An e-mail with access to a web page containing a README and two CD image files will be sent. Burn the **previewkit-mvl310_xilinx-ml300-encrypt.img** image to a CD. The source image is not needed.

Getting U-Boot Up and Running

The exercises in this section demonstrate a quick way to get U-Boot up and running on the ML300 development board.

Downloading and Building an ML300 Reference Design

This section shows how to build the EDK reference design for ML300.

1. Extract the EDK ML300 reference design from the ml300_edk3.zip file

   The EDK reference design file for ML300 is located in the xapp542.zip package. Save "ml300_edk3.zip" to $HOME/xilinx/ and unzip the file.

```
$ cd $HOME/xilinx
$ unzip ml300_edk3.zip
```

2. Build the hardware bitstream by doing the following:

```
$ cd $HOME/xilinx/projects/ml300_edk3/
$ xps -nw system_linux.xmp
XPS% run init_bram
```

3. The generated hardware bitstream is located here: $HOME/xilinx/projects/ml300_edk3/implementation/download.bit. This will be used in "Generating an ACE File," page 3 to generate the ACE file.

   A pre-generated bitstream, download.bit, is available within the xapp542.zip package.

Downloading and Compiling the Source Code for U-Boot

This section provides instructions for downloading and compiling the source code for U-Boot.

1. The compressed source code file for U-Boot can be found within the xapp542.zip package. Save "u-boot.tar.bz2" to $HOME/xilinx and uncompress the file.

```
$ cd $HOME/xilinx
$ bunzip2 u-boot.tar.bz2
$ tar xvf u-boot.tar
```

2. Compile the source code for U-Boot using the "ppc_405-" tools from MontaVista. Please refer to XAPP765, *Getting Started with EDK and MontaVista Linux*, for instructions on how to install the MontaVista Linux Preview Kit for ML300 and include the MontaVista tools in the search path.

```
$ cd $HOME/xilinx/u-boot-1.1.1
$ export CROSS_COMPILE=ppc_405-
$ make distclean
$ make ml300_config
$ make
```

3. The executable ELF file is located here:

```
$HOME/xilinx/u-boot-1.1.1/u-boot
```

A pre-compiled U-Boot ELF executable file, uboot.elf, is available within the xapp542.zip package.

Generating an ACE File

This section shows how to generate the ACE file from the hardware bitstream and the ELF file for U-Boot generated from the previous two exercises.

1. Generate the ACE file using the bitstream and the ELF file for U-Boot.

   ```
   $ cd $HOME/xilinx/projects/ml300_edk3/
   $ xmd -tcl genace.tcl -jprog -board ml300 -hw implementation/download.bit \
   -elf ../../u-boot-1.1.1/u-boot -ace uboot.ace
   ```

   A pre-generated ACE file, uboot.ace, is available within the xapp542.zip package.

2. Save the uboot.ace file to the Microdrive that ships with the ML300 development board under the XILINX/myace directory.

Building a U-Boot Compatible Linux Kernel

1. Copy the Linux BSP from the EDK project to the Linux kernel.

   Assuming the source for the kernel from the MontaVista Linux 3.1 Preview Kit is copied to the $HOME/linux-2.4.20_mvl31/ directory, the following shows how to copy the Linux BSP from the EDK project to the Linux kernel:

   ```
   $ cd $HOME/linux-2.4.20_mvl31/
   $ gtar cvf - -C \
   $HOME/xilinx/projects/ml300_edk3/ppc405_0/libsrc/linux_v2_00_b/linux .| \
   gtar xf -
   ```

2. Apply the ml300.uboot.patch.

   Before configuring the Linux kernel, apply the ml300.uboot.patch file, available within the xapp542.zip package, to the source of the kernel. This patch makes the board structure compatible with U-Boot.

   ```
   $ cd $HOME/linux-2.4.20_mvl31/
   $ patch -p1 -i $HOME/ml300.uboot.patch
   ```

3. Configure the Linux kernel.

   Follow the steps in the “Building the MontaVista Linux Kernel for the ML300” section in XAPP765.

   Note that normal build targets like “zImage” or “bzImage” are not used by U-Boot. The MontaVista Linux 3.1 Preview Kit for ML300 supports a new build target called “uImage” that can build image files usable by U-Boot.

