The success's road

Boot loader in embedded systems

www.farsight.com.cn
内容安排

1、Bootloader在嵌入式linux设备中的作用
2、u-boot介绍
3、u-boot启动代码分析（S3C2410）
4、u-boot中如何增加对网络芯片的支持
5、如何在u-boot中增加自定义命令
6、如何移植u-boot到新的平台（step-by-step）
7、JTAG Flash编程序的工作原理
Functions of a boot loader

- **Bootloader**
  - It is just a SW component that boots a device and launches the main software, usually OS like Linux.

- **Monitor**
  - In addition to booting capabilities, it provides a command-line interface that can be used for debugging, reading/writing memory, flash reprogramming, configuring, etc.

- Usually we can refer to both types of software as "boot loaders"

- Note: uCLinux boot loader is a part of zImage which can decompress kernel image and start up.
Introduction of u-boot

- Wolfgang Denk, DENX Software Engineering, GPL license
- [http://sourceforge.net/projects/u-boot](http://sourceforge.net/projects/u-boot)
- U-Boot can be installed in a boot ROM and used to initialize and test the hardware or to download and run application code.
- U-Boot aims at becoming the standard bootloader
  - The richest, most flexible, and most actively developed open-source bootloader
  - Supports 100 different PPC-based boards, more than a dozen ARM-based boards, a handful of x86-based boards, and more…
  - Capable of booting a kernel through TFTP, from an IDE or SCSI disk, and from a DOC
### Platform dependence
- board, cpu, lib_ppc, lib_arm, include...

### Platform independence
- common, net, fs, drivers...

### Tools and Documentation
- tools, doc

| Board         | Board dependent files, RPXlite(mpc8xx), smdk2410/arm920t, sc520_cdp(x86)...
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU specific files, mpc8xx, pp4xx, arm720t, arm920t, xscale, i386</td>
</tr>
<tr>
<td>lib_ppc</td>
<td>Files generic to PowerPC architecture</td>
</tr>
<tr>
<td>lib_arm</td>
<td>Files generic to ARM architecture</td>
</tr>
<tr>
<td>lib_i386</td>
<td>Files generic to X86 architecture</td>
</tr>
<tr>
<td>include</td>
<td>Header Files and board configs</td>
</tr>
</tbody>
</table>
### u-boot code analysis (cont’)

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>common</td>
<td>Misc functions</td>
</tr>
<tr>
<td>lib_generic</td>
<td>Generic library functions</td>
</tr>
<tr>
<td>net</td>
<td>Networking code</td>
</tr>
<tr>
<td>fs</td>
<td>File System Code</td>
</tr>
<tr>
<td>post</td>
<td>Power On Self Test</td>
</tr>
<tr>
<td>drivers</td>
<td>Common used device drivers</td>
</tr>
<tr>
<td>disk</td>
<td>Hard disk interface code</td>
</tr>
<tr>
<td>rtc</td>
<td>Real Time Clock drivers</td>
</tr>
<tr>
<td>dtt</td>
<td>Digital Thermometer and Thermostat drivers</td>
</tr>
</tbody>
</table>
U-boot Startup Sequence

_start:
--cpu/arm920t/start.S
reset:
cpu_init_crit:
relocate:
stack_setup:
start_armboot():
  init_sequence[] = {cpu_init, board_init, …}
flash_init():
  --board/smdk2410/flash.c
env_relocate():
devices_init():
console_init_r():
main_loop():
  --common/main.c
Compiling and Installing

- All source code is organized by Makefiles. Assume using GNU make.
- Configure u-boot for your target
  - make <BOARD_NAME>_config
  - make smdk2410_config
- Building in cross development environment
  - make CROSS_COMPILE = arm-linux-
- Files generated:
  - System.map: The symbol map
  - u-boot: U-Boot in ELF binary format
  - u-boot.bin: U-Boot raw binary image
  - u-boot.srec: U-Boot image in Motorola's S-Record format
Add Ethernet support in u-boot

- All Ethernet are located in u-boot/drivers/
- Common used Ethernet IC supported, DM9000/SMC91C111/CS8900/RTL8019, etc. Need to change code like address, chip select, irq number to suite for your platform.
- Three major functions needed:
  - eth_init
  - eth_rx
  - eth_send
- Need Ethernet packet analysis tools to debug.
Add self-defined command support in u-boot

- All command handle files are located in u-boot/common/cmd_0xxx.c
- Following the rules to defined a command, command’s help information and handler function.

    U_BOOT_CMD(
        go, CFG_MAXARGS, 1, do_go,
        "go      - start application at address 'addr'
        addr [arg ...]\n          - start application at address 'addr'
             passing 'arg' as arguments"
    );

    int do_go (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
    {
    }
Porting U-boot

- U-boot can support PowerPC, ARM, XSCALE and x86 Processors. More and more boards are supported now.
- Porting is very easy if you have a board which is very similar with a supported board.
- Make sure if the processor and I/O devices of your board is supported by u-boot. You should be aware of your hardware platform before porting u-boot.
- Add board specific files, configure and build it.
Porting Preparation

- Check the latest u-boot package to make sure if a similar board has been supported.
- For our S3C2410 board, SMDK2410 is supported in u-boot. So we develop a new u-boot based on SMDK2410.
- Configure and build smdk2410. We should solve any compiling problems.
  - Modify cross-compiler in Makefile
    ```
    CROSS_COMPILE=arm-linux-
    # make smdk2410_config
    # make
    ```
Porting Steps

1. Add a new configuration option for your board to the top level "Makefile", using the existing entries as examples.
   
   ```
   fs2410_config : unconfig
   
   @./mkconfig $(@:_config=) arm arm920t fs2410 NULL s3c24x0
   ```

2. Create a new directory to hold your board specific code. Add any files you need.

   ```
   board/fs2410/
   ```

3. Create a new configuration file for your board

   ```
   include/configs/fs2410.h
   ```

   If you're porting U-Boot to a new CPU, then also create a new directory to hold your CPU specific code.
Porting Steps (cont’)

4. Configure for your new board name.
   # make fs2410_config

5. Building u-boot with “make”, Hope u-boot image can work

6. Modify configuration of your board or initializing or functions, such as flash erase and write function.

7. Debug u-boot source code till u-boot boot up your board from power on.

[Of course, this last step is much harder than it sounds.]
How Flash programmer works? (1)

Refer to IEEE1149.1 for details

FARSIGHT
How Flash programmer works?(2)

You can control the operation of JTAG-compliant devices via a PC that contains a JTAG controller or via a stand-alone programmer.
www.farsight.com.cn

Thanks！