Booting Embedded Systems

Outline

Overview of Boot Loader

Functional Blocks of Boot Loader

Introduction and Comparison of
uClinux Boot Loader, Das U-Boot and RedBoot

Reference

Overview of Boot Loader

- Definition of Boot Loader
  - The first section of code to be executed after the embedded system is powered on or reset on any platform

Comparison between PC and Embedded System

- In an embedded system the role of the boot loader is more complicated since these systems do not have a BIOS to perform the initial system configuration.
- Boot Loader in x86 PC consists of two parts
  - BIOS(Basic Input/Output System)
  - OS Loader(located in MBR of Hard Disk)
    - Ex. LILO and GRUB

Boot Loaders for embedded System

- U-Boot ("Universal Boot loader". Boot loader for PowerPC or ARM based embedded Linux systems.)
- RedBoot (RedHat eCos derived, portable, embedded system boot loader)
- rrload (Boot loader for ARM based embedded Linux systems)

Boot Loaders for embedded System(Cont.)

- FILO (x86 compatible boot loader which loads boot images from the local filesystem, without help from legacy BIOS services. Expected usage is to flash it into the BIOS ROM together with LinuxBIOS.)
- CRL/OHH (Flash boot loader for ARM based embedded Linux systems)

Boot Loaders for embedded System(Cont.)

- PPCBOOT (Boot loader for PowerPC based embedded Linux systems)
- Alios (Assembler based Linux loader which can do basic hardware initialization from ROM or RAM. The goal is to eliminate the need for a firmware BIOS on embedded systems.)

Overview of Boot Loader

- As we can see, there are various boot loaders for different CPU architecture and Target Board
- Therefore, it’s nearly possible to develop a boot loader suited for every platform.
- But we can still summarize concepts of the design of Boot Loaders

Concepts of the design of Boot Loader

- Boot Loader is varied from CPU to CPU, from board to board. The design of SoC (System on Chip) may effect the design of Boot loader,too.
- Installation Medium
  - All the system software and data are stored in some kind of nonvolatile memory like FLASH and ROM.

Concepts of the design of Boot Loader(Cont.)

- Operation Mode of Boot Loader
  - Boot Loading
    - Normal Operation Mode for Boot Loader
    - Designed for end user
  - Downloading
    - Used when “first” load software components into embedded system
    - Designed for developer

Concepts of the design of Boot Loader(Cont.)

- Booting Sequence
  - Single Stage or Multi-Stage
  - Multi-Stage Boot Loader usually provide more complicated functions and and better portability
  - Boot Loader reside in nonvolatile memory usually take two stage, listed below:

Concepts of the design of Boot Loader(Cont.)

- 1st Stage
  - Initialize hardware components
  - Prepare memory space for loading 2nd Stage program
  - Copy 2nd Stage program into memory space
  - Set up sp(stack pointer)
  - Jump to entry point of 2nd Stage program
- 2nd Stage
  - Initialize hardware components used in this stage
  - Check memory map
  - Copy kernel and root file system images into memory
  - Set parameters
  - Boot kernel

Functional Blocks of Boot Loader

- Main Function Module
- I/O Channel Driver Module
- Memory Device Driver Module

Main Function Module

- Main Tasks
  - Initialize CPU and set clock rate
  - Mask all interrupts
  - Set up stack pointer
  - Enable Power Management

Main Function Module

- Main Tasks(Cont.)
  - Load the images of Linux kernel and root file system to system's RAM and then execute kernel
  - Have the ability to write data to flash memory to support downloading features such as downloading Linux kernel and upgrading boot loader itself

Basic Hardware Initialization Steps

1. Mask all interrupts
2. Set CPU speed and clock rate
3. Initialize RAM
4. Initialize LED
5. Disable CPU internal Instruction/Data Cache

I/O Channel Driver Module

- Main Tasks
  - Provide interface to send command to target board or to inform users the status of target board
  - Initialize I/O port(ex. Serial Port or Ethernet) when it needs to export its console to the host(another platform)

Memory Device Driver Module

- Main Tasks
  - Initialize memory,including the memory device and the registers in memory controller
  - Configure page size,memory size and other memory management registers. If it’s necessary, Some boot loader will will enable MMU(Memory Management Unit)

Introduction of uClinux Boot Loader

- What is uClinux
  - uClinux is a derivative of Linux 2.0 kernel intended for microcontrollers without Memory Management Units (MMUs).