   A pre-compiled Linux image file, uboot.uImage, built with “make dep uImage” is available for download within the xapp542.zip package.

4. Create a “linux” directory on the Microdrive.

   Create a new directory called “linux” on the FAT partition in the Microdrive and store the uboot.uImage file there. U-Boot can load this image file to the on-board system memory space during boot up to start Linux.

Starting U-Boot on the ML300

This section illustrates the steps necessary to enable U-Boot to start up Linux automatically upon board power up.

1. Connect the serial cable from ML300 to the host workstation.

   Open a terminal window to connect to the COM port with the following settings: 9600 baud, 8 data bits, no parity, 1 stop bit, hardware or no flow control. This is the standard I/O for U-Boot.

2. Set the System ACE CF dial to 4 and power on the board.

   System ACE™ CF automatically loads the bitstream and starts U-Boot from the ACE file. When U-Boot starts, it waits for 3 seconds before going through the auto-boot process. If any key is pressed within the 3-second period, then U-Boot does not go through the auto-boot process and waits for users to enter commands. Since this is the first time U-Boot is booted on the ML300 board, please press **<Enter>** before the 3-second period to stop U-Boot from going into the auto-boot process.

3. The size of the persistent environment information for U-Boot is 1 KB by default. It is stored in the inter-integrated circuit (IIC) bus EEPROM available on the ML300 board starting at address location 0xC00. Because no environment is stored in this section of the IIC EEPROM initially, the output on the terminal window from U-Boot shows a warning message relating to environment bad CRC, which is normal at this point. Another bad CRC warning relating to board specific parameters might also be shown if the board specific parameters section (0x400 to 0x7FF) in the IIC EEPROM is corrupted. Under normal cases, only the environment CRC warning shows up the first time U-Boot starts. The output is something similar to the following:

```
U-Boot 1.1.1 (Jul 21 2004 - 12:47:21)


### No HW ID - assuming ML300
DRAM:  128 MB
*** Warning - bad CRC, using default environment


In:    serial
Out:   serial
Err:   serial
*** Warning - board specific parameters CRC error
Hit any key to stop autoboot:  3
```

4. To resolve the environment CRC warning, set the desired environment variables and save the environment information back to the IIC EEPROM using the **`saveenv`** command. First, display the default environment information used by U-Boot using the **`printenv`** command. Then, set the appropriate environment variables to have U-Boot launch Linux by default and then save the environment back to the IIC EEPROM. When asked by U-Boot whether to overwrite the board specific area, press “n” if no board CRC warning was shown in step 3 above, otherwise press “y” to fix the corrupted board specific parameters section in the IIC EEPROM. Shown below is the command exchange from the terminal with user commands shown in bold:

```
=> printenv
bootargs=console=ttyS0,9600 ip=off root=/dev/xsysace/disc0/part3 rw
bootcmd=bootp
bootdelay=3
baudrate=9600
loads_echo=1
```

```
stdin=serial
stdout=serial
stderr=serial
ethaddr=00.0a.35.00.22.01
=> setenv bootargs console=ttyS0,9600 ip=off root=/dev/xsysace/disc0/part3 rw
=> setenv bootfile linux/uboot.uImage
=> setenv bootcmd fatload ace 0 400000 \${bootfile}\;bootm 400000
=> saveenv
Saving Environment to EEPROM...

Only MAC address and CRC value will be written,
do you want to let U-Boot update your board specific
area with only these two parameters? <y/n> n

=> printenv
bootdelay=3
baudrate=9600
loads_echo=1
stdin=serial
stdout=serial
stderr=serial
ethaddr=00.0a.35.00.22.01
bootargs=console=ttyS0,9600 ip=off root=/dev/xsysace/disc0/part3 rw
bootfile=linux/uboot.uImage
bootcmd=fatload ace 0 400000 ${bootfile};bootm 400000
=>
```

