Summary:摘要

This directory contains the source code for U-Boot, a boot loader for
Embedded boards based on PowerPC, ARM, MIPS and several other
processors, which can be installed in a boot ROM and used to
initialize and test the hardware or to download and run application
code.
此目录包含uboot源代码，uboot是基于POWERPC，ARM，MIPS以及其他处理器的嵌入式boards
的bootloader，它可以用来安装到bootrom，初始化和测试硬件，下载运行应用程序等。

The development of U-Boot is closely related to Linux: some parts of
the source code originate in the Linux source tree, we have some
header files in common, and special provision has been made to
support booting of Linux images.
uboot的开发与linux相似：一部分源代码来自linux源码树，两者有通用的头文件；对linux
映像提供特别的支持。

Some attention has been paid to make this software easily
configurable and extendable. For instance, all monitor commands are
implemented with the same call interface, so that it's very easy to
add new commands. Also, instead of permanently adding rarely used
code (for instance hardware test utilities) to the monitor, you can
load and run it dynamically.
我们做了一些工作以使这个软件容易配置和扩展，比如，所有的monitor命令都使用一样
的调用接口，所以可以很容易的添加新命令。当然，如果你不想固定添加某些很少使用的代码（
例如用来测试硬件的工具代码），你可以动态的加载运行它们。

Status:

In general, all boards for which a configuration option exists in the
Makefile have been tested to some extent and can be considered
"working". In fact, many of them are used in production systems.
通常，Makefile里面有一个配置选项的所有boards都经过一定程度的测试，可以认为它们是working的
。事实上，它们当中很多都已经用到产品系统上了。

In case of problems see the CHANGELOG and CREDITS files to find out
who contributed the specific port.
遇到问题时，请在CHANGELOG和CREDITS两个文件中查找一个具体的移植是由谁贡献的。

Where to get help:

获取帮助

In case you have questions about, problems with or contributions for
U-Boot you should send a message to the U-Boot mailing list at

<u-boot-users@lists.sourceforge.net>. There is also an archive of
previous traffic on the mailing list - please search the archive
before asking FAQ's. Please see
http://lists.sourceforge.net/lists/listinfo/u-boot-users/
如果你对U-Boot有疑问，或者想为U-Boot贡献，你应该向U-Boot邮件列表<u-boot-users@lists.
sourceforge.net>发送消息。在提问前，请搜索邮件列表的历史记录：http://lists.sourceforg
e.net/lists/listinfo/u-boot-users/

Where we come from:
===================
我们的历史

- 从8xxrom源代码开始
- 建立了 PPCBoot 项目（http://sourceforge.net/projects/ppcboot）
- 清理代码
- 让代码更易于添加自定义单板
- 可以添加其它类型的 [PowerPC] CPU
- 扩展了函数，特别是：
  * 为 Linux 引导程序提供了扩展的接口
  * 下载 S-Record
  * 从网络引导
  * 从 PCMCIA / CompactFlash / ATA disk / SCSI ... 等设备引导
- 建立了ARMBoot项目（http://sourceforge.net/projects/armboot）
- 添加了其它CPU家族（从ARM开始）
- 建立了 U-Boot 项目（http://sourceforge.net/projects/u-boot）
- 当前项目的主页：http://www.denx.de/wiki/U-Boot

Names and Spelling:
===================
名字与拼写
本项目的官方名称为"Das U-Boot"。在所有文档中都应使用"U-Boot"。比如：
 这是U-Boot项目的README文件。文件名等，也应基于u-boot字符串。比如：
include/asm-ppc/u-boot.h
#include <asm/u-boot.h>
变量名，宏定义等，也需要基于u_boot或者U_BOOT来定义。比如：

```
    U_BOOT_VERSION        u_boot_logo
    IH_OS_U_BOOT        u_boot_hush_start
```

The "official" name of this project is "Das U-Boot". The spelling
"U-Boot" shall be used in all written text (documentation, comments
in source files etc.). Example:

```
       This is the README file for the U-Boot project.
```

File names etc. shall be based on the string "u-boot". Examples:

```
       include/asm-ppc/u-boot.h

       #include <asm/u-boot.h>
```

Variable names, preprocessor constants etc. shall be either based on
the string "u_boot" or on "U_BOOT". Example:

```
       U_BOOT_VERSION             u_boot_logo
       IH_OS_U_BOOT        u_boot_hush_start
```

Versioning:
===========
版本
U-Boot uses a 3 level version number containing a version, a
sub-version, and a patchlevel: "U-Boot-2.34.5" means version "2",
sub-version "34", and patchlevel "4".
uboot的3级版本数字包含一个主版本，子版本，补丁级别：uboot－2.34.5表示主版本2，
子版本34，补丁级别4。