Introduction of uClinux Boot Loader

- linux vs. uClinux?
  - Most user applications that run on top of uClinux, however, will not require multitasking
  - most of the binaries and source code for the kernel have been rewritten to tighten-up and slim-down the code base
  - uClinux kernel is much, much smaller than the original Linux 2.0 kernel, while retaining the main advantages of the linux operating system: stability, superior network capability, and excellent file system support.

Introduction of uClinux Boot Loader

- uClinux source distribution also provides boot loaders
- Take ARM-based embedded System for example, the source code can be found in arch/arm（armnommu）/boot/bootp.
- Boot loader here is Single Stage and hence relatively simple.

Introduction of Das U-Boot

- What is Das U-Boot
  - Das U-Boot is a GPL'ed cross-platform boot loader shepherded by project leader Wolfgang Denk and backed by an active developer and user community.
  - U-Boot provides out-of-the-box support for hundreds of embedded boards and a wide variety of CPUs including PowerPC, ARM, XScale, MIPS, Coldfire, NIOS, Microblaze, and x86.

Introduction of Das U-Boot

- Before building and installing U-Boot you need a cross-development tool chain for your target architecture
- Building U-Boot for a supported platform is very straight forward, very similar to the familiar "untar, configure, make" method used by many software projects

Introduction of Das U-Boot

- You can fine tune the default configuration for your particular environment and board by editing the configuration file, "include/configs/board>.h". This file contains several C-preprocessor #define macros that you can modify for your needs

Introduction of Das U-Boot

- The user interface to U-Boot consists of a command line interrupter, much like a Linux shell prompt
- Much like a traditional Linux shell the U-Boot shell uses environment variables to tailor its operation
- The U-Boot commands to manipulate environment variables have the same names as the BASH shell

Introduction of Das U-Boot

- Network Support
  - U-Boot supports TFTP (Trivial FTP), a stripped down FTP that does not require user authentication, for downloading images into the board's RAM
- Flash Support
  - U-Boot offers several commands for programming, erasing and protecting the flash memory. Ex. flinfo to get all information about flash on board.

Introduction of RedBoot

- What is RedBoot
  - acronym for "Red Hat Embedded Debug and Bootstrap“
  - provides a complete bootstrap environment for a range of embedded operating systems, such as embedded Linux™ and eCos™, and includes facilities such as network downloading and debugging
  - provides a simple flash file system for boot images

Introduction of RedBoot

- Highlights of RedBoot’s capability
  - Boot scripting support
  - Simple command line interface for RedBoot configuration and management, accessible via serial (terminal) or Ethernet (telnet)
  - Integrated GDB stubs for connection to a host-based debugger via serial or ethernet. (Ethernet connectivity is limited to local network only)
  - Attribute Configuration - user control of aspects such as system time and date (if applicable), default Flash image to boot from, default failsafe image, static IP address, etc.

Introduction of RedBoot

- Highlight of RedBoot’s capability(Cont.)
  - Configurable and extensible, specifically adapted to the target environment
  - Network bootstrap support including setup and download, via BOOTP, DHCP and TFTP
  - X/YModem support for image download via serial
  - Power On Self Test

Introduction of RedBoot

provides a command line user interface (CLI)

- also contains a set of GDB "stubs", consisting of code which supports the GDB remote protocol
- takes up both flash and RAM resources depending on its startup mode and number of enabled features

Introduction of RedBoot

- Startup Mode
  - ROM mode
    - RedBoot both resides and executes from ROM memory (flash or EPROM)
  - ROMRAM mode
    - RedBoot resides in ROM memory (flash or EPROM), but is copied to RAM memory before it starts executing
  - RAM mode
    - RedBoot both resides and executes from RAM memory

Comparison

- uClinux Boot Loader is Single Stage, relatively simple, provides basic functions only
- The difference between U-Boot and RedBoot is that RedBoot is based on HAL(Hardware Abstraction Layer) , and therefore provide better portability.

Reference

1. http://www-900.ibm.com/developerWorks/cn/linux/l-btloader/index.shtml
2. http://www.linuxdevices.com/articles/AT5085702347.html
3. http://www.uclinux.org/index.html
4. EMBEDDED SOFTWARE DEVELOPMENT with eCos
   ANTHONY J. MASSA