5. Now that the environment information is set, press the System ACE CF reset again. This time, the CRC warning message does not appear. Do not press any key and let U-Boot continue with the auto-boot process. When the Linux login prompt shows up, it means U-Boot has successfully booted Linux. Use “linux” for login name and “ml300” for password. Below is the output from the terminal window (the bold lines are explained next):

```
U-Boot 1.1.1 (Aug 10 2004 - 10:59:48)
### No HW ID - assuming ML300
DRAM:  128 MB
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
reading linux/uboot.uImage
590893 bytes read
## Booting image at 00400000 ...
   Image Name:   Linux-2.4.20_mvl31-ml300
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    641348 Bytes = 626.3 kB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
Linux version 2.4.20_mvl31-ml300...
Xilinx Virtex-II Pro port ...
On node 0 totalpages: 32768
zone(0): 32768 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: console=ttyS0,9600 ip=off root=/dev/xsysace ...
...
...
...
Starting internet superserver: inetd.
Hostname: ml300.
Starting xdm

MontaVista Linux 2.1, Professional Edition

ml300 login: linux
password: ml300
```

Here is a brief explanation of the whole process when ML300 powers up.

a. With System ACE CF dial set to 4, System ACE CF uses the ACE configuration file located in the XILINX/myace directory on the Microdrive in the FAT partition to configure the board.

b. The bitstream in the ACE file is used to configure the FPGA and load the ELF executable file for U-Boot to memory and start U-Boot on the ML300 board.

c. When U-Boot starts, it looks for its environment information in the EEPROM and loads that into the on-board system memory and uses it to guide itself through the rest of the booting sequence.

d. Based on the environment settings used in step 4 above, U-Boot reads the Linux image file and stores it to address location 0x400000. From the above U-Boot output snippet, this is shown when U-Boot outputs the line:

   ```
   reading linux/uboot.uImage
   ```

e. U-Boot performs check sum verification on the Linux image file to make sure that it is not corrupted and uncompresses the image file if necessary. This is shown when U-Boot outputs the line:

   ```
   ## Booting image at 00400000
   ```

f. Finally, U-Boot transfers control to Linux. This is shown when U-Boot outputs the line:

   ```
   Linux version 2.4.20_mvl31-ml300
   ```

Running Stand-Alone Applications

One of the interesting features of U-Boot is its ability to execute stand-alone applications. They can be used as simple hardware diagnostic tools to verify any hardware on the board. These applications are located in the $HOME/xilinx/u-boot-1.1.1/examples directory. The two demos in this section illustrate the use of stand-alone applications to access information from on-board sensors and the Real-Time-Clock (RTC). The executable files for both demonstrations are included in the xapp542.zip package. The executable file for the system monitor application is called “system_monitor.bin” and the executable file for the RTC application is called “rtc_clock.bin”. Store these files in the root directory of the FAT partition on the Microdrive so U-Boot can load them from there into system memory.

System Monitor Program

This program, system_monitor.bin, is designed to load at address 0x50000. Notice that there is an offset of 0x4 between the starting point of execution and the location of the executable. The **go** command starts program execution at the specified address location. Below is the command exchange from the terminal window:

```
=> fatload ace 0 50000 system_monitor.bin
reading system_monitor.bin

3656 bytes read
=> go 50004
## Starting application at 0x00050004 ...
Example expects ABI version 2
Actual U-Boot ABI version 2

CPU.U255 Power Monitor (1.8-5V) Reading...
        1.8 Vin Measurement:  1.771V
        2.5 Vin Measurement:  2.265V
        5   Vin Measurement:  4.765V
            VCC Measurement:  4.382V
           Temp Measurement:     29C

CPU.U256 Power Monitor (2.5-12V) Reading...
        2.5 Vin Measurement:  2.460V
        12  Vin Measurement: 12.062V
        3.3 Vin Measurement:  3.145V
            VCC Measurement:  4.765V
           Temp Measurement:     30C

PIO.U2 Power Monitor (1.8-5V) Reading...
        1.8 Vin Measurement:  1.781V
        2.5 Vin Measurement:  2.278V
        5   Vin Measurement:  4.765V
            VCC Measurement:  4.382V
           Temp Measurement:     29C

PIO.U4 Power Monitor (2.5-12V) Reading...
        2.5 Vin Measurement:  2.460V
```

```
            12  Vin Measurement: 12.000V
            3.3 Vin Measurement:  3.145V
                VCC Measurement:  4.765V
               Temp Measurement:     31C

    On-Chip Temperature Sensor Reading...
            Ambient Temp:           34C
            On-die Temp:            41C

    Ambient Temperature Sensor Reading...
            Ambient Temperature:  32.3750C

    ## Application terminated, rc = 0x0
    =>
```

When stand-alone applications exit, U-Boot prints out a termination message, `## Application terminated, rc = 0x0`, as shown above.