The patchlevel is used to indicate certain stages of development
between released versions, i. e. officially released versions of
U-Boot will always have a patchlevel of "0".
补丁级别用来指示发布版本之间的特定开发阶段，比如，官方发布版本通常补丁级别是0。

```
Directory Hierarchy:
====================
目录结构
- board          与单板相关的文件
- common         一些与架构无关的函数
- cpu            针对特定CPU的文件
  - 74xx_7xx     针对Freescale MPC74xx 和 7xx CPU 的文件
  - arm720t      针对ARM 720 CPU 的文件
  -arm920t       针对ARM 920 CPU 的文件
   -at91rm9200   针对Atmel AT91RM9200 CPU 的文件
   -imx          针对Freescale MC9328 i.MX CPU 的文件
   -s3c24x0      针对Samsung S3C24X0 CPU 的文件
  - arm925t      针对ARM 925 CPU 的文件
  - arm926ejs    针对ARM 926 CPU 的文件
  - arm1136      针对ARM 1136 CPU 的文件
  - at32ap       针对Atmel AVR32 AP CPU 的文件
  - i386         针对i386 CPU 的文件
  - ixp          针对Intel XScale IXP CPU 的文件
  - leon2        针对Gaisler LEON2 SPARC CPU 的文件
  - leon3        针对Gaisler LEON3 SPARC CPU 的文件
  - mcf52x2      针对Freescale ColdFire MCF52x2 CPU 的文件
  - mcf5227x     针对Freescale ColdFire MCF5227x CPU 的文件
  - mcf532x      针对Freescale ColdFire MCF5329 CPU 的文件
  - mcf5445x     针对Freescale ColdFire MCF5445x CPU 的文件
  - mcf547x_8x   针对Freescale ColdFire MCF547x_8x CPU 的文件
  - mips         针对MIPS CPU 的文件
  - mpc5xx       针对Freescale MPC5xx  CPU 的文件
  - mpc5xxx      针对Freescale MPC5xxx CPU 的文件
  - mpc8xx       针对Freescale MPC8xx  CPU 的文件
  - mpc8220      针对Freescale MPC8220 CPU 的文件
  - mpc824x      针对Freescale MPC824x CPU 的文件
  - mpc8260      针对Freescale MPC8260 CPU 的文件
  - mpc85xx      针对Freescale MPC85xx CPU 的文件
  - nios         针对Altera NIOS CPU 的文件
  - nios2        针对Altera Nios-II CPU 的文件
  - ppc4xx       针对AMCC PowerPC 4xx CPU 的文件
  - pxa          针对Intel XScale PXA CPU 的文件
  - s3c44b0      针对Samsung S3C44B0 CPU 的文件
  - sa1100       针对Intel StrongARM SA1100 CPU 的文件
- disk           处理磁盘驱动器分区的代码
- doc            文档
- drivers        常用的设备驱动
- dtt            数字温度计及调节器的驱动
- examples       示范代码
- include        头文件
- lib_arm        针对ARM架构的文件
- lib_avr32      针对AVR32架构的文件
- lib_generic    针对所有架构的文件
- lib_i386       针对i386架构的文件
- lib_m68k       针对m68k架构的文件
- lib_mips       针对MIPS架构的文件
- lib_nios       针对NIOS架构的文件
- lib_ppc        针对PowerPC 架构的文件
- lib_sparc      针对SPARC架构的文件
- libfdt         支持平坦设备树(flattened device trees)的库文件
- net            网络代码
- post           上电自检
- rtc            实时时钟驱动
- tools          编译S-Record或U-Boot映像等相关工具

Software Configuration:
=======================
软件配置

Configuration is usually done using C preprocessor defines; the
rationale behind that is to avoid dead code whenever possible.
配置一般使用c宏定义；原因是避免可能存在的不可达代码。

There are two classes of configuration variables:
两种类型的配置变量
```

```
* _OPTIONS_ 配置:
  这类配置以"CONFIG_"开始，用户可以自行选择。

* _SETTINGS_ 配置:
  这类配置以"CFG_"开始，与硬件相关。如果你不清楚它的含义，则不要随便修改。

后面我们会添加一个配置工具，可能类似于Linux Kernel配置工具。目前，我们还得手动完成配置，比如建一些软链接，编辑一些配置
文件。下面我们使用TQM8xxL单板作为范例。
* Configuration _OPTIONS_:
  These are selectable by the user and have names beginning with
  "CONFIG_".

* Configuration _SETTINGS_:
  These depend on the hardware etc. and should not be meddled with if
  you don't know what you're doing; they have names beginning with
  "CFG_".

Later we will add a configuration tool - probably similar to or even
identical to what's used for the Linux kernel. Right now, we have to
do the configuration by hand, which means creating some symbolic
links and editing some configuration files. We use the TQM8xxL boards
as an example here.


Selection of Processor Architecture and Board Type:
---------------------------------------------------
选择处理器架构与单板类型:
For all supported boards there are ready-to-use default
configurations available; just type "make <board_name>_config".
对所有已经支持的单板，有默认配置可以直接使用；只需要输入"make <board_name>_config".

比如: 对一个 TQM823L 单板类型:
Example: For a TQM823L module type:

	cd u-boot
	make TQM823L_config

For the Cogent platform, you need to specify the cpu type as well;
e.g. "make cogent_mpc8xx_config". And also configure the cogent
directory according to the instructions in cogent/README.
对于Cogent平台，你需要指定CPU类型；比如 "make cogent_mpc8xx_config"。还要根据
cogent/README的指导来配置cogent目录。

Configuration Options:
----------------------
配置选项:
Configuration depends on the combination of board and CPU type; all
such information is kept in a configuration file
"include/configs/<board_name>.h".
配置依赖于单板类型与CPU类型；所有这些信息都保存在一个配置文件 "include/configs/<board_name>.h" 中

Example: For a TQM823L module, all configuration settings are in
"include/configs/TQM823L.h".
比如: 对一个 TQM823L 单板，所有配置设置都在文件"include/configs/TQM823L.h"中。

Many of the options are named exactly as the corresponding Linux
kernel configuration options. The intention is to make it easier to
build a config tool - later.
有许多选项与对应的Linux内核配置选项名称相同，目的是之后容易做一个配置工具。

The following options need to be configured:
需要配置的选项如下:
- CPU Type:	Define exactly one of
 CPU类型:	只能定义一个，比如CONFIG_MPC85XX.
		PowerPC based CPUs:
		-------------------
		CONFIG_MPC823,	CONFIG_MPC850,	CONFIG_MPC855,	CONFIG_MPC860
	or	CONFIG_MPC5xx
	or	CONFIG_MPC8220
	or	CONFIG_MPC824X, CONFIG_MPC8260
	or	CONFIG_MPC85xx
```

```
       or     CONFIG_IOP480
       or     CONFIG_405GP
       or     CONFIG_405EP
       or     CONFIG_440
       or     CONFIG_MPC74xx
       or     CONFIG_750FX

              ARM based CPUs:
              ---------------
              CONFIG_SA1110
              CONFIG_ARM7
              CONFIG_PXA250
              CONFIG_CPU_MONAHANS

              MicroBlaze based CPUs:
              ----------------------
              CONFIG_MICROBLAZE

              Nios-2 based CPUs:
              ----------------------
              CONFIG_NIOS2

              AVR32 based CPUs:
              ----------------------
              CONFIG_AT32AP

- Board Type: Define exactly one of
- 单板类型:    只能定义一个，比如CONFIG_MPC8540ADS.
              PowerPC based boards:
              ---------------------

              CONFIG_ADCIOP       CONFIG_FPS860L            CONFIG_OXC
              CONFIG_ADS860       CONFIG_GEN860T            CONFIG_PCI405
              CONFIG_AMX860       CONFIG_GENIETV            CONFIG_PCIPPC2
              CONFIG_AP1000       CONFIG_GTH          CONFIG_PCIPPC6
              CONFIG_AR405        CONFIG_gw8260       CONFIG_pcu_e
              CONFIG_BAB7xx       CONFIG_hermes       CONFIG_PIP405
              CONFIG_BC3450       CONFIG_hymod        CONFIG_PM826
              CONFIG_c2mon        CONFIG_IAD210       CONFIG_ppmc8260
              CONFIG_CANBT        CONFIG_ICU862       CONFIG_QS823
              CONFIG_CCM          CONFIG_IP860        CONFIG_QS850
              CONFIG_CMI          CONFIG_IPHASE4539   CONFIG_QS860T
              CONFIG_cogent_mpc8260       CONFIG_IVML24       CONFIG_RBC823
              CONFIG_cogent_mpc8xx CONFIG_IVML24_128  CONFIG_RPXClassic
              CONFIG_CPCI405              CONFIG_IVML24_256   CONFIG_RPXlite
              CONFIG_CPCI4052             CONFIG_IVMS8        CONFIG_RPXsuper
              CONFIG_CPCIISER4    CONFIG_IVMS8_128    CONFIG_rsdproto
              CONFIG_CPU86        CONFIG_IVMS8_256    CONFIG_sacsng
              CONFIG_CRAYL1       CONFIG_JSE          CONFIG_Sandpoint8240
              CONFIG_CSB272       CONFIG_LANTEC       CONFIG_Sandpoint8245
              CONFIG_CU824        CONFIG_LITE5200B    CONFIG_sbc8260
              CONFIG_DASA_SIM             CONFIG_lwmon        CONFIG_sbc8560
              CONFIG_DB64360              CONFIG_MBX          CONFIG_SM850
              CONFIG_DB64460              CONFIG_MBX860T              CONFIG_SPD823TS
              CONFIG_DU405        CONFIG_MHPC         CONFIG_STXGP3
              CONFIG_DUET_ADS             CONFIG_MIP405       CONFIG_SXNI855T
              CONFIG_EBONY        CONFIG_MOUSSE       CONFIG_TQM823L
              CONFIG_ELPPC        CONFIG_MPC8260ADS   CONFIG_TQM8260
              CONFIG_ELPT860              CONFIG_MPC8540ADS   CONFIG_TQM850L
              CONFIG_ep8260       CONFIG_MPC8540EVAL  CONFIG_TQM855L
              CONFIG_ERIC         CONFIG_MPC8560ADS   CONFIG_TQM860L
              CONFIG_ESTEEM192E   CONFIG_MUSENKI            CONFIG_TTTech
              CONFIG_ETX094       CONFIG_MVS1         CONFIG_UTX8245
              CONFIG_EVB64260             CONFIG_NETPHONE             CONFIG_V37
              CONFIG_FADS823              CONFIG_NETTA        CONFIG_W7OLMC
              CONFIG_FADS850SAR   CONFIG_NETVIA       CONFIG_W7OLMG
              CONFIG_FADS860T             CONFIG_NX823        CONFIG_WALNUT
              CONFIG_FLAGADM              CONFIG_OCRTC        CONFIG_ZPC1900
              CONFIG_FPS850L              CONFIG_ORSG         CONFIG_ZUMA

              ARM based boards:
              -----------------
```

```
		CONFIG_ARMADILLO,	CONFIG_AT91RM9200DK,	CONFIG_CERF250,
		CONFIG_CSB637,		CONFIG_DELTA,		CONFIG_DNP1110,
		CONFIG_EP7312,		CONFIG_H2_OMAP1610,	CONFIG_HHP_CRADLE,
		CONFIG_IMPA7,	    CONFIG_INNOVATOROMAP1510,	CONFIG_INNOVATOROMAP1610,
		CONFIG_KB9202,		CONFIG_LART,		CONFIG_LPD7A400,
		CONFIG_LUBBOCK,		CONFIG_OSK_OMAP5912,	CONFIG_OMAP2420H4,
		CONFIG_PLEB2,		CONFIG_SHANNON,		CONFIG_P2_OMAP730,
		CONFIG_SMDK2400,	CONFIG_SMDK2410,	CONFIG_TRAB,
		CONFIG_VCMA9

		MicroBlaze based boards:
		------------------------

		CONFIG_SUZAKU

		Nios-2 based boards:
		------------------------

		CONFIG_PCI5441 CONFIG_PK1C20
		CONFIG_EP1C20 CONFIG_EP1S10 CONFIG_EP1S40

		AVR32 based boards:
		-------------------

		CONFIG_ATSTK1000

- CPU Daughterboard Type: (if CONFIG_ATSTK1000 is defined)
		Define exactly one of
- CPU子卡类型:（如果定义了CONFIG_ATSTK1000）
	只能定义一个，比如CONFIG_ATSTK1002		CONFIG_ATSTK1002


- CPU Module Type: (if CONFIG_COGENT is defined)
		Define exactly one of
		CONFIG_CMA286_60_OLD
 CPU Module 类型（如果定义了CONFIG_COGENT）
	只能定义下列中的一个：CONFIG_CMA286_60_OLD

--- FIXME --- 尚未经过测试的:
		CONFIG_CMA286_60, CONFIG_CMA286_21, CONFIG_CMA286_60P,
		CONFIG_CMA287_23, CONFIG_CMA287_50

- Motherboard Type: (if CONFIG_COGENT is defined)
– 母板类型:（如果定义了CONFIG_COGENT）
	可以选择下列定义：
		Define exactly one of
		CONFIG_CMA101, CONFIG_CMA102

- Motherboard I/O Modules: (if CONFIG_COGENT is defined)
– 母板I/O Modules: (如果定义了CONFIG_COGENT)
	可以择下面的一个或多个定义：
		Define one or more of
		CONFIG_CMA302

- Motherboard Options: (if CONFIG_CMA101 or CONFIG_CMA102 are defined)
– 母板选项:（如果定义了CONFIG_CMA101或者CONFIG_CMA102）
	可以择下面的一个或多个定义：
		Define one or more of
	在LCD上每秒钟用旋转字符(即|\-/|\-/)更新字符的位置
		CONFIG_LCD_HEARTBEAT	- update a character position on
					  the lcd display every second with
					  a "rotator" |\-/|\-/

- Board flavour: (if CONFIG_MPC8260ADS is defined)
		CONFIG_ADSTYPE
		Possible values are:
			CFG_8260ADS	- original MPC8260ADS
			CFG_8266ADS	- MPC8266ADS
			CFG_PQ2FADS	- PQ2FADS-ZU or PQ2FADS-VR
			CFG_8272ADS	- MPC8272ADS
```

```
- MPC824X Family Member (if CONFIG_MPC824X is defined)
MPC824X 家族成员（如果定义了CONFIG_MPC824X）
        只能定义下列中的一个：
                Define exactly one of
                CONFIG_MPC8240, CONFIG_MPC8245

- 8xx CPU Options: (if using an MPC8xx cpu)
8xx CPU 选项:（如果使用MPC8xx CPU）
        CONFIG_8xx_GCLK_FREQ    - 不推荐: 如果get_gclk_freq()不能工作(比如,
没有32KHz PIT/RTC参考时钟)，用该宏定义CPU时钟。
                CONFIG_8xx_GCLK_FREQ - deprecated: CPU clock if
                                       get_gclk_freq() cannot work
                                       e.g. if there is no 32KHz
                                       reference PIT/RTC clock
                CONFIG_8xx_OSCLK     - PLL input clock (either EXTCLK
                                       or XTAL/EXTAL)

- 859/866/885 CPU options: (if using a MPC859 or MPC866 or MPC885 CPU):
                CFG_8xx_CPUCLK_MIN
                CFG_8xx_CPUCLK_MAX
                CONFIG_8xx_CPUCLK_DEFAULT
                        See doc/README.MPC866

                CFG_MEASURE_CPUCLK
        定义该宏来测量实际的CPU时钟，否则需要保证配置的正确性。通常用于单板确认锁相环是否锁定到
        预期的频率上。注意，本功能需要一个稳定的参考时钟(32kHz RTC时钟或者CFG_8XX_XIN)
                Define this to measure the actual CPU clock instead
                of relying on the correctness of the configured
                values. Mostly useful for board bringup to make sure
                the PLL is locked at the intended frequency. Note
                that this requires a (stable) reference clock (32 kHz
                RTC clock or CFG_8XX_XIN)

- Intel Monahans options:
                CFG_MONAHANS_RUN_MODE_OSC_RATIO

                Defines the Monahans run mode to oscillator
                ratio. Valid values are 8, 16, 24, 31. The core
                frequency is this value multiplied by 13 MHz.
                定义Monahans的运行模式频率与晶振频率的比值. 有效值为8, 16, 24, 31.
                核心频率为该值乘以13MHz.
                CFG_MONAHANS_TURBO_RUN_MODE_RATIO

                Defines the Monahans turbo mode to oscillator
                ratio. Valid values are 1 (default if undefined) and
                2. The core frequency as calculated above is multiplied
                by this value.
                定义Monahans turbo 模式频率与晶振频率的比值. 有效值为1(不定义的默认值)
                和2. 核心频率是上面计算出的值与该值的乘积.
- Linux Kernel Interface:
- Linux内核接口:
                CONFIG_CLOCKS_IN_MHZ
                U-Boot在内部使用Hz保存所有时钟信息。为了与旧的Linux内核(要求bd_info数据内
                的时钟是MHZ单位)达到二进制兼容，可以定义环境变量"clocks_in_mhz"，U-Boot在
                传递给Linux内核前，将时钟数据转换为MHZ。
                当定义了 CONFIG_CLOCKS_IN_MHZ 时，"clocks_in_mhz=1"的定义会自动包含到默
                认的环境中。
                U-Boot stores all clock information in Hz
                internally. For binary compatibility with older Linux
                kernels (which expect the clocks passed in the
                bd_info data to be in MHz) the environment variable
                "clocks_in_mhz" can be defined so that U-Boot
                converts clock data to MHZ before passing it to the
                Linux kernel.
                When CONFIG_CLOCKS_IN_MHZ is defined, a definition of
                "clocks_in_mhz=1" is  automatically  included  in  the
                default environment.

                CONFIG_MEMSIZE_IN_BYTES           [relevant for MIPS only]
                CONFIG_MEMSIZE_IN_BYTES        [只与 MIPS 相关]
```

```
		当向linux传递memsize参数时，有一些版本的单位是字节，其它的则是MB，如果定义
		CONFIG_MEMSIZE_IN_BYTES，则参数单位为字节。
		When transfering memsize parameter to linux, some versions
		expect it to be in bytes, others in MB.
		Define CONFIG_MEMSIZE_IN_BYTES to make it in bytes.

		CONFIG_OF_FLAT_TREE
		新的内核版本要求使用平坦设备树(基于开放固件的概念)将固件设置传递给内核
		New kernel versions are expecting firmware settings to be
		passed using flat open firmware trees.
		The environment variable "disable_of", when set, disables this
		functionality.

		CONFIG_OF_FLAT_TREE_MAX_SIZE

		The maximum size of the constructed OF tree.
		FLAT tree结构最大容量

		OF_CPU - The proper name of the cpus node.
		OF_SOC - The proper name of the soc node.
		OF_TBCLK - The timebase frequency.
		OF_STDOUT_PATH - The path to the console device

		CONFIG_OF_HAS_BD_T

		The resulting flat device tree will have a copy of the bd_t.
		Space should be pre-allocated in the dts for the bd_t.
		最终的flat设备tree会保留一份bd_t的copy，在dts应预留空间分配给bd_t.

		CONFIG_OF_HAS_UBOOT_ENV

		The resulting flat device tree will have a copy of u-boot's
		environment variables
		最终的flat设备tree会保留一份uboot环境变量的copy

		CONFIG_OF_BOARD_SETUP

		Board code has addition modification that it wants to make
		to the flat device tree before handing it off to the kernel
		单板在将平坦设备树传递给内核前需要做额外的修改

		CONFIG_OF_BOOT_CPU

		This define fills in the correct boot cpu in the boot
		param header, the default value is zero if undefined.
		 该定义填充到boot参数头部的CPU部分。如果不定义，默认值为0。

- Serial Ports:
–	串行端口
		CFG_PL010_SERIAL

		Define this if you want support for Amba PrimeCell PL010 UARTs.
		如果希望支持Amba PrimeCell PL010串口控制器，则定义此宏。
		CFG_PL011_SERIAL

		Define this if you want support for Amba PrimeCell PL011 UARTs.
		果希望支持Amba PrimeCell PL011串口控制器，则定义此宏。

		CONFIG_PL011_CLOCK

		If you have Amba PrimeCell PL011 UARTs, set this variable to
		the clock speed of the UARTs.
		如果你使用Amba PrimeCell PL011串口控制器，将该宏定义为串口控制器的时钟频率。

		CONFIG_PL01x_PORTS

		If you have Amba PrimeCell PL010 or PL011 UARTs on your board,
		define this to a list of base addresses for each (supported)
		port. See e.g. include/configs/versatile.h
		如果你的单板使用Amba PrimeCell PL010 或者 PL011 串口控制器，定义该宏为一个列
		表(注：即数组初始化列表，该数组类型是一个指针数组)，列表里每一项为串口的基地址。例子
		参见 "include/configs/versatile.h"

- Console Interface:
–控制台终端接口
		根据单板的情况，定义一个串口终端(比如 CONFIG_8xx_CONS_SMC1, CONFIG_8xx_CONS_
		SMC2,CONFIG_8xx_CONS_SCC1, ...)，或者定义CONFIG_8xx_CONS_NONE来关闭串口终端

		Depending on board, define exactly one serial port
		(like CONFIG_8xx_CONS_SMC1, CONFIG_8xx_CONS_SMC2,
		CONFIG_8xx_CONS_SCC1, ...), or switch off the serial
		console by defining CONFIG_8xx_CONS_NONE

		Note: if CONFIG_8xx_CONS_NONE is defined, the serial
		port routines must be defined elsewhere
		(i.e. serial_init(), serial_getc(), ...)
		注意：如果定义了 CONFIG_8xx_CONS_NONE，串口例程必须在其它地方定义(比如 serial_init(),
		serial_getc(), ...)

		CONFIG_CFB_CONSOLE
		Enables console device for a color framebuffer. Needs following
		defines (cf. smiLynxEM, i8042, board/eltec/bab7xx)
		使能终端设备的彩色帧缓冲. 需要下列定义（cf. smiLynxEM, i8042, board/eltec/bab7xx）

			VIDEO_FB_LITTLE_ENDIAN		graphic memory organisation
			图像存储组织（默认为大端序）
						(default big endian)
			VIDEO_HW_RECTFILL	graphic chip supports
			图像芯片支持矩形填充(rectangle fill）
						rectangle fill
						(cf. smiLynxEM)
			VIDEO_HW_BITBLT			graphic chip supports
			图像芯片支持位块传输(bit-blit)（cf. smiLynxEM）
						bit-blit (cf. smiLynxEM)
			VIDEO_VISIBLE_COLS	visible pixel columns
			可视像素列（cols=pitch）
						(cols=pitch)
			VIDEO_VISIBLE_ROWS	visible pixel rows
			可视像素行
			VIDEO_PIXEL_SIZE	bytes per pixel
			每像素字节数
			VIDEO_DATA_FORMAT	graphic data format
			图像数据格式（0–5，cf. cfb_console.c）
						(0-5, cf. cfb_console.c)
			VIDEO_FB_ADRS		framebuffer address
			帧缓冲地址
			VIDEO_KBD_INIT_FCT	keyboard int fct
						(i.e. i8042_kbd_init())
			VIDEO_TSTC_FCT			test char fct
						(i.e. i8042_tstc)
			VIDEO_GETC_FCT			get char fct
						(i.e. i8042_getc)
			CONFIG_CONSOLE_CURSOR		cursor drawing on/off
			打开/关闭 光标绘制（需要 blink timer cf. i8042.c）
						(requires blink timer
						cf. i8042.c)
			CFG_CONSOLE_BLINK_COUNT blink interval (cf. i8042.c)
			闪烁间隔
			CONFIG_CONSOLE_TIME	display time/date info in
			在右上角显示时间/日期信息（需要 CONFIG_CMD_DATE）
						upper right corner
						(requires CFG_CMD_DATE)
			CONFIG_VIDEO_LOGO	display Linux logo in
			在左上角显示Linux logo
						upper left corner
			CONFIG_VIDEO_BMP_LOGO		use bmp_logo.h instead of
			使用bmp_logo.h作为logo(默认是linux_logo.h)。 需要
						linux_logo.h for logo.
						Requires CONFIG_VIDEO_LOGO
			CONFIG_CONSOLE_EXTRA_INFO
			除了logo外的其它单板信息
```

```
                                        addional board info beside
                                        the logo

                When CONFIG_CFB_CONSOLE is defined, video console is
                default i/o. Serial console can be forced with