The program above shows information from system monitors and temperature sensors on the ML300 board.

Real-Time-Clock Program

This program, “rtc_clock.bin”, is designed to load at address 0x40000. Below is the command exchange from the terminal window:

```
=> fatload ace 0 40000 rtc_clock.bin
reading rtc_clock.bin

2432 bytes read
=> go 40004
## Starting application at 0x00040004 ...
Example expects ABI version 2
Actual U-Boot ABI version 2

***** Clock Reading: 08/26/2004 11:35:10 Thursday

Do you want to modify current clock setting [y/n]? n

## Application terminated, rc = 0x0
=>
```

This program shows the date information from the RTC that is on the ML300 board.

U-Boot Commands for ML300

This section demonstrates a few U-Boot commands for the ML300 board; in particular, the IIC commands that are useful in manipulating IIC devices, and the System ACE CF commands that are useful in retrieving CompactFlash or Microdrive data.

IIC Commands

1. Start off by checking to see which IIC bus devices are available. Use the **iprobe** command to probe for available hardware.

```
=> iprobe
Valid chip addresses: 14 15 16 17 18 4B 50 53 56 57 6F
```

Chip addresses are represented in hex and do not include the IIC read or write command bit. This means, for example, an EEPROM memory chip with chip address of 0x50 will show up on the IIC bus as 0xA0 for write and 0xA1 for read. Table 2 below shows a complete listing of the IIC device types and their associated chip addresses. Please refer to the ML300 Users Guide for details on each device.

Table 2: **IIC Device Types and Chip Addresses**

Device Type	Chip Address
PWR Monitor (1.8 - 5V)	0x14
PWR Monitor (2.5 - 12V)	0x15
PWR Monitor (1.8 - 5V)	0x16
PWR Monitor (2.5 - 12V)	0x17
On-Chip Temperature	0x18
Ambient Temperature	0x4B
EEPROM	0x50
Audio Trimpot	0x53
Brightness Trimpot	0x56
RTC - EEPROM	0x57
RTC - Clock	0x6F

2. To display what is in the IIC EEPROM, use the **imd** command to display the memory content.

```
=> imd 50 400.2 30
0400: 45 3d 30 30 30 61 33 35 30 30 32 32 30 31 00 43    E=000a35002201.C
0410: 3d 38 30 00 00 ff ff ff ff ff ff ff ff ff ff ff    =80.............
0420: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
=> imd 50 c00.2 60
0c00: 78 a3 ed 54 62 6f 6f 74 61 72 67 73 3d 63 6f 6e    x..Tbootargs=con
0c10: 73 6f 6c 65 3d 74 74 79 53 30 2c 39 36 30 30 20    sole=ttyS0,9600
0c20: 69 70 3d 6f 66 66 20 72 6f 6f 74 3d 2f 64 65 76    ip=off root=/dev
0c30: 2f 78 73 79 73 61 63 65 2f 64 69 73 63 30 2f 70    /xsysace/disc0/p
0c40: 61 72 74 33 20 72 77 00 62 6f 6f 74 64 65 6c 61    art3 rw.bootdela
0c50: 79 3d 33 00 62 61 75 64 72 61 74 65 3d 39 36 30    y=3.baudrate=960
=>
```

Note: Address location 0xC00 is the default location where U-Boot stores its environment values; 0x400 is the default location for board specific information.

3. Device 0x56 (chip address) is the brightness trimpot. To change the brightness setting for the TFT LCD on the ML300 board, use the **inm** command to write a single byte of data to location 0xf8.

```
=> inm.b 56 f8
000000f8: 00 ? 20
000000f8: 20 ? b0
000000f8: b0 ? x(type a non-hexadecimal char to stop)
```

   The brightness of the LCD changes when the value at 0xf8 is changed from 0x20 to 0xb0. A value of 0xff is the brightest and 0x00 is the darkest.

System ACE CF Commands

1. Information about the FAT partition on the Microdrive can be displayed using the **fatinfo** command.

```
=> fatinfo ace 0
Partition 1: Filesystem: FAT16 "NO NAME    "
```

2. To show directory listing, use the **fatls** command.

```
=> fatls ace 0
      224   xilinx.sys
      977   readme.txt
            xilinx/
            vxworks/
            pr_ace/
            bmp/
            uboot/
2 file(s), 5 dir(s)

=> fatls ace 0 uboot
            ./
            ../
  1396931   final.ace
   567833   no_fat
   590930   fat_ace8
   590893   fat_ace16
   592910   fat_ace16_nfs
  1396931   final_16.ace
   592955   fat_ace8_nfs

7 file(s), 2 dir(s)

=>
```

Command Reference for U-Boot

The following tables offer a quick reference of useful commands along with examples for usage. For additional commands and details, refer to section **5.9, U-boot Command Line Interface**, in the U-Boot guide at http://www.denx.de/twiki/bin/view/DULG/Manual.

Table 3: **Download and Execution Related Commands**

Command	Description
bootm	Boots application image from memory Usage: bootm [addr [arg ...]]
go	Starts application at address 'addr' Usage: go <addr> [arg ...]

Table 3: **Download and Execution Related Commands *(Continued)***

Command	Description
bootp	Boots image via network using BOOTP to gather network information first and then use TFTP protocol to download the "bootfile". Usage: bootp [loadAddress] [bootfilename]
tftpboot	Boots image via network using TFTP protocol Usage: tftpboot [loadAddress] [bootfilename]

Table 4: **IIC Related Commands**

Command	Description
iprobe	Probes the IIC bus to discover active devices. Usage: iprobe
imd	Displays memory content. The length of the address field can be specified with the .0, .1, or .2 modifier. If none is specified, then the default .1 is used. Usage: imd <i2c_chip> <addr>[.0|.1|.2] [count]
imw	Writes (fills) memory with a byte value. Usage: imw <i2c_chip> <addr>[.0|.1|.2] <data> [count]
imm	Modifies memory with auto-increment address. Usage: imm[.b|.w|.l] <i2c_chip> <addr>[.0|.1|.2]
inm	Modifies memory with constant address. Usage: inm[.b|.w|.l] <i2c_chip> <addr>[.0|.1|.2]
icrc32	Calculates checksum. Usage: icrc32 <i2c_chip> <addr>[.0|.1|.2] <count>

Table 5: **FAT Related Commands**

Command	Description
fatinfo	Displays FAT partition information. Interface defines the type of storage used, e.g. "scsi" for SCSI, "ide" for IDE, or "ace" for System ACE CF. Dev defines the zero-base device number used if there are multiple devices using the same interface. Part defines the partition number being accessed on a particular device through a particular interface. Usage: fatinfo <interface> <dev[:part]>
fatls	Displays content of a directory. Usage: fatls <interface> <dev[:part]> [directory]
fatload	Loads file from a FAT partition to a user specified address location. Usage: fatload <interface> <dev[:part]> <addr> <filename> [bytes]

Table 6: **Miscellaneous Environment Variable Related Commands**

Command	Description
printenv	Prints environment variable information. Usage: printenv [env name]
setenv	Sets an environment variable. Usage: setenv <name> <value>
saveenv	Saves current environment variable information back to EEPROM. Usage: saveenv

Additional Features

This section covers additional features relating to EDK and U-Boot that the users might find useful. In particular, it covers the following topics:

- "Regenerating Hardware Parameters"
- "Downloading U-Boot using XMD"
- "Configuring U-Boot to Boot from a BOOTP/TFTP Server"
- "Accessing the CVS Server for U-Boot"

Regenerating Hardware Parameters

This section shows how to regenerate hardware parameters for U-Boot when the reference design is changed.