                environment 'console=serial'.
                如果定义了CONFIG_CFB_CONSOLE, 视频终端是默认的i/o. 使用'console
                =serial'环境可以强制使用串口终端
                When CONFIG_SILENT_CONSOLE is defined, all console
                messages (by U-Boot and Linux!) can be silenced with
                the "silent" environment variable. See
                doc/README.silent for more information.
                如果定义了 CONFIG_SILENT_CONSOLE, 可以用'silent'环境变量屏蔽所有终端
                消息(包括U-Boot和Linux的输出!)。更多信息见doc/README.silent

- Console Baudrate:
控制台波特率

                CONFIG_BAUDRATE - in bps
                选择一个CFG_BAUDRATE_TABLE列出的波特率。
                Select one of the baudrates listed in
                CFG_BAUDRATE_TABLE, see below.

                CFG_BRGCLK_PRESCALE, baudrate prescale
                波特率预分频系数

- Interrupt driven serial port input:
终端驱动的串口输入

                CONFIG_SERIAL_SOFTWARE_FIFO

                PPC405GP only.仅用于PPC405GP。
                使用一个中断处理程序来接收串口数据。它将使能并使用硬件握手(RTS/CTS)和串口内
                置FIFO。必须设置中断驱动的输入缓冲的字节数
                Use an interrupt handler for receiving data on the
                serial port. It also enables using hardware handshake
                (RTS/CTS) and UART's built-in FIFO. Set the number of
                bytes the interrupt driven input buffer should have.

                不定义该宏则禁用此特性，包括禁用缓冲和硬件握手。
                Leave undefined to disable this feature, including
                disable the buffer and hardware handshake.

- Console UART Number:
控制台UART编号
                CONFIG_UART1_CONSOLE

                AMCC PPC4xx only.
                如果定义该宏，则使用内部UART1作为默认的U-Boot终端（否则使用UART0）
                If defined internal UART1 (and not UART0) is used
                as default U-Boot console.

- Boot Delay:   CONFIG_BOOTDELAY - in seconds
引导延时，单位秒

                Delay before automatically booting the default image;
                set to -1 to disable autoboot.
                在自动引导默认映像前的延时，设为-1表示禁用自动引导。

                See doc/README.autoboot for these options that
                work with CONFIG_BOOTDELAY. None are required.
                与CONFIG_BOOTDELAY相关的选项见 doc/README.autoboot. 这些选项都不是必需的。

                CONFIG_BOOT_RETRY_TIME
                CONFIG_BOOT_RETRY_MIN
                CONFIG_AUTOBOOT_KEYED
                CONFIG_AUTOBOOT_PROMPT
                CONFIG_AUTOBOOT_DELAY_STR
                CONFIG_AUTOBOOT_STOP_STR
                CONFIG_AUTOBOOT_DELAY_STR2
                CONFIG_AUTOBOOT_STOP_STR2
```

```
		CONFIG_ZERO_BOOTDELAY_CHECK
		CONFIG_RESET_TO_RETRY

- Autoboot Command:
自动引导命令

		CONFIG_BOOTCOMMAND
		只有定义了CONFIG_BOOTDELAY时，才需要定义这个宏。如果在引导延时内没有字符输入，
		则自动执行该宏定义的命令字符串。
		Only needed when CONFIG_BOOTDELAY is enabled;
		define a command string that is automatically executed
		when no character is read on the console interface
		within "Boot Delay" after reset.

		CONFIG_BOOTARGS
		该宏用于向bootm命令传递参数。CONFIG_BOOTARGS的值也被赋给环境变量"bootargs"
		This can be used to pass arguments to the bootm
		command. The value of CONFIG_BOOTARGS goes into the
		environment value "bootargs".

		CONFIG_RAMBOOT and CONFIG_NFSBOOT
		两个宏值分别被赋给环境变量"ramboot"和"nfsboot"。用于简化从RAM和NFS两种引导途径之间的切换。
		The value of these goes into the environment as
		"ramboot" and "nfsboot" respectively, and can be used
		as a convenience, when switching between booting from
		ram and nfs.

- Pre-Boot Commands:
预引导命令

		CONFIG_PREBOOT
		如果定义了该选项，则在进行引导延时的计时前或者运行自动引导命令前，检查环境变量"preboot"是否存在，
		如果存在则进入交互模式。
		When this option is #defined, the existence of the
		environment variable "preboot" will be checked
		immediately before starting the CONFIG_BOOTDELAY
		countdown and/or running the auto-boot command resp.
		entering interactive mode.

		该功能在"preboot"是由程序自动生成或修改的情况下比较有用。比如，LWMON单板的代码：当引导系统时，如果
		用户按下特定组合键，preboot会被修改。
		This feature is especially useful when "preboot" is
		automatically generated or modified. For an example
		see the LWMON board specific code: here "preboot" is
		modified when the user holds down a certain
		combination of keys on the (special) keyboard when
		booting the systems

- Serial Download Echo Mode:
串口下载回显模式

		CONFIG_LOADS_ECHO
		If defined to 1, all characters received during a
		serial download (using the "loads" command) are
		echoed back. This might be needed by some terminal
		emulations (like "cu"), but may as well just take
		time on others. This setting #define's the initial
		value of the "loads_echo" environment variable.
		如果定义为1，在串口下载(使用"loads"命令)过程中，会对所有收到的字符进行回显。在某些终端上可能有用(如"cu")
		，但对大多数终端只是浪费时间。这个设置定义了"loads_echo"环境变量的默认值。
- Kgdb Serial Baudrate: (if CFG_CMD_KGDB is defined)
Kgdb串口波特率
		CONFIG_KGDB_BAUDRATE
		从CFG_BAUDRATE_TABLE里面选择一个波特率。
		Select one of the baudrates listed in
		CFG_BAUDRATE_TABLE, see below.

- Monitor Functions:
Monitor功能

		CONFIG_COMMANDS
```

```
Most monitor functions can be selected (or
de-selected) by adjusting the definition of
CONFIG_COMMANDS; to select individual functions,
#define CONFIG_COMMANDS by "OR"ing any of the
following values:
大多数monitor功能可以调整CONFIG_COMMANDS定义来选择或者去掉；用#define CONFIG_COMMANDS或上任何
下面的值来选择功能。

#define enables commands:
-------------------------
CFG_CMD_ASKENV                    * ask for env variable
CFG_CMD_AUTOSCRIPT  Autoscript Support
CFG_CMD_BDI                       bdinfo
CFG_CMD_BEDBUG                    * Include BedBug Debugger
CFG_CMD_BMP                       * BMP support
CFG_CMD_BSP                       * Board specific commands
CFG_CMD_BOOTD              bootd
CFG_CMD_CACHE              * icache, dcache
CFG_CMD_CONSOLE            coninfo
CFG_CMD_DATE               * support for RTC, date/time...
CFG_CMD_DHCP               * DHCP support
CFG_CMD_DIAG               * Diagnostics
CFG_CMD_DOC                       * Disk-On-Chip Support
CFG_CMD_DTT                       * Digital Therm and Thermostat
CFG_CMD_ECHO               echo arguments
CFG_CMD_EEPROM                    * EEPROM read/write support
CFG_CMD_ELF                       * bootelf, bootvx
CFG_CMD_ENV                       saveenv
CFG_CMD_FDC                       * Floppy Disk Support
CFG_CMD_FAT                       * FAT partition support
CFG_CMD_FDOS               * Dos diskette Support
CFG_CMD_FLASH              flinfo, erase, protect
CFG_CMD_FPGA               FPGA device initialization support
CFG_CMD_HWFLOW                    * RTS/CTS hw flow control
CFG_CMD_I2C                       * I2C serial bus support
CFG_CMD_IDE                       * IDE harddisk support
CFG_CMD_IMI                       iminfo
CFG_CMD_IMLS               List all found images
CFG_CMD_IMMAP              * IMMR dump support
CFG_CMD_IRQ                       * irqinfo
CFG_CMD_ITEST              Integer/string test of 2 values
CFG_CMD_JFFS2              * JFFS2 Support
CFG_CMD_KGDB               * kgdb
CFG_CMD_LOADB              loadb
CFG_CMD_LOADS              loads
CFG_CMD_MEMORY                    md, mm, nm, mw, cp, cmp, crc, base,
                                                 loop, loopw, mtest
CFG_CMD_MISC               Misc functions like sleep etc
CFG_CMD_MMC                       * MMC memory mapped support
CFG_CMD_MII                       * MII utility commands
CFG_CMD_NAND               * NAND support
CFG_CMD_NET                       bootp, tftpboot, rarpboot
CFG_CMD_PCI                       * pciinfo
CFG_CMD_PCMCIA                    * PCMCIA support
CFG_CMD_PING               * send ICMP ECHO_REQUEST to network host
CFG_CMD_PORTIO                    * Port I/O
CFG_CMD_REGINFO            * Register dump
CFG_CMD_RUN                       run command in env variable
CFG_CMD_SAVES              * save S record dump
CFG_CMD_SCSI               * SCSI Support
CFG_CMD_SDRAM              * print SDRAM configuration information
                                                 (requires CFG_CMD_I2C)
CFG_CMD_SETGETDCR   Support for DCR Register access (4xx only)
CFG_CMD_SPI                       * SPI serial bus support
CFG_CMD_USB                       * USB support
CFG_CMD_VFD                       * VFD support (TRAB)
CFG_CMD_BSP                       * Board SPecific functions
CFG_CMD_CDP                       * Cisco Discover Protocol support
----------------------------------------------
CFG_CMD_ALL   all

CONFIG_CMD_DFL      Default configuration;
```

```
             默认配置，目前为止包含上面列举的所有不含星号的命令。
             at the moment this is includes all commands, except the ones marked
             with "*" in the listabove.

             If you don't define CONFIG_COMMANDS it defaults to
             CONFIG_CMD_DFL in include/cmd_confdefs.h. A board can
             override the default settings in the respective
             include file.
             如果你没有在include/cmd_confdefs.h定义默认的命令，各个单板可以在各自的包含头文件
             覆盖默认设定。

             EXAMPLE: If you want all functions except of network
             support you can write:
             假设你想包含除了网络以外所有的功能，你可以这样写：
             #define CONFIG_COMMANDS (CFG_CMD_ALL & ~CFG_CMD_NET)

             注意：如果你不清楚，请不要开启"icache"和"dcache"命令(配置参数CONFIG_CMD_CACHE)。
             在8xx或8260上无法使能数据cache(访问IMMR区间必须不过cache)，在其它使用数据cache
             作为初期栈和数据保存的系统里也不能禁用数据cache。
       Note: Don't enable the "icache" and "dcache" commands
             (configuration option CFG_CMD_CACHE) unless you know
             what you (and your U-Boot users) are doing. Data
             cache cannot be enabled on systems like the 8xx or
             8260 (where accesses to the IMMR region must be
             uncached), and it cannot be disabled on all other
             systems where we (mis-) use the data cache to hold an
             initial stack and some data.


             XXX - this list needs to get updated!

- Watchdog:
看门狗
             CONFIG_WATCHDOG
             如果定义该变量，则使能看门狗支持。必须有平台特定的看门狗实现。对8xx和8260 CPU，SIU看
             门狗功能在SYPCR寄存器中使能。
             If this variable is defined, it enables watchdog
             support. There must be support in the platform specific
             code for a watchdog. For the 8xx and 8260 CPUs, the
             SIU Watchdog feature is enabled in the SYPCR
             register.

- U-Boot Version:
版本
             CONFIG_VERSION_VARIABLE
             如果定义该宏，U-Boot创建一个"ver"环境变量，用于显示"version"命令所打印出的U-Boot版本。本变量是只读的。
             If this variable is defined, an environment variable
             named "ver" is created by U-Boot showing the U-Boot
             version as printed by the "version" command.
             This variable is readonly.

- Real-Time Clock:
实时时钟

             When CFG_CMD_DATE is selected, the type of the RTC
             has to be selected, too. Define exactly one of the
             following options:
             如果选择了 CONFIG_CMD_DATE，需要同时选择RTC的类型。使用下列定义中的一个：
             CONFIG_RTC_MPC8xx    - use internal RTC of MPC8xx
             CONFIG_RTC_PCF8563   - use Philips PCF8563 RTC
             CONFIG_RTC_MC146818  - use MC146818 RTC
             CONFIG_RTC_DS1307    - use Maxim, Inc. DS1307 RTC
             CONFIG_RTC_DS1337    - use Maxim, Inc. DS1337 RTC
             CONFIG_RTC_DS1338    - use Maxim, Inc. DS1338 RTC
             CONFIG_RTC_DS164x    - use Dallas DS164x RTC
             CONFIG_RTC_MAX6900   - use Maxim, Inc. MAX6900 RTC

             Note that if the RTC uses I2C, then the I2C interface
             must also be configured. See I2C Support, below.
             注意：如果RTC使用I2C，则需要配置I2C接口。参加见下面的I2C支持。
- Timestamp Support:
```

```
– 时间戳支持:
              如果定义CONFIG_TIMESTAMP，与映像相关的命令如bootm或iminfo会
              打印给定映像的时间戳(日期和时间)。如果定义了CONFIG_CMD_DATE，该选项自动使能。
              When CONFIG_TIMESTAMP is selected, the timestamp
              (date and time) of an image is printed by image
              commands like bootm or iminfo. This option is
              automatically enabled when you select CFG_CMD_DATE .

- Partition Support:
分区支持

              CONFIG_MAC_PARTITION and/or CONFIG_DOS_PARTITION
              and/or CONFIG_ISO_PARTITION

              如果使能了IDE或SCSI支持(CONFIG_CMD_IDE 或者    CONFIG_CMD_SCSI)，你必须至少配置一种分区类型的支持
              If IDE or SCSI support     is  enabled  (CFG_CMD_IDE  or
              CFG_CMD_SCSI) you must configure support for at least
              one partition type as well.

- IDE Reset method:
IDE复位方式
              CONFIG_IDE_RESET_ROUTINE - this is defined in several
              board configurations files but used nowhere!
              在几个单板的配置文件中定义，但并未使用！
              CONFIG_IDE_RESET - is this is defined, IDE Reset will
              be performed by calling the function
              如果定义了，会调用下面的函数复位IDE，
                     ide_set_reset(int reset)
              which has to be defined in a board specific file
              这个函数必须在单板相关文件里定义。
- ATAPI Support:
              CONFIG_ATAPI

              Set this to enable ATAPI support.

- LBA48 Support
              CONFIG_LBA48
              定义该宏可支持大于137GB的磁盘。需要检查CFG_64BIT_LBA, CFG_64BIT_VSPRINTF 和 CFG_64BIT_STRTOUL
              选项。如果不定义它们，LBA48使用32位变量，只能支持到最大2.1TB的磁盘。
              Set this to enable support for disks larger than 137GB
              Also look at CFG_64BIT_LBA ,CFG_64BIT_VSPRINTF and CFG_64BIT_STRTOUL
              Whithout these , LBA48 support uses 32bit variables and will 'only'
              support disks up to 2.1TB.

              CFG_64BIT_LBA:
                     When enabled, makes the IDE subsystem use 64bit sector addresses.
                     Default is 32bit.
                     如果使能，则IDE子系统使用64位的扇区地址，默认是32位。
- SCSI Support:
              当前只支持 SYM53C8XX SCSI 控制器；定义 CONFIG_SCSI_SYM53C8XX 来使能。
              At the moment only there is only support for the
              SYM53C8XX SCSI controller; define
              CONFIG_SCSI_SYM53C8XX to enable it.

              CFG_SCSI_MAX_LUN [8], CFG_SCSI_MAX_SCSI_ID [7] and
              CFG_SCSI_MAX_DEVICE [CFG_SCSI_MAX_SCSI_ID *
              CFG_SCSI_MAX_LUN] can be adjusted to define the
              maximum numbers of LUNs, SCSI ID's and target
              devices.
              CFG_SCSI_SYM53C8XX_CCF to fix clock timing (80Mhz)
              可以调整CFG_SCSI_MAX_LUN [8], CFG_SCSI_MAX_SCSI_ID [7] 及 CFG_SCSI_MAX_DEVICE
              [CFG_SCSI_MAX_SCSI_ID * CFG_SCSI_MAX_LUN] 定义最大LUN, SCSI ID及目标的最大设备数。
                定义CFG_SCSI_SYM53C8XX_CCF 以修正clock timing(时钟时序?) (80Mhz)

- NETWORK Support (PCI):
– 网络支持（PCI设备）:
              CONFIG_E1000
              Support for Intel 8254x gigabit chips.

              CONFIG_EEPRO100
              支持Intel 82557/82559/82559ER芯片。还可以定义CONFIG_EEPRO100_SROM_WRITE使能首次初始
```

```
		化时的EEPROM写流程.
		Support for Intel 82557/82559/82559ER chips.
		Optional CONFIG_EEPRO100_SROM_WRITE enables eeprom
		write routine for first time initialisation.

		CONFIG_TULIP
		Support for Digital 2114x chips.
		Optional CONFIG_TULIP_SELECT_MEDIA for board specific
		modem chip initialisation (KS8761/QS6611).
		支持 Digital 2114x 芯片。
		 还可以定义 CONFIG_TULIP_SELECT_MEDIA 支持单板特定的modem芯片初始化(KS8761/QS6611).

		CONFIG_NATSEMI
		Support for National dp83815 chips.

		CONFIG_NS8382X
		Support for National dp8382[01] gigabit chips.

- NETWORK Support (other):
– 网络支持（其它设备):

		CONFIG_DRIVER_LAN91C96
		Support for SMSC's LAN91C96 chips.

			CONFIG_LAN91C96_BASE
			Define this to hold the physical address
			of the LAN91C96's I/O space

			CONFIG_LAN91C96_USE_32_BIT
			Define this to enable 32 bit addressing

		CONFIG_DRIVER_SMC91111
		Support for SMSC's LAN91C111 chip

			CONFIG_SMC91111_BASE
			Define this to hold the physical address
			of the device (I/O space)

			CONFIG_SMC_USE_32_BIT
			Define this if data bus is 32 bits

			CONFIG_SMC_USE_IOFUNCS
			Define this to use i/o functions instead of macros
			(some hardware wont work with macros)

- USB Support:
		目前只支持UHCI host controller (PIP405, MIP405, MPC5200); 定义 CONFIG_USB_UHCI 来使能。
		定义 CONFIG_USB_KEYBOARD 使能USB键盘支持，定义 CONFIG_USB_STORAGE 使能USB存储设备。
		At the moment only the UHCI host controller is
		supported (PIP405, MIP405, MPC5200); define
		CONFIG_USB_UHCI to enable it.
		define CONFIG_USB_KEYBOARD to enable the USB Keyboard
		and define CONFIG_USB_STORAGE to enable the USB
		storage devices.

	Note:
		注意: 受支持的是USB键盘和USB软驱（TEAC FD-05PUB).
		Supported are USB Keyboards and USB Floppy drives
		(TEAC FD-05PUB).
		MPC5200 USB requires additional defines:
			CONFIG_USB_CLOCK
				for 528 MHz Clock: 0x0001bbbb
			CONFIG_USB_CONFIG
				for differential drivers: 0x00001000
				for single ended drivers: 0x00005000


- MMC Support:
		U-Boot支持Intel PXA上的MMC控制器，定义CONFIG_MMC以使能支持。用与flash类似的方式将MMC设备映射到物理
内存，即
		可在boot提示符下访问设备。CONFIG_CMD_MMC使能命令行。MMC驱动支持FAT文件系统，可以通过CONFIG_CMD_FAT
```

```
开启.
		The MMC controller on the Intel PXA is supported. To
		enable this define CONFIG_MMC. The MMC can be
		accessed from the boot prompt by mapping the device
		to physical memory similar to flash. Command line is
		enabled with CFG_CMD_MMC. The MMC driver also works with
		the FAT fs. This is enabled with CFG_CMD_FAT.

- Journaling Flash filesystem support:
– 日志Flash文件系统支持:
		CONFIG_JFFS2_NAND, CONFIG_JFFS2_NAND_OFF, CONFIG_JFFS2_NAND_SIZE,
		CONFIG_JFFS2_NAND_DEV
		Define these for a default partition on a NAND device
		用于NAND设备上的默认分区的参数
		CFG_JFFS2_FIRST_SECTOR,
		CFG_JFFS2_FIRST_BANK, CFG_JFFS2_NUM_BANKS
		Define these for a default partition on a NOR device
		用于NOR设备上的默认分区的参数

		CFG_JFFS_CUSTOM_PART
		定义此宏来创建一个私有分区，你必须提供一个函数: struct part_info* jffs2_part_info(int part_num)
		Define this to create an own partition. You have to provide a
		function struct part_info* jffs2_part_info(int part_num)

		如果只定义一个JFFS2分区，也可以定义下面的宏来禁用chpart命令。
	#define CFG_JFFS_SINGLE_PART	1
		在没有自定义分区的时候，这个是默认行为。
		If you define only one JFFS2 partition you may also want to
		#define CFG_JFFS_SINGLE_PART	1
		to disable the command chpart. This is the default when you
		have not defined a custom partition

- Keyboard Support:
– 键盘支持:
		CONFIG_ISA_KEYBOARD

		定义该宏以支持标准PC键盘
		Define this to enable standard (PC-Style) keyboard
		support

		CONFIG_I8042_KBD
		支持标准PC键盘，US(默认)和GERMAN键盘布局(通过"keymap=de"切换)。为cfb_console提供函数
i8042_kbd_init,
		i8042_tstc和i8042_getc。支持光标闪烁。
		Standard PC keyboard driver with US (is default) and
		GERMAN key layout (switch via environment 'keymap=de') support.
		Export function i8042_kbd_init, i8042_tstc and i8042_getc
		for cfb_console. Supports cursor blinking.

- Video support:
–视频支持

		CONFIG_VIDEO
		定义该宏以支持视频（输出到视频).
		Define this to enable video support (for output to
		video).

		CONFIG_VIDEO_CT69000

		支持 Chips & Technologies 69000 视频芯片
		Enable Chips & Technologies 69000 Video chip

		CONFIG_VIDEO_SMI_LYNXEM
		使能Silicon Motion SMI 712/710/810 视频芯片. 视频输出由环境变量'videoout' (1 = LCD and 2 =
CRT).
		如果不定义videoout,默认为CRT
		Enable Silicon Motion SMI 712/710/810 Video chip. The
		video output is selected via environment 'videoout'
		(1 = LCD and 2 = CRT). If videoout is undefined, CRT is
		assumed.
```

For the CT69000 and SMI_LYNXEM drivers, videomode is
selected via environment 'videomode'. Two diferent ways
are possible:
对CT69000 和 SMI_LYNXEM 驱动，视频模式由环境变量'videomode'选择，有两个选择：
- "videomode=num"   'num' is a standard LiLo mode numbers.
Following standard modes are supported   (* is default):
支持下列标准模式 (* 为默认值)：

```
      Colors  640x480 800x600 1024x768 1152x864 1280x1024
-------------+---------------------------------------------
      8 bits |      0x301* 0x303   0x305   0x161          0x307
     15 bits |      0x310  0x313   0x316   0x162          0x319
     16 bits |      0x311  0x314   0x317   0x163          0x31A
     24 bits |      0x312  0x315   0x318     ?      0x31B
-------------+---------------------------------------------
(i.e. setenv videomode 317; saveenv; reset;)
```

- "videomode=bootargs" 所有视频参数从bootargs提取。(见 drivers/video/videomodes.c)
- "videomode=bootargs" all the video parameters are parsed
from the bootargs. (See drivers/videomodes.c)

CONFIG_VIDEO_SED13806
使能 Epson SED13806 驱动。该驱动支持8bpp和16bpp模式(由CONFIG_VIDEO_SED13806_8BPP
或CONFIG_VIDEO_SED13806_16BPP定义)
Enable Epson SED13806 driver. This driver supports 8bpp
and 16bpp modes defined by CONFIG_VIDEO_SED13806_8BPP
or CONFIG_VIDEO_SED13806_16BPP

- Keyboard Support:
– 键盘支持：
CONFIG_KEYBOARD
定义该宏来使能自定义键盘支持。它简单地调用drv_keyboard_init()，该函数需要在单板相关文件中提供。
目前使用该宏的单板只有RBC823。
Define this to enable a custom keyboard support.
This simply calls drv_keyboard_init() which must be
defined in your board-specific files.
The only board using this so far is RBC823.

- LCD Support: CONFIG_LCD

定义该宏以支持 LCD (输出至LCD)；同时从下列的显示支持中选择一种：
Define this to enable LCD support (for output to LCD
display); also select one of the supported displays
by defining one of these:

CONFIG_NEC_NL6448AC33:

NEC NL6448AC33-18. Active, color, single scan.

CONFIG_NEC_NL6448BC20

NEC NL6448BC20-08. 6.5", 640x480.
Active, color, single scan.

CONFIG_NEC_NL6448BC33_54

NEC NL6448BC33-54. 10.4", 640x480.
Active, color, single scan.

CONFIG_SHARP_16x9

Sharp 320x240. Active, color, single scan.
It isn't 16x9, and I am not sure what it is.

CONFIG_SHARP_LQ64D341

Sharp LQ64D341 display, 640x480.
Active, color, single scan.