If the EDK reference design for ML300 is changed, or if a new design is built from scratch either with the Base System Builder (BSB) or all by hand, please make sure that the following hardware components are included as part of the new design before proceeding:

- Ethernet
- IIC
- System ACE CF
- 16x50 serial port/UART (at least one)

Most likely, the new hardware information is different from the original reference design. To let U-Boot know of the new hardware parameters, there are two options:

1. Edit $HOME/xilinx/u-boot-1.1.1/board/xilinx/ml300/xparameters.h directly to reflect the changes in the new hardware design.
2. The preferred way is to utilize the MLD technology provided by EDK to make the changes automatically. To do this, follow the steps below:
   a. Copy the System project file and software configuration file:
   ```
   $ cd $HOME/xilinx/projects/ml300_edk3/
   $ cp system.xmp system_uboot.xmp
   $ cp system.mss system_uboot.mss
   ```
   b. Edit system_uboot.xmp and change the "MSS file" setting to system_uboot.mss. For example:
   ```
   Original (system.xmp):
   IntStyle: default
   MHS File: system.mhs
   MSS File: system.mss
   NPL File: projnav/system.npl
   ```

```
Architecture: virtex2p
New (system_uboot.xmp):
IntStyle: default
MHS File: system.mhs
MSS File: system_uboot.mss
NPL File: projnav/system.npl
Architecture: virtex2p
```

c. Modify “system_uboot.mss” by adding a LIBRARY section to the end as follows:

```
BEGIN LIBRARY
PARAMETER LIBRARY_NAME = uboot
PARAMETER LIBRARY_VER = 1.00.a
PARAMETER TARGET_DIR = ../../../../../u-boot-1.1.1/
PARAMETER CONNECTED_PERIPHS =
   (opb_uart16550_0,opb_iic_0,opb_ethernet_0,opb_sysace_0)
END
```

d. Copy the sw_services and uboot_v1_00_a directories from U-Boot to the EDK project directory.

```
$ cd $HOME/xilinx/
$ cp -R u-boot-1.1.1/board/xilinx/ml300/sw_services \
   $HOME/xilinx/projects/ml300_edk3/
$ cd projects/ml300_edk3
$ cp -R sw_services/uboot_v1_00_a/ bsp/
```

e. Now, generate the Xilinx ML300 BSP for U-Boot:

```
$ cd $HOME/xilinx/projects/ml300_edk3/
$ xps -nw system_uboot.xmp
XPS% run libsclean
XPS% run libs
```

f. The new configuration files are copied to the right places within the U-Boot source tree. The $HOME/xilinx/u-boot-1.1.1/board/xilinx/ml300/xparameters.h file reflects the new hardware design parameters.

Downloading U-Boot using XMD

This sections shows how to download U-Boot using XMD.

Instead of generating an ACE file and using System ACE CF to start off U-Boot, it can also be loaded and started through XMD. Use “run download” from within EDK to load the bitstream. Then, use the **`XMD`** command to download and start U-Boot as shown:

```
$ cd $HOME/xilinx/u-boot-1.1.1/
$ xmd
XMD% ppcconnect
XMD% dow u-boot
XMD% run
```

Configuring U-Boot to Boot from a BOOTP/TFTP Server

This section shows how to configure U-Boot to boot from a BOOTP/TFTP server.

For information on how to configure the BOOTP/TFTP server, please refer to the U-Boot guide at http://www.denx.de/twiki/bin/view/DULG/Manual. Once the server is set up, save the Linux image file to /tftpboot/uboot.uImage and set this as the “bootfile”.

If a BOOTP server is set up, then the only environment information needed is the MAC address (ethaddr). Use “bootp” instead of “tftpboot” when setting the 'bootcmd” environment in this case. If only a TFTP server is set up, then “ipaddr”, "serverip", and "bootfile" environments also need to be set in addition to “ethaddr”. Use the appropriate values according to the TFTP server configuration. Below is an example when only a TFTP server is set up:

```
=> setenv ethaddr 00.0a.35.00.22.01
=> setenv ipaddr 125.128.54.56
=> setenv serverip 125.128.64.51
=> setenv bootfile /tftpboot/uboot.uImage
=> setenv bootcmd tftpboot\;bootm 400000
=> saveenv
Saving Environment to EEPROM...

Only MAC address and CRC value will be written,
do you want to let U-Boot update your board specific
area with only these two parameters? <y/n> n

=> printenv
bootdelay=3
loads_echo=1
baudrate=9600
stdin=serial
stdout=serial
stderr=serial
bootargs=console=ttyS0,9600 ip=off root=/dev/xsysace/disc0/part3 rw
ethaddr=00.0a.35.00.22.01
ipaddr=125.128.54.56
serverip=125.128.64.51
bootfile=/tftpboot/uboot.uImage
bootcmd=tftpboot;bootm 400000
=>
```