CONFIG_HLD1045

```
			HLD1045 display, 640x480.
			Active, color, single scan.

		CONFIG_OPTREX_BW

			Optrex	 CBL50840-2 NF-FW 99 22 M5
			or
			Hitachi	 LMG6912RPFC-00T
			or
			Hitachi	 SP14Q002

			320x240. Black & white.

		Normally display is black on white background; define
		CFG_WHITE_ON_BLACK to get it inverted.

- Splash Screen Support: CONFIG_SPLASH_SCREEN
		如果设置该选项，将检查环境变量"splashimage". 如果变量存在则显示一张BMP图像，普通的logo，版权及系统
		信息不再显示. "splashimage"指定了位图的存放地址. 终端也被重定向到"nulldev". 由于启动的早期就会加载
		Splash screen，使用它可以实现安静的启动.
		If this option is set, the environment is checked for
		a variable "splashimage". If found, the usual display
		of logo, copyright and system information on the LCD
		is suppressed and the BMP image at the address
		specified in "splashimage" is loaded instead. The
		console is redirected to the "nulldev", too. This
		allows for a "silent" boot where a splash screen is
		loaded very quickly after power-on.

- Gzip compressed BMP image support: CONFIG_VIDEO_BMP_GZIP
		  如果设置该选项，除了支持标准BMP图像外，splashscreen或者bmp命令可以使用gzip压缩的BMP图像.
		If this option is set, additionally to standard BMP
		images, gzipped BMP images can be displayed via the
		splashscreen support or the bmp command.

- Compression support:
压缩支持

		CONFIG_BZIP2
		 如果设置了该选项，u-boot将包含对bzip2压缩映像的支持。如果不设置该选项，则只支持未压缩的或者gzip压缩的
映像。
		If this option is set, support for bzip2 compressed
		images is included. If not, only uncompressed and gzip
		compressed images are supported.

		注意: bzip2 算法需要大量的RAM，因此malloc区大小(通过CFG_MALLOC_LEN定义)至少为4MB。
		NOTE: the bzip2 algorithm requires a lot of RAM, so
		the malloc area (as defined by CFG_MALLOC_LEN) should
		be at least 4MB.

- MII/PHY support:
-MII/PHU支持

		CONFIG_PHY_ADDR

		The address of PHY on MII bus.
		PHY在MII总线上的地址
		CONFIG_PHY_CLOCK_FREQ (ppc4xx)
		MII总线的时钟频率。
		The clock frequency of the MII bus

		CONFIG_PHY_GIGE
		  如果设置该选项，则支持GE PHY的速度/双工检测
		If this option is set, support for speed/duplex
		detection of Gigabit PHY is included.

		CONFIG_PHY_RESET_DELAY
		有一些PHY (如Intel LXT971A) 在复位后需要经过一定的时延才能访问MII寄存器，需要设置该
		宏。该设置的单位为usec (LXT971A至少需设为300usec)
		Some PHY like Intel LXT971A need extra delay after
		reset before any MII register access is possible.
		For such PHY, set this option to the usec delay
		required. (minimum 300usec for LXT971A)

		CONFIG_PHY_CMD_DELAY (ppc4xx)
		有一些PHY (如Intel LXT971A) 在发起MII命令时，需要额外的时延才能读取MII状态寄存器。
		Some PHY like Intel LXT971A need extra delay after
		command issued before MII status register can be read

- Ethernet address:
以太网地址
		CONFIG_ETHADDR
		CONFIG_ETH2ADDR
		CONFIG_ETH3ADDR
		为对应的以太网接口定义一个默认的MAC地址，当不能自动确定时使用该定义.
		Define a default value for ethernet address to use
		for the respective ethernet interface, in case this
		is not determined automatically.

- IP address:
- IP 地址:
		CONFIG_IPADDR
		为默认的以太网接口定义一个默认的IP地址(在没有确定IP地址的时候使用，比如通过bootp确定IP地址)。
		Define a default value for the IP address to use for
		the default ethernet interface, in case this is not
		determined through e.g. bootp.

- Server IP address:
– 服务器 IP 地址:
		CONFIG_SERVERIP
		定义默认的TFTP服务器IP地址。在使用"tftboot"命令时，将使用该地址连接TFTP服务器
		Defines a default value for theIP address of a TFTP
		server to contact when using the "tftboot" command.

- BOOTP Recovery Mode:
- BOOTP 恢复模式:
		CONFIG_BOOTP_RANDOM_DELAY

		If you have many targets in a network that try to
		boot using BOOTP, you may want to avoid that all
		systems send out BOOTP requests at precisely the same
		moment (which would happen for instance at recovery
		from a power failure, when all systems will try to
		boot, thus flooding the BOOTP server. Defining
		CONFIG_BOOTP_RANDOM_DELAY causes a random delay to be
		inserted before sending out BOOTP requests. The
		following delays are insterted then:
		如果你的网络中有多个系统使用BOOTP启动，你可能需要避免出现所有系统同时发送BOOTP请求的情况(如因电源
		恢复产生的系统同时上电)，将导致BOOTP服务器过载. 定义 CONFIG_BOOTP_RANDOM_DELAY 将在发送BOOTP请
		求前插入随机延时. 插入规则如下:
		1st BOOTP request:  delay 0 ... 1 sec
		2nd BOOTP request:  delay 0 ... 2 sec
		3rd BOOTP request:  delay 0 ... 4 sec
		4th and following
		BOOTP requests:          delay 0 ... 8 sec

- DHCP Advanced Options:
- DHCP 高级选项:
		CONFIG_BOOTP_MASK

		You can fine tune the DHCP functionality by adding
		these flags to the CONFIG_BOOTP_MASK define:
		可以定义下列CONFIG_BOOTP_*宏来微调DHCP的功能:
		CONFIG_BOOTP_DNS2 - If a DHCP client requests the DNS
		serverip from a DHCP server, it is possible that more
		than one DNS serverip is offered to the client.
		If CONFIG_BOOTP_DNS2 is enabled, the secondary DNS
		serverip will be stored in the additional environment
		variable "dnsip2". The first DNS serverip is always
		stored in the variable "dnsip", when CONFIG_BOOTP_DNS
		is added to the CONFIG_BOOTP_MASK.
		CONFIG_BOOTP_SEND_HOSTNAME - Some DHCP servers are capable
		to do a dynamic update of a DNS server. To do this, they
		need the hostname of the DHCP requester.
		If CONFIG_BOOP_SEND_HOSTNAME is added to the
		CONFIG_BOOTP_MASK, the content of the "hostname"
		environment variable is passed as option 12 to
		the DHCP server.

 - CDP Options:
 - CDP 选项:
		CONFIG_CDP_DEVICE_ID

		The device id used in CDP trigger frames.

		CONFIG_CDP_DEVICE_ID_PREFIX
		添加到MAC地址的2个字符前缀.
		A two character string which is prefixed to the MAC address
		of the device.

		CONFIG_CDP_PORT_ID
		包含端口ascii名称的printf格式串. 一般设为"eth%d", 对第一个以太网口为eth0, 对第二个以太网口为eth1等等.
		A printf format string which contains the ascii name of
		the port. Normally is set to "eth%d" which sets
		eth0 for the first ethernet, eth1 for the second etc.

		CONFIG_CDP_CAPABILITIES
		表示设备能力的32位整数. 0x00000010 表示普通的不转发的主机.
		A 32bit integer which indicates the device capabilities;
		0x00000010 for a normal host which does not forwards.

		CONFIG_CDP_VERSION
		包含软件版本的ascii字符串.
		An ascii string containing the version of the software.

		CONFIG_CDP_PLATFORM
		包含平台名称的ascii字符串.
		An ascii string containing the name of the platform.

		CONFIG_CDP_TRIGGER

		A 32bit integer sent on the trigger.

		CONFIG_CDP_POWER_CONSUMPTION

		A 16bit integer containing the power consumption of the
		device in .1 of milliwatts.

		CONFIG_CDP_APPLIANCE_VLAN_TYPE

		A byte containing the id of the VLAN.

- Status LED:	CONFIG_STATUS_LED
		有几个配置可以用LED显示当前的状态. 比如, 在运行U-Boot代码时, LED快闪, 收到BOOTP应答时, 停止快闪, 在运行
linux
		内核时慢闪(由linux内核的状态LED驱动支持). 定义 CONFIG_STATUS_LED 可以打开U-Boot的这个功能.
		Several configurations allow to display the current
		status using a LED. For instance, the LED will blink
		fast while running U-Boot code, stop blinking as
		soon as a reply to a BOOTP request was received, and
		start blinking slow once the Linux kernel is running
		(supported by a status LED driver in the Linux
		kernel). Defining CONFIG_STATUS_LED enables this
		feature in U-Boot.

- CAN Support:	CONFIG_CAN_DRIVER
		定义 CONFIG_CAN_DRIVER 在支持 CAN 的系统上使能驱动支持(可选), 比如 TQM8xxL.
		Defining CONFIG_CAN_DRIVER enables CAN driver support
		on those systems that support this (optional)
		feature, like the TQM8xxL modules.
```

```
- I2C Support:      CONFIG_HARD_I2C | CONFIG_SOFT_I2C

              使能I2C串行总线命令. 定义任一个宏可以在选择的CPU内包含相应的I2C驱动(但不能同时定义).
              These enable I2C serial bus commands. Defining either of
              (but not both of) CONFIG_HARD_I2C or CONFIG_SOFT_I2C will
              include the appropriate I2C driver for the selected cpu.

              可以在u-boot命令行下使用i2c命令(只要在CONFIG_COMMANDS中设置了CONFIG_CMD_I2C),
              并与基于i2c的实时时钟芯片通讯。命令接口的说明见common/cmd_i2c.c.
              This will allow you to use i2c commands at the u-boot
              command line (as long as you set CFG_CMD_I2C in
              CONFIG_COMMANDS) and communicate with i2c based realtime
              clock chips. See common/cmd_i2c.c for a description of the
              command line interface.

              CONFIG_HARD_I2C selects the CPM hardware driver for I2C.
              选择一个I2C控制器硬件。

              CONFIG_SOFT_I2C configures u-boot to use a software (aka
              bit-banging) driver instead of CPM or similar hardware
              support for I2C.
              配置u-boot使用软件(即bit-banging)驱动代替CPM或相似的I2C支持。

              There are several other quantities that must also be
              defined when you define CONFIG_HARD_I2C or CONFIG_SOFT_I2C.
              你定义了CONFIG_HARD_I2C 或者 CONFIG_SOFT_I2C后，还有其它几个数值也需要定义。


              两种定义下都需要再定义 CFG_I2C_SPEED 为你预定的i2c总线运行频率(Hz单位)，定义CFG_I2C_SLAVE
              为本节点的地址(即CPU的i2c节点地址)。
              In both cases you will need to define CFG_I2C_SPEED
              to be the frequency (in Hz) at which you wish your i2c bus
              to run and CFG_I2C_SLAVE to be the address of this node (ie
              the cpu's i2c node address).

              目前，u-boot在mpc8xx上的i2c代码(cpu/mpc8xx/i2c.c)将CPU设为主节点，因此地址应设为0(见手册，
              如MPC823e User's Manual p.16-473)。将CFG_I2C_SLAVE设为0。
              Now, the u-boot i2c code for the mpc8xx (cpu/mpc8xx/i2c.c)
              sets the cpu up as a master node and so its address should
              therefore be cleared to 0 (See, eg, MPC823e User's Manual
              p.16-473). So, set CFG_I2C_SLAVE to 0.

              对CONFIG_HARD_I2C，上面就是所有需要的设置。
              That's all that's required for CONFIG_HARD_I2C.

              如果你使用软件i2c接口(CONFIG_SOFT_I2C)，还需要定义下列宏(例子取自include/configs/lwmon.h):
              If you use the software i2c interface (CONFIG_SOFT_I2C)
              then the following macros need to be defined (examples are
              from include/configs/lwmon.h):

              I2C_INIT
               (可选). 定义用于使能i2c控制器或配置端口的命令。
              (Optional). Any commands necessary to enable the I2C
              controller or configure ports.

              eg: #define I2C_INIT (immr->im_cpm.cp_pbdir |=  PB_SCL)

              I2C_PORT
              (仅用于MPC8260 CPU). 定义要使用的I/O (the code assumes both bits are on the same port).
              有效的值为0..3，对应端口A..D。
              (Only for MPC8260 CPU). The I/O port to use (the code
              assumes both bits are on the same port). Valid values
              are 0..3 for ports A..D.

              I2C_ACTIVE
              使I2C数据线处于激活状态(driven)的必要代码。如果数据线对collector是开放的，定义可以为空。
              The code necessary to make the I2C data line active
              (driven).  If the data line is open collector, this
              define can be null.

              eg: #define I2C_ACTIVE (immr->im_cpm.cp_pbdir |=  PB_SDA)
```

```
		I2C_TRISTATE
		使I2C数据线处于三态(非激活)的必要代码。如果数据线对collector是开放的，定义可以为空。
		The code necessary to make the I2C data line tri-stated
		(inactive).  If the data line is open collector, this
		define can be null.

		eg: #define I2C_TRISTATE (immr->im_cpm.cp_pbdir &= ~PB_SDA)

		I2C_READ
		定义一段代码，在I2C数据线为高时返回TRUE，为低时返回FALSE。
		Code that returns TRUE if the I2C data line is high,
		FALSE if it is low.

		eg: #define I2C_READ ((immr->im_cpm.cp_pbdat & PB_SDA) != 0)

		I2C_SDA(bit)
		如果 <bit> 为TRUE，将I2C总线置为高，否则置为低。
		If <bit> is TRUE, sets the I2C data line high. If it
		is FALSE, it clears it (low).

		eg: #define I2C_SDA(bit) \
			if(bit) immr->im_cpm.cp_pbdat |=  PB_SDA; \
			else    immr->im_cpm.cp_pbdat &= ~PB_SDA

		I2C_SCL(bit)
		如果 <bit> 为TRUE，将I2C时钟线置为高，否则置为低。
		If <bit> is TRUE, sets the I2C clock line high. If it
		is FALSE, it clears it (low).

		eg: #define I2C_SCL(bit) \
			if(bit) immr->im_cpm.cp_pbdat |=  PB_SCL; \
			else    immr->im_cpm.cp_pbdat &= ~PB_SCL

		I2C_DELAY
		每个时钟周期会调用该延时4次，因此它控制了数据传输速率。数据速率为 1 / (I2C_DELAY * 4)。通常定义成下面的
样子：
		This delay is invoked four times per clock cycle so this
		controls the rate of data transfer.  The data rate thus
		is 1 / (I2C_DELAY * 4). Often defined to be something
		like:

		#define I2C_DELAY  udelay(2)

		CFG_I2C_INIT_BOARD
		单板在i2c总线正在传输时复位，芯片会认为当前传输仍然在进行。对一些单板而言，直接访问i2c SCLK线是可行的，
可以
		将处理器引脚作为GPIO使用，或者另连一根引脚连到总线上。如果定义了该选项，boards/xxx/board.c中自定义
		的i2c_init_board()例程会在boot过程较早阶段执行。
		When a board is reset during an i2c bus transfer
		chips might think that the current transfer is still
		in progress. On some boards it is possible to access
		the i2c SCLK line directly, either by using the
		processor pin as a GPIO or by having a second pin
		connected to the bus. If this option is defined a
		custom i2c_init_board() routine in boards/xxx/board.c
		is run early in the boot sequence.

		CONFIG_I2CFAST (PPC405GP|PPC405EP only)
		该选项使能u-boot基于环境变量'i2cfast'配置bd_info结构体内的bi_iic_fast[]标志。(见i2cfast)
		This option enables configuration of bi_iic_fast[] flags
		in u-boot bd_info structure based on u-boot environment
		variable "i2cfast". (see also i2cfast)

- SPI Support:	CONFIG_SPI
－SPI支持

		Enables SPI driver (so far only tested with
		SPI EEPROM, also an instance works with Crystal A/D and
		D/As on the SACSng board)
		CONFIG_SPI_X

		Enables extended (16-bit) SPI EEPROM addressing.
		(symmetrical to CONFIG_I2C_X)

		CONFIG_SOFT_SPI

		Enables a software (bit-bang) SPI driver rather than
		using hardware support. This is a general purpose
		driver that only requires three general I/O port pins
		(two outputs, one input) to function. If this is
		defined, the board configuration must define several
		SPI configuration items (port pins to use, etc). For
		an example, see include/configs/sacsng.h.

- FPGA Support: CONFIG_FPGA_COUNT
－FPGA支持
		Specify the number of FPGA devices to support.

		CONFIG_FPGA

		Used to specify the types of FPGA devices.  For example,
		#define CONFIG_FPGA  CFG_XILINX_VIRTEX2

		CFG_FPGA_PROG_FEEDBACK

		Enable printing of hash marks during FPGA configuration.

		CFG_FPGA_CHECK_BUSY

		Enable checks on FPGA configuration interface busy
		status by the configuration function. This option
		will require a board or device specific function to
		be written.

		CONFIG_FPGA_DELAY

		If defined, a function that provides delays in the FPGA
		configuration driver.

		CFG_FPGA_CHECK_CTRLC
		Allow Control-C to interrupt FPGA configuration

		CFG_FPGA_CHECK_ERROR

		Check for configuration errors during FPGA bitfile
		loading. For example, abort during Virtex II
		configuration if the INIT_B line goes low (which
		indicated a CRC error).

		CFG_FPGA_WAIT_INIT

		Maximum time to wait for the INIT_B line to deassert
		after PROB_B has been deasserted during a Virtex II
		FPGA configuration sequence. The default time is 500
		mS.

		CFG_FPGA_WAIT_BUSY

		Maximum time to wait for BUSY to deassert during
		Virtex II FPGA configuration. The default is 5 mS.

		CFG_FPGA_WAIT_CONFIG

		Time to wait after FPGA configuration. The default is
		200 mS.

- Configuration Management:
－配置管理
		CONFIG_IDENT_STRING

		如果定义了，该字串会添加到uboot版本信息里面。
		If defined, this string will be added to the U-Boot
		version information (U_BOOT_VERSION)

- Vendor Parameter Protection:
```
－供应商参数保护

uboot中，供应商制造商只能设定环境变量"serial#"和"ethaddr"的值一次，以保护这些变量被user非正式的修改。一旦设定了这些变量，它们是只读的，写和删除访问都会被拒绝。当然你也是可以配置成可读写的：

```
		U-Boot considers the values of the environment
		variables "serial#" (Board Serial Number) and
		"ethaddr" (Ethernet Address) to be parameters that
		are set once by the board vendor / manufacturer, and
		protects these variables from casual modification by
		the user. Once set, these variables are read-only,
		and write or delete attempts are rejected. You can
		change this behviour:
```

如果在你的配置文件中#define CONFIG_ENV_OVERWRITE，供应商参数写保护就会被完全禁止。任何人都可以更改或者

删除这些参数。

```
		If CONFIG_ENV_OVERWRITE is #defined in your config
		file, the write protection for vendor parameters is
		completely disabled. Anybody can change or delete
		these parameters.
```

或者，如果你定义了CONFIG_OVERWRITE_ETHADDR_ONCE和CONFIG_ETHADDR，一个默认的以太网地址会在环境中安装，

它可以被user修改一次。the serial#不受影响，它依然是只读的。

```
		Alternatively, if you #define _both_ CONFIG_ETHADDR
		_and_ CONFIG_OVERWRITE_ETHADDR_ONCE, a default
		ethernet address is installed in the environment,
		which can be changed exactly ONCE by the user. [The
		serial# is unaffected by this, i. e. it remains
		read-only.]

- Protected RAM:
```
－保护ram

```
		CONFIG_PRAM
```

定义这个变量允许protected ram保留一定的内存，它不会被uboot覆盖掉。你可以通过定义一个环境变量pram设定保留的

内存大小代替默认值。注意board info结构仍然会显示整个内存的大小，如果pram定义了，一个新的环境变量mem会以一定的

格式保存剩余内存大小，它可以以启动参数的形式传递给linux。

```
		Define this variable to enable the reservation of
		"protected RAM", i. e. RAM which is not overwritten
		by U-Boot. Define CONFIG_PRAM to hold the number of
		kB you want to reserve for pRAM. You can overwrite
		this default value by defining an environment
		variable "pram" to the number of kB you want to
		reserve. Note that the board info structure will
		still show the full amount of RAM. If pRAM is
		reserved, a new environment variable "mem" will
		automatically be defined to hold the amount of
		remaining RAM in a form that can be passed as boot
		argument to Linux, for instance like that:

			setenv bootargs ... mem=\${mem}
			saveenv
```

通过这种方式你可以告诉linux不要使用这些不受重启影响的内存。

```
		This way you can tell Linux not to use this memory,
		either, which results in a memory region that will
		not be affected by reboots.
```

警告，如果你的单板配置使用了自动检测ram大小，你必须保证这些内存检测是非破坏性的。

```
		*WARNING* If your board configuration uses automatic
		detection of the RAM size, you must make sure that
		this memory test is non-destructive. So far, the
		following board configurations are known to be
		"pRAM-clean":

			ETX094, IVMS8, IVML24, SPD8xx, TQM8xxL,
			HERMES, IP860, RPXlite, LWMON, LANTEC,
			PCU_E, FLAGADM, TQM8260
```

- Error Recovery:
－错误恢复

		CONFIG_PANIC_HANG

定义这个变量，在系统遇到致命错误时候停止运作，这样你必须手动复位它。这对于你期望能自动快速的重启的嵌入式系统来说

可能不是什么好主意。但是它在开发阶段可能有用，这样你就可以试着调试导致这情况的原因。

```
		Define this variable to stop the system in case of a
		fatal error, so that you have to reset it manually.
		This is probably NOT a good idea for an embedded
		system where you want to system to reboot
		automatically as fast as possible, but it may be
		useful during development since you can try to debug
		the conditions that lead to the situation.

		CONFIG_NET_RETRY_COUNT
```

此变量定义了网络操作如ARP, RARP, TFTP, or BOOTP在放弃操作之前的重试次数。默认值是5。

```
		This variable defines the number of retries for
		network operations like ARP, RARP, TFTP, or BOOTP
		before giving up the operation. If not defined, a
		default value of 5 is used.
```

- Command Interpreter:
命令解析器

```
		CONFIG_AUTO_COMPLETE

		Enable auto completion of commands using TAB.
```

用tab键自动补全命令

注意这个功能在hush shell里面暂时没有实现。

```
		  Note that this feature has NOT been implemented yet
		  for the "hush" shell.


		CFG_HUSH_PARSER
```

定义这个变量允许hush shell作为命令行解析器，它允许强大的命令行表达式如if...then...else...fi

```
		Define this variable to enable the "hush" shell (from
		Busybox) as command line interpreter, thus enabling
		powerful command line syntax like
		if...then...else...fi conditionals or `&&' and '||'
		constructs ("shell scripts").
```

如果没有定义，你只能使用更少内存消耗的旧的简单的特性。

```
		If undefined, you get the old, much simpler behaviour
		with a somewhat smaller memory footprint.


		CFG_PROMPT_HUSH_PS2
```

这定义了第二个字串提示符，当命令解析器需要更多的输入来完成一个命令时，它会显示出来，通常是 “>”

```
		This defines the secondary prompt string, which is
		printed when the command interpreter needs more input
		to complete a command. Usually "> ".

	Note:
```

在目前实现中，局部和全局变量空间是分开的。局部变量是那些你简单的输入 `name＝value `的变量。此后要访问一个局部

变量，你可以用$name或者${name}引用它；要执行一个变量的内容，可直接在命令行输入$name。

```
		In the current implementation, the local variables
		space and global environment variables space are
```

```
                separated. Local variables are those you define by
                simply typing `name=value'. To access a local
                variable later on, you have write `$name' or
                `${name}'; to execute the contents of a variable
                directly type `$name' at the command prompt.
```

全局环境变量是那些你用setenv/printenv来工作的。要运行保存这样一个变量的命令，你需要运行命令，而且你不能用$name
的方式访问它们。

```
                Global environment variables are those you use
                setenv/printenv to work with. To run a command stored
                in such a variable, you need to use the run command,
                and you must not use the '$' sign to access them.
```

为了保存命令和特定的字符在一个变量中，请用双引号包围整个变量的文字，而不是在分号前面使用一个反斜杠或者别的符号。

```
                To store commands and special characters in a
                variable, please use double quotation marks
                surrounding the whole text of the variable, instead
                of the backslashes before semicolons and special
                symbols.

- Commandline Editing and History:
```
－命令行编辑跟历史

```
                CONFIG_CMDLINE_EDITING
```

允许编辑和历史功能，以便命令行输入操作的互动。

```
                Enable editiong and History functions for interactive
                commandline input operations

- Default Environment:
```
－默认环境

```
                CONFIG_EXTRA_ENV_SETTINGS
```

定义包含任意数量以空格终止的字符串，它将是编译进boot映像的默认环境的一部分。

```
                Define this to contain any number of null terminated
                strings (variable = value pairs) that will be part of
                the default environment compiled into the boot image.

                For example, place something like this in your
                board's config file:
```
例如，在你的board配置文件放置如下文字

```
                #define CONFIG_EXTRA_ENV_SETTINGS \
                        "myvar1=value1\0" \
                        "myvar2=value2\0"
```

警告：这个方法的前提是你了解内部环境的格式是如何存储在uboot代码中的。这不是正式的输出接口。当然这种格式不太可能
很快就改变，但是也不保证。你最好清楚在这里你所做的。

```
                Warning: This method is based on knowledge about the
                internal format how the environment is stored by the
                U-Boot code. This is NOT an official, exported
                interface! Although it is unlikely that this format
                will change soon, there is no guarantee either.
                You better know what you are doing here.
```

note：过分依赖默认的环境变量是不鼓励这样做的。确保其他方式预设的环境如自动脚本功能或者启动命令。

```
                Note: overly (ab)use of the default environment is
                discouraged. Make sure to check other ways to preset
                the environment like the autoscript function or the
                boot command first.

- DataFlash Support:
                CONFIG_HAS_DATAFLASH
```