Press the System ACE CF reset button. This time, U-Boot automatically downloads the Linux image file in /tftpboot/uboot.uImage from the designated TFTP server and starts Linux. Below is the output from U-Boot in the terminal window:

```
U-Boot 1.1.1 (Aug 10 2004 - 10:59:48)
### No HW ID - assuming ML300
DRAM:  128 MB
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
TFTP from server 125.128.64.51; our IP address is 125.128.54.56
Filename '/tftpboot/uboot.uImage'.
Load address: 0x400000
Loading: #################################################
         #################################################
            #########################################
done
Bytes transferred = 872566 (d5076 hex)

## Booting image at 00400000 ...
   Image Name:   Linux-2.4.20_mvl31-ml300
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    641348 Bytes = 626.3 kB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
Linux version 2.4.20_mvl31-ml300...
Xilinx Virtex-II Pro port ...
On node 0 totalpages: 32768
zone(0): 32768 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: console=ttyS0,9600 ip=off root=/dev/xsysace ...
...
...
...
Starting internet superserver: inetd.
Hostname: ml300.
Starting xdm

MontaVista Linux 2.1, Professional Edition

ml300 login:
```

Accessing the CVS Server for U-Boot

This section shows how to access the CVS server for U-Boot.

The following CVS commands are useful for users who wish to access the CVS repository for U-Boot.

1. The **`cvs REPOSITORY login`** command lets users log into the U-Boot project. When prompted by a password, do not enter anything. Below is an example and the command is in bold:

   ```
   $ cvs -d:pserver:anonymous@cvs.sf.net:/cvsroot/u-boot login
   Logging in to :pserver:anonymous@cvs.sf.net:2401/cvsroot/u-boot
   CVS password:
   $
   ```

2. The **`cvs REPOSITORY checkout [OPTIONS] PROJECT(S)`** command lets users check out the U-Boot project. This checks out the entire source in the U-Boot project.

   ```
   $ cvs -d:pserver:anonymous@cvs.sf.net:/cvsroot/u-boot checkout u-boot
   cvs checkout: Updating u-boot
   U u-boot/CHANGELOG
   U u-boot/COPYING
   U u-boot/CREDITS
   U u-boot/MAINTAINERS
   U u-boot/MAKEALL
   U u-boot/Makefile
   …
   …
   …
   ```

3. The **`cvs update [OPTIONS] [FILES]`** command lets users update their existing U-Boot source code files. The original source code needs to be checked out using the command above in order to use this update command. If certain files have been updated in the repository, this downloads the changes to the local copy of the source code files. There are different options available, so please refer to the CVS documentation accessible from http://sourceforge.net/projects/u-boot for more information.

4. The **`cvs diff [OPTIONS] [FILES]`** command lets users do a comparison between the local copy of the source code files and the ones in the repository. When users want to submit a patch, the output from the **`diff`** command is what is needed. For more information on different command options please refer to http://sourceforge.net/projects/u-boot.

References

1. Xilinx, Inc., XAPP765: Getting Started with EDK and MontaVista Linux on Virtex-II Pro, http://www.xilinx.com/bvdocs/appnotes/xapp765.pdf
2. Xilinx, Inc., UG038: ML300 User Guide, http://www.xilinx.com/products/boards/ml300/docs/ml300_ug.pdf
3. ML300 evaluation platform website, http://www.xilinx.com/ml300
4. EDK website, http://www.xilinx.com/edk
5. ISE design tools center website, http://www.xilinx.com/ise
6. EDK example designs page: http://www.xilinx.com/ise/embedded/edk_examples.htm
7. U-Boot project home page: http://u-boot.sourceforge.net
8. DENX Software Engineering, DENX U-Boot and Linux Guide (DULG) for TQM8xxL, http://www.denx.de/twiki/bin/view/DULG/Manual
9. MontaVista Linux 3.1 Preview Kit for ML300, http://www.mvista.com/previewkit

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/27/04	1.0	Initial Xilinx release.