允许dataflash功能，允许通过标准命令如cp读写dataflash。

```
                Defining this option enables DataFlash features and
                allows to read/write in Dataflash via the standard
                commands cp, md...
```

```
- SystemACE Support:
		CONFIG_SYSTEMACE

		Adding this option adds support for Xilinx SystemACE
		chips attached via some sort of local bus. The address
		of the chip must alsh be defined in the
		CFG_SYSTEMACE_BASE macro. For example:

		#define CONFIG_SYSTEMACE
		#define CFG_SYSTEMACE_BASE 0xf0000000

		When SystemACE support is added, the "ace" device type
		becomes available to the fat commands, i.e. fatls.

- TFTP Fixed UDP Port:
－TFTP固定UDP端口

		CONFIG_TFTP_PORT

		如果定义了，环境变量tftpsrcp用来提供TFTP UDP源端口的值，如果tftpsrcp没有定义，则正常的随机端口号将被使用。
		If this is defined, the environment variable tftpsrcp
		is used to supply the TFTP UDP source port value.
		If tftpsrcp isn't defined, the normal pseudo-random port
		number generator is used.

		当然，环境变量tftpdstp用来提供TFTP UDP目标端口值，如果tftpdstp没有定义，通常使用69号端口。
		Also, the environment variable tftpdstp is used to supply
		the TFTP UDP destination port value.  If tftpdstp isn't
		defined, the normal port 69 is used.

		tftpsrcp的目的是允许TFTP服务端使用预先配置好的目标IP地址和UDP端口开始TFTP传输。这个穿透影响了防火墙，允许其他

		的TFTP传输和进程正常进出。一个比较好的解决方法是正确配置防火墙，但是这有时候又是不允许的。
		The purpose for tftpsrcp is to allow a TFTP server to
		blindly start the TFTP transfer using the pre-configured
		target IP address and UDP port. This has the effect of
		"punching through" the (Windows XP) firewall, allowing
		the remainder of the TFTP transfer to proceed normally.
		A better solution is to properly configure the firewall,
		but sometimes that is not allowed.

- Show boot progress:
－显示引导进度

		CONFIG_SHOW_BOOT_PROGRESS

		定义这个选项，允许添加一些特定的板极代码（调用一个用户提供的名为"show_boot_progress(int)"的函数）使用你

		的board上面的显示器显示系统启动进度，目前，下面的检查点将实施。
		Defining this option allows to add some board-
		specific code (calling a user-provided function
		"show_boot_progress(int)") that enables you to show
		the system's boot progress on some display (for
		example, some LED's) on your board. At the moment,
		the following checkpoints are implemented:

  Arg	Where			When
    1	common/cmd_bootm.c	before attempting to boot an image
   -1	common/cmd_bootm.c	Image header has bad	 magic number
    2	common/cmd_bootm.c	Image header has correct magic number
   -2	common/cmd_bootm.c	Image header has bad	 checksum
    3	common/cmd_bootm.c	Image header has correct checksum
   -3	common/cmd_bootm.c	Image data	 has bad	 checksum
    4	common/cmd_bootm.c	Image data	 has correct checksum
   -4	common/cmd_bootm.c	Image is for unsupported architecture
    5	common/cmd_bootm.c	Architecture check OK
   -5	common/cmd_bootm.c	Wrong Image Type (not kernel, multi, standalone)
    6	common/cmd_bootm.c	Image Type check OK
   -6	common/cmd_bootm.c	gunzip uncompression error
   -7  common/cmd_bootm.c  Unimplemented compression type
    7  common/cmd_bootm.c  Uncompression OK
   -8  common/cmd_bootm.c  Wrong Image Type (not kernel, multi, standalone)
    8  common/cmd_bootm.c  Image Type check OK
   -9  common/cmd_bootm.c  Unsupported OS (not Linux, BSD, VxWorks, QNX)
    9  common/cmd_bootm.c  Start initial ramdisk verification
  -10  common/cmd_bootm.c  Ramdisk header has bad        magic number
  -11  common/cmd_bootm.c  Ramdisk header has bad        checksum
   10  common/cmd_bootm.c  Ramdisk header is OK
  -12  common/cmd_bootm.c  Ramdisk data   has bad        checksum
   11  common/cmd_bootm.c  Ramdisk data   has correct checksum
   12  common/cmd_bootm.c  Ramdisk verification complete, start loading
  -13  common/cmd_bootm.c  Wrong Image Type (not PPC Linux Ramdisk)
   13  common/cmd_bootm.c  Start multifile image verification
   14  common/cmd_bootm.c  No initial ramdisk, no multifile, continue.
   15  common/cmd_bootm.c  All preparation done, transferring control to OS

  -30  lib_ppc/board.c             Fatal error, hang the system
  -31  post/post.c         POST test failed, detected by post_output_backlog()
  -32  post/post.c         POST test failed, detected by post_run_single()

   -1  common/cmd_doc.c    Bad usage of "doc" command
   -1  common/cmd_doc.c    No boot device
   -1  common/cmd_doc.c    Unknown Chip ID on boot device
   -1  common/cmd_doc.c    Read Error on boot device
   -1  common/cmd_doc.c    Image header has bad magic number

   -1  common/cmd_ide.c    Bad usage of "ide" command
   -1  common/cmd_ide.c    No boot device
   -1  common/cmd_ide.c    Unknown boot device
   -1  common/cmd_ide.c    Unknown partition table
   -1  common/cmd_ide.c    Invalid partition type
   -1  common/cmd_ide.c    Read Error on boot device
   -1  common/cmd_ide.c    Image header has bad magic number

   -1  common/cmd_nand.c   Bad usage of "nand" command
   -1  common/cmd_nand.c   No boot device
   -1  common/cmd_nand.c   Unknown Chip ID on boot device
   -1  common/cmd_nand.c   Read Error on boot device
   -1  common/cmd_nand.c   Image header has bad magic number

   -1  common/env_common.c Environment has a bad CRC, using default


Modem Support:
--------------

目前只支持SMDK2400和TRAB的板
[so far only for SMDK2400 and TRAB boards]

- Modem support endable:
              CONFIG_MODEM_SUPPORT

- RTS/CTS Flow control enable:
流控制
              CONFIG_HWFLOW

- Modem debug support:
调试支持
              CONFIG_MODEM_SUPPORT_DEBUG

              Enables debugging stuff (char screen[1024], dbg())
              for modem support. Useful only with BDI2000.

- Interrupt support (PPC):
中断支持


              There are common interrupt_init() and timer_interrupt()
              for all PPC archs. interrupt_init() calls interrupt_init_cpu()
              for cpu specific initialization. interrupt_init_cpu()
              should set decrementer_count to appropriate value. If
		cpu resets decrementer automatically after interrupt
		(ppc4xx) it should set decrementer_count to zero.
		timer_interrupt() calls timer_interrupt_cpu() for cpu
		specific handling. If board has watchdog / status_led
		/ other_activity_monitor it works automatically from
		general timer_interrupt().

- General:

		In the target system modem support is enabled when a
		specific key (key combination) is pressed during
		power-on. Otherwise U-Boot will boot normally
		(autoboot). The key_pressed() fuction is called from
		board_init(). Currently key_pressed() is a dummy
		function, returning 1 and thus enabling modem
		initialization.

		If there are no modem init strings in the
		environment, U-Boot proceed to autoboot; the
		previous output (banner, info printfs) will be
		supressed, though.

		See also: doc/README.Modem


Configuration Settings:
-----------------------
配置设定

- CFG_LONGHELP: Defined when you want long help messages included;
		undefine this when you're short of memory.
		当你需要详细的help信息时候定义它；当你内存有限时候最好不定义。
- CFG_PROMPT:	This is what U-Boot prints on the console to
		prompt for user input.
		终端命令提示符。

- CFG_CBSIZE:	Buffer size for input from the Console
					控制台输入的buffer容量
- CFG_PBSIZE:	Buffer size for Console output
					控制台输出的buffer容量
- CFG_MAXARGS:	max. Number of arguments accepted for monitor commands
					monitor命令最大接受参数个数
- CFG_BARGSIZE: Buffer size for Boot Arguments which are passed to
		the application (usually a Linux kernel) when it is
		booted
					传递给应用程序（通常是linux kernel）的启动参数的buffer 大小
- CFG_BAUDRATE_TABLE:
		List of legal baudrate settings for this board.
		合法的波特率列表
- CFG_CONSOLE_INFO_QUIET
		Suppress display of console information at boot.
		启动时候禁止控制台信息
- CFG_CONSOLE_IS_IN_ENV
		If the board specific function
			extern int overwrite_console (void);
		returns 1, the stdin, stderr and stdout are switched to the
		serial port, else the settings in the environment are used.
		如果board定义了这个函数extern int overwrite_console (void);返回1，那么stdin，stderr，stdout都会
切换到
		串行端口，否则的话环境的设定将会用到。

- CFG_CONSOLE_OVERWRITE_ROUTINE
		Enable the call to overwrite_console().
		允许调用overwrite_console().

- CFG_CONSOLE_ENV_OVERWRITE
		Enable overwrite of previous console environment settings.
		允许覆盖以前的控制台环境设定
- CFG_MEMTEST_START, CFG_MEMTEST_END:
		Begin and End addresses of the area used by the
		simple memory test.
```

```
              简单内存测试的开始和结束地址。
- CFG_ALT_MEMTEST:
              Enable an alternate, more extensive memory test.
              允许一个备用的更深度的内存测试。
- CFG_MEMTEST_SCRATCH:
              Scratch address used by the alternate memory test
              You only need to set this if address zero isn't writeable
              构建一个用来代替内存测试的地址，你仅在地址0不可写入的情况下才需要设置
- CFG_TFTP_LOADADDR:
              Default load address for network file downloads
              默认网络文件下载的加载地址
- CFG_LOADS_BAUD_CHANGE:
              Enable temporary baudrate change while serial download
              允许一个临时的波特率当需要串口下载时候
- CFG_SDRAM_BASE:
              Physical start address of SDRAM. _Must_ be 0 here.
              SDRAM物理开始地址，必须是0？？
- CFG_MBIO_BASE:
              Physical start address of Motherboard I/O (if using a
              Cogent motherboard)
              主板IO物理开始地址，如果使用cogent的主板的话
- CFG_FLASH_BASE:
              Physical start address of Flash memory.
              Flash内存开始的物理地址。
- CFG_MONITOR_BASE:
              Physical start address of boot monitor code (set by
              make config files to be same as the text base address
              (TEXT_BASE) used when linking) - same as
              CFG_FLASH_BASE when booting from flash.
              boot minitor的物理开始地址（make config文件设置，和linking时候用的TEXT_BASE一样），如果在flash启
动，则和
              CFG_FLASH_BASE一样。
- CFG_MONITOR_LEN:
              Size of memory reserved for monitor code, used to
              determine _at_compile_time_ (!) if the environment is
              embedded within the U-Boot image, or in a separate
              flash sector.
              monitor代码保留的内存大小，在编译阶段决定，如果环境是嵌入在uboot映像或者独立的flash扇区。
- CFG_MALLOC_LEN:
              Size of DRAM reserved for malloc() use.
              保留给malloc函数使用的DRAM大小。
- CFG_BOOTM_LEN:
              Normally compressed uImages are limited to an
              uncompressed size of 8 MBytes. If this is not enough,
              you can define CFG_BOOTM_LEN in your board config file
              to adjust this setting to your needs.
              正常压缩后的uImages解压后大小限制在8MB以下。如果不够，你可以在你的board配置文件配置CFG_BOOTM_LEN以适
应你的
              需要。
- CFG_BOOTMAPSZ:
              Maximum size of memory mapped by the startup code of
              the Linux kernel; all data that must be processed by
              the Linux kernel (bd_info, boot arguments, eventually
              initrd image) must be put below this limit.
              linux kernel启动代码在内存分布的最大容量。linux kernel必须处理的所有数据(bd_info,boot
arguments,initrd
              image)都在这个限制之下。
- CFG_MAX_FLASH_BANKS:
              Max number of Flash memory banks
              flash 最大banks
- CFG_MAX_FLASH_SECT:
              Max number of sectors on a Flash chip
              flash最大扇区数
- CFG_FLASH_ERASE_TOUT:
              Timeout for Flash erase operations (in ms)
              flash擦除操作超时时间，单位ms
- CFG_FLASH_WRITE_TOUT:
              Timeout for Flash write operations (in ms)
              flash写操作超时时间，单位ms
- CFG_FLASH_LOCK_TOUT
              Timeout for Flash set sector lock bit operation (in ms)
```

```
- CFG_FLASH_UNLOCK_TOUT
		Timeout for Flash clear lock bits operation (in ms)
		flash清除上锁位的超时，单位ms
- CFG_FLASH_PROTECTION
		If defined, hardware flash sectors protection is used
		instead of U-Boot software protection.
		如果定义了，硬件的flash扇区保护将被使用，而不是uboot的软件保护。
- CFG_DIRECT_FLASH_TFTP:

		Enable TFTP transfers directly to flash memory;
		without this option such a download has to be
		performed in two steps: (1) download to RAM, and (2)
		copy from RAM to flash.
		允许TFTP传输直接存储到flash内存；没有选中这个选项时，一个下载通常分两个步骤：（1）下载到RAM(2)从RAM拷
贝到flash

		The two-step approach is usually more reliable, since
		you can check if the download worked before you erase
		the flash, but in some situations (when sytem RAM is
		too limited to allow for a tempory copy of the
		downloaded image) this option may be very useful.
		分两步的下载方式通常更可靠一些，因为你可以在擦除flash之前检查下载是否正常。
- CFG_FLASH_CFI:
		Define if the flash driver uses extra elements in the
		common flash structure for storing flash geometry.
		定义flash驱动是否使用为存储flash矩阵设计的通用flash接口的额外特性
- CFG_FLASH_CFI_DRIVER
		This option also enables the building of the cfi_flash driver
		in the drivers directory
		这个选项允许在驱动目录构建cfi_flash驱动
- CFG_FLASH_QUIET_TEST
		If this option is defined, the common CFI flash doesn't
		print it's warning upon not recognized FLASH banks. This
		is useful, if some of the configured banks are only
		optionally available.
		如果定义了这个选项，通用的CFI flash不会打印它未识别的flash banks。如果某些配置的banks以可选的方式存
在的话
		这可能有用。
- CFG_RX_ETH_BUFFER:
		Defines the number of ethernet receive buffers. On some
		ethernet controllers it is recommended to set this value
		to 8 or even higher (EEPRO100 or 405 EMAC), since all
		buffers can be full shortly after enabling the interface
		on high ethernet traffic.
		Defaults to 4 if not defined.
		定义以太网接收buffers的数量。此值在某些以太网控制器默认设置是8或者更高(EEPRO100或者405EMAC)，因为在高
速的以太网
		传输中，buffers可能很快就满了。

The following definitions that deal with the placement and management
of environment data (variable area); in general, we support the
following configurations:
接下来的定义主要处理环境变量数据的布局管理；通常我们支持以下配置：
- CFG_ENV_IS_IN_FLASH:

	Define this if the environment is in flash memory.
	如果环境存储在flash内存的话，定义这个配置选项。
	a) The environment occupies one whole flash sector, which is
	   "embedded" in the text segment with the U-Boot code. This
	   happens usually with "bottom boot sector" or "top boot
	   sector" type flash chips, which have several smaller
	   sectors at the start or the end. For instance, such a
	   layout can have sector sizes of 8, 2x4, 16, Nx32 kB. In
	   such a case you would place the environment in one of the
	   4 kB sectors - with U-Boot code before and after it. With
	   "top boot sector" type flash chips, you would put the
	   environment in one of the last sectors, leaving a gap
	   between U-Boot and the environment.
		环境占用了整个flash扇区，它嵌入在uboot代码的代码段中。这通常发生在带顶部启动扇区或者底部启动扇区类型的
flash芯片
```

```
		，它的顶部和底部扇区通常都比较小。比如，布局可以是8，2X4，16，NX32KB的扇区大小。这样你可以把环境变量放
在一个4KB
		大小的扇区，uboot代码在它的前面或者它的后面。在顶部启动扇区类型的flash芯片中，你可以把环境变量放在最后
一个扇区，
		它和uboot代码段之间有一个间隔。

	- CFG_ENV_OFFSET:

	  Offset of environment data (variable area) to the
	  beginning of flash memory; for instance, with bottom boot
	  type flash chips the second sector can be used: the offset
	  for this sector is given here.
		环境变量数据跟flash内存的开始端之间的偏移量；比如，底部启动扇区类型的flash芯片的第二个扇区可以用做这里
给出的偏移
	  CFG_ENV_OFFSET is used relative to CFG_FLASH_BASE.
		CFG_ENV_OFFSET是与CFG_FLASH_BASE相关联的。

	- CFG_ENV_ADDR:

	  This is just another way to specify the start address of
	  the flash sector containing the environment (instead of
	  CFG_ENV_OFFSET).
		这是另外一个方式代替CFG_ENV_OFFSET定义flash扇区保存环境变量的开始地址。

	- CFG_ENV_SECT_SIZE:

	  Size of the sector containing the environment.
		保存环境变量的扇区大小。

	b) Sometimes flash chips have few, equal sized, BIG sectors.
	  In such a case you don't want to spend a whole sector for
	  the environment.
		有时flash芯片只有少量的，相等大小，很大的扇区，这样的话你就不会为环境变量划分一整个扇区的大小了。

	- CFG_ENV_SIZE:

	  If you use this in combination with CFG_ENV_IS_IN_FLASH
	  and CFG_ENV_SECT_SIZE, you can specify to use only a part
	  of this flash sector for the environment. This saves
	  memory for the RAM copy of the environment.
		如果你同时配置了CFG_ENV_IS_IN_FLASH和CFG_ENV_SECT_SIZE,你可以定义只使用这个flash扇区的一部分保存
环境变量
		。这节省了环境变量拷贝到RAM时候的内存消耗。

	  It may also save flash memory if you decide to use this
	  when your environment is "embedded" within U-Boot code,
	  since then the remainder of the flash sector could be used
	  for U-Boot code. It should be pointed out that this is
	  STRONGLY DISCOURAGED from a robustness point of view:
	  updating the environment in flash makes it always
	  necessary to erase the WHOLE sector. If something goes
	  wrong before the contents has been restored from a copy in
	  RAM, your target system will be dead.
		当你的环境变量是嵌入到uboot代码段的时候，使用这个配置可能节省了flash，因为flash扇区的剩余部分可以给
uboot代码使
		用。当然从鲁棒性的观点出发，这也是有很大的缺点的：更新flash里面的环境变量使得必须擦除整个扇区。如果在
RAM里面的拷贝
		恢复内容之前发生错误，你的目标系统可能会当机。

	- CFG_ENV_ADDR_REDUND
	  CFG_ENV_SIZE_REDUND

	  These settings describe a second storage area used to hold
	  a redundand copy of the environment data, so that there is
	  a valid backup copy in case there is a power failure during
	  a "saveenv" operation.
		这两个设定描述了用来保存环境变量拷贝的第二个存储区域，这样当saveenv操作的时候如果发生掉电时候可以有一份
有效
		的备份。

BE CAREFUL! Any changes to the flash layout, and some changes to the
```

```
source code will make it necessary to adapt <board>/u-boot.lds*
accordingly!
              注意！任何对flash布局以及源代码的修改都有必要重新相应的调整 <board>/u-boot.lds。

- CFG_ENV_IS_IN_NVRAM:

       Define this if you have some non-volatile memory device
       (NVRAM, battery buffered SRAM) which you want to use for the
       environment.
       如果你使用某些非易失的内存设备来保存环境变量，定义这个选项。

       - CFG_ENV_ADDR:
       - CFG_ENV_SIZE:

         These two #defines are used to determin the memory area you
         want to use for environment. It is assumed that this memory
         can just be read and written to, without any special
         provision.
              这两个选项用来决定你保存环境变量的内存区域。假定这种内存可以不用其他设置就可以读写。

BE CAREFUL! The first access to the environment happens quite early
in U-Boot initalization (when we try to get the setting of for the
console baudrate). You *MUST* have mappend your NVRAM area then, or
U-Boot will hang.
注意！第一次环境变量的访问发生在uboot初始化的早期（当我们试着得到控制台波特率的设置的时候）。你必须先安排好NVRAM的布局
，不然uboot会当机的。

Please note that even with NVRAM we still use a copy of the
environment in RAM: we could work on NVRAM directly, but we want to
keep settings there always unmodified except somebody uses "saveenv"
to save the current settings.
请注意，虽然使用的是NVRAM，我们仍然使用环境变量在RAM中的拷贝：当然我们也可以直接用NVRAM，但我们想要保存没有修改的设定，
除非某人用saveenv命令保存当前设定。

- CFG_ENV_IS_IN_EEPROM:

       Use this if you have an EEPROM or similar serial access
       device and a driver for it.
       定义这个选项，如果你有一个EEPROM或者类似的串行访问设备并且有它的驱动。

       - CFG_ENV_OFFSET:
       - CFG_ENV_SIZE:

         These two #defines specify the offset and size of the
         environment area within the total memory of your EEPROM.
              这两个选项定义了你的EEPROM的全部内存中环境变量的偏移量和大小。

       - CFG_I2C_EEPROM_ADDR:
         If defined, specified the chip address of the EEPROM device.
         The default address is zero.
              定义EEPROM设备的芯片地址，默认是0。

       - CFG_EEPROM_PAGE_WRITE_BITS:
         If defined, the number of bits used to address bytes in a
         single page in the EEPROM device.  A 64 byte page, for example
         would require six bits.
              定义EEPROM设备的一页中要用地址字节中多少位。一个64字节的页面，一般要求6位来表示。

       - CFG_EEPROM_PAGE_WRITE_DELAY_MS:
         If defined, the number of milliseconds to delay between
         page writes.       The default is zero milliseconds.
              页面写入之间的延时，默认是0ms。

       - CFG_I2C_EEPROM_ADDR_LEN:
         The length in bytes of the EEPROM memory array address.  Note
         that this is NOT the chip address length!
              定义EEPROM内存阵列地址的字节长度。注意这不是芯片地址长度。

       - CFG_I2C_EEPROM_ADDR_OVERFLOW:
         EEPROM chips that implement "address overflow" are ones
         like Catalyst 24WC04/08/16 which has 9/10/11 bits of
```

```
		address and the extra bits end up in the "chip address" bit
		slots. This makes a 24WC08 (1Kbyte) chip look like four 256
		byte chips.
		     EEPROM实现了地址溢出的芯片，它有9/10/11位的地址，额外的地址位占用了芯片地址位，这使得一片24WC08（1k字节）看起来
		     好像是有4片256字节的芯片。

		Note that we consider the length of the address field to
		still be one byte because the extra address bits are hidden
		in the chip address.
		     注意，我们仍然认为地址的长度是一个字节的，因为额外的地址位是隐藏在芯片地址中的。

	      - CFG_EEPROM_SIZE:
		The size in bytes of the EEPROM device.
		     定义EEPROM设备的字节容量

- CFG_ENV_IS_IN_DATAFLASH:

	Define this if you have a DataFlash memory device which you
	want to use for the environment.
	如果你有dataflash内存设备用来保存环境变量的话，定义此项。

	- CFG_ENV_OFFSET:
	- CFG_ENV_ADDR:
	- CFG_ENV_SIZE:

	  These three #defines specify the offset and size of the
	  environment area within the total memory of your DataFlash placed
	  at the specified address.
	     这三个选项定义了在你的dataflash总内存中的特定地址环境变量的偏移量和大小。

- CFG_ENV_IS_IN_NAND:

	Define this if you have a NAND device which you want to use
	for the environment.
	如果你是用NAND闪存来保存环境变量的话，定义此项。

	- CFG_ENV_OFFSET:
	- CFG_ENV_SIZE:

	  These two #defines specify the offset and size of the environment
	  area within the first NAND device.
	     这两项定义了环境变量在第一个NAND设备上的偏移量和大小。

	- CFG_ENV_OFFSET_REDUND

	  This setting describes a second storage area of CFG_ENV_SIZE
	  size used to hold a redundant copy of the environment data,
	  so that there is a valid backup copy in case there is a
	  power failure during a "saveenv" operation.
	     这个设定描述了用来保存环境变量拷贝的第二个存储区域，这样当saveenv操作的时候如果发生掉电时候可以有一份有效
	     的备份。
	Note: CFG_ENV_OFFSET and CFG_ENV_OFFSET_REDUND must be aligned
	to a block boundary, and CFG_ENV_SIZE must be a multiple of
	the NAND devices block size.
	注意：CFG_ENV_OFFSET和CFG_ENV_OFFSET_REDUND必须对齐到一个block的边界， 而且CFG_ENV_SIZE必须是NAND设备块
	大小的倍数。

- CFG_SPI_INIT_OFFSET

	Defines offset to the initial SPI buffer area in DPRAM. The
	area is used at an early stage (ROM part) if the environment
	is configured to reside in the SPI EEPROM: We need a 520 byte
	scratch DPRAM area. It is used between the two initialization
	calls (spi_init_f() and spi_init_r()). A value of 0xB00 seems
	to be a good choice since it makes it far enough from the
	start of the data area as well as from the stack pointer.
	定义DPRAM初始化SPI缓冲区域的偏移量。这个区域是用在如果环境变量配置成位于SPI EEPROM的早期阶段：我们需要DPRAM区域
	的520字节空间。它用在两个初始化调用之间(spi_init_f() and spi_init_r())。值0XB00看起来是个很好的选择，因为它
```

```
离
       数据区域和堆栈指针都很远。

Please note that the environment is read-only as long as the monitor
has been relocated to RAM and a RAM copy of the environment has been
created; also, when using EEPROM you will have to use getenv_r()
until then to read environment variables.
请注意，环境变量直到monitor已经重定位到RAM并且RAM有一份环境变量的拷贝还是只读的。当然，使用EEPROM时候你可以使用use
getenv_r()直到读出环境变量为止。

The environment is protected by a CRC32 checksum. Before the monitor
is relocated into RAM, as a result of a bad CRC you will be working
with the compiled-in default environment - *silently*!!! [This is
necessary, because the first environment variable we need is the
"baudrate" setting for the console - if we have a bad CRC, we don't
have any device yet where we could complain.]
环境变量受CRC32校验和的保护。在monitor重定位到RAM中之前，坏的CRC的结果是你将使用编译时候的默认环境。这是必须的，因为
我们首先需要设置控制台的第一个环境变量就是baudrate，如果CRC出错，我们没有任何设备可以报告了。

Note: once the monitor has been relocated, then it will complain if
the default environment is used; a new CRC is computed as soon as you
use the "saveenv" command to store a valid environment.
注意：一旦monitor重定位后，如果使用默认的环境变量它会抱怨的。新的CRC会在使用saveenv命令保存有效的环境变量之后重新计
算。

- CFG_FAULT_ECHO_LINK_DOWN:
              Echo the inverted Ethernet link state to the fault LED.
              在故障LED显示错误的以太网连接。

              Note: If this option is active, then CFG_FAULT_MII_ADDR
                    also needs to be defined.
                            如果激活这个选项，CFG_FAULT_MII_ADDR也需要定义。

- CFG_FAULT_MII_ADDR:
              MII address of the PHY to check for the Ethernet link state.
              PHY的MII地址用来检查以太网连接的状态。

- CFG_64BIT_VSPRINTF:
              Makes vsprintf (and all *printf functions) support printing
              of 64bit values by using the L quantifier

- CFG_64BIT_STRTOUL:
              Adds simple_strtoull that returns a 64bit value

Low Level (hardware related) configuration options:
---------------------------------------------------
底层硬件相关的配置选项

- CFG_CACHELINE_SIZE:
              Cache Line Size of the CPU.

- CFG_DEFAULT_IMMR:
              Default address of the IMMR after system reset.
              系统复位后IMMR默认地址

              Needed on some 8260 systems (MPC8260ADS, PQ2FADS-ZU,
              and RPXsuper) to be able to adjust the position of
              the IMMR register after a reset.
              在一些8260系统，复位后以适应IMMR寄存器的位置。

- Floppy Disk Support:
一软驱支持

              CFG_FDC_DRIVE_NUMBER

              the default drive number (default value 0)
              默认驱动器编号

              CFG_ISA_IO_STRIDE

              defines the spacing between fdc chipset registers
```

```
		(default value 1)
		定义fdc芯片寄存器之间的间隔

		CFG_ISA_IO_OFFSET

		defines the offset of register from address. It
		depends on which part of the data bus is connected to
		the fdc chipset. (default value 0)
		定义IO地址寄存器的偏移量。它取决于那些数据总线连接到fdc芯片。

		If CFG_ISA_IO_STRIDE CFG_ISA_IO_OFFSET and
		CFG_FDC_DRIVE_NUMBER are undefined, they take their
		default value.
		如果CFG_ISA_IO_STRIDE CFG_ISA_IO_OFFSET和CFG_FDC_DRIVE_NUMBER没有定义，取默认值。

		if CFG_FDC_HW_INIT is defined, then the function
		fdc_hw_init() is called at the beginning of the FDC
		setup. fdc_hw_init() must be provided by the board
		source code. It is used to make hardware dependant
		initializations.
		如果 CFG_FDC_HW_INIT定义了，FDC setup的时候会调用fdc_hw_init()函数，这个函数必须由board源代码提
供。它提供
		硬件相关的初始化。

- CFG_IMMR:	Physical address of the Internal Memory.
		DO NOT CHANGE unless you know exactly what you're
		doing! (11-4) [MPC8xx/82xx systems only]
		内部存储器的物理地址，除非你清楚否则不要更改。

- CFG_INIT_RAM_ADDR:

		Start address of memory area that can be used for
		initial data and stack; please note that this must be
		writable memory that is working WITHOUT special
		initialization, i. e. you CANNOT use normal RAM which
		will become available only after programming the
		memory controller and running certain initialization
		sequences.
		内存区域的开始地址可以用来初始化数据和堆栈。注意这个区域必须是不用任何特殊的初始化就可以写入的内存，例如
你不能用的是
		要对内存控制器编程以及要运行特定的初始化顺序的普通RAM来定义是INIT_RAM。

		U-Boot uses the following memory types:
		- MPC8xx and MPC8260: IMMR (internal memory of the CPU)
		- MPC824X: data cache
		- PPC4xx:  data cache

- CFG_GBL_DATA_OFFSET:

		Offset of the initial data structure in the memory
		area defined by CFG_INIT_RAM_ADDR. Usually
		CFG_GBL_DATA_OFFSET is chosen such that the initial
		data is located at the end of the available space
		(sometimes written as (CFG_INIT_RAM_END -
		CFG_INIT_DATA_SIZE), and the initial stack is just
		below that area (growing from (CFG_INIT_RAM_ADDR +
		CFG_GBL_DATA_OFFSET) downward.
		在CFG_INIT_RAM_ADDR定义的初始化数据结构的内存区域的偏移量。CFG_GBL_DATA_OFFSET通常选择那些位于有效
空间尾端
		的初始化数据。初始化堆栈位于这个区域的下面。

	Note:
		On the MPC824X (or other systems that use the data
		cache for initial memory) the address chosen for
		CFG_INIT_RAM_ADDR is basically arbitrary - it must
		point to an otherwise UNUSED address space between
		the top of RAM and the start of the PCI space.
		注意：在MPC824X或者其他使用data cache作初始化内存的系统中，CFG_INIT_RAM_ADDR地址的选择可以任意，它
必须指向
		在RAM顶部和PCI空间开始之间的未用到的地址空间。
```

```
- CFG_SIUMCR: SIU Module Configuration (11-6)

- CFG_SYPCR:  System Protection Control (11-9)

- CFG_TBSCR:  Time Base Status and Control (11-26)

- CFG_PISCR:  Periodic Interrupt Status and Control (11-31)

- CFG_PLPRCR: PLL, Low-Power, and Reset Control Register (15-30)

- CFG_SCCR:   System Clock and reset Control Register (15-27)

- CFG_OR_TIMING_SDRAM:
              SDRAM timing

- CFG_MAMR_PTA:
              periodic timer for refresh

- CFG_DER:    Debug Event Register (37-47)

- FLASH_BASE0_PRELIM, FLASH_BASE1_PRELIM, CFG_REMAP_OR_AM,
  CFG_PRELIM_OR_AM, CFG_OR_TIMING_FLASH, CFG_OR0_REMAP,
  CFG_OR0_PRELIM, CFG_BR0_PRELIM, CFG_OR1_REMAP, CFG_OR1_PRELIM,
  CFG_BR1_PRELIM:
              Memory Controller Definitions: BR0/1 and OR0/1 (FLASH)

- SDRAM_BASE2_PRELIM, SDRAM_BASE3_PRELIM, SDRAM_MAX_SIZE,
  CFG_OR_TIMING_SDRAM, CFG_OR2_PRELIM, CFG_BR2_PRELIM,
  CFG_OR3_PRELIM, CFG_BR3_PRELIM:
              Memory Controller Definitions: BR2/3 and OR2/3 (SDRAM)

- CFG_MAMR_PTA, CFG_MPTPR_2BK_4K, CFG_MPTPR_1BK_4K, CFG_MPTPR_2BK_8K,
  CFG_MPTPR_1BK_8K, CFG_MAMR_8COL, CFG_MAMR_9COL:
              Machine Mode Register and Memory Periodic Timer
              Prescaler definitions (SDRAM timing)

- CFG_I2C_UCODE_PATCH, CFG_I2C_DPMEM_OFFSET [0x1FC0]:
              enable I2C microcode relocation patch (MPC8xx);
              define relocation offset in DPRAM [DSP2]

- CFG_SPI_UCODE_PATCH, CFG_SPI_DPMEM_OFFSET [0x1FC0]:
              enable SPI microcode relocation patch (MPC8xx);
              define relocation offset in DPRAM [SCC4]

- CFG_USE_OSCCLK:
              Use OSCM clock mode on MBX8xx board. Be careful,
              wrong setting might damage your board. Read
              doc/README.MBX before setting this variable!

- CFG_CPM_POST_WORD_ADDR: (MPC8xx, MPC8260 only)
              Offset of the bootmode word in DPRAM used by post
              (Power On Self Tests). This definition overrides
              #define'd default value in commproc.h resp.
              cpm_8260.h.

- CFG_PCI_SLV_MEM_LOCAL, CFG_PCI_SLV_MEM_BUS, CFG_PICMR0_MASK_ATTRIB,
  CFG_PCI_MSTR0_LOCAL, CFG_PCIMSK0_MASK, CFG_PCI_MSTR1_LOCAL,
  CFG_PCIMSK1_MASK, CFG_PCI_MSTR_MEM_LOCAL, CFG_PCI_MSTR_MEM_BUS,
  CFG_CPU_PCI_MEM_START, CFG_PCI_MSTR_MEM_SIZE, CFG_POCMR0_MASK_ATTRIB,
  CFG_PCI_MSTR_MEMIO_LOCAL, CFG_PCI_MSTR_MEMIO_BUS, CPU_PCI_MEMIO_START,
  CFG_PCI_MSTR_MEMIO_SIZE, CFG_POCMR1_MASK_ATTRIB, CFG_PCI_MSTR_IO_LOCAL,
  CFG_PCI_MSTR_IO_BUS, CFG_CPU_PCI_IO_START, CFG_PCI_MSTR_IO_SIZE,
  CFG_POCMR2_MASK_ATTRIB: (MPC826x only)
              Overrides the default PCI memory map in cpu/mpc8260/pci.c if set.

- CONFIG_ETHER_ON_FEC[12]
              Define to enable FEC[12] on a 8xx series processor.

- CONFIG_FEC[12]_PHY
              Define to the hardcoded PHY address which corresponds
              to the given FEC; i. e.
                     #define CONFIG_FEC1_PHY 4
		means that the PHY with address 4 is connected to FEC1

		When set to -1, means to probe for first available.

- CONFIG_FEC[12]_PHY_NORXERR
		The PHY does not have a RXERR line (RMII only).
		(so program the FEC to ignore it).

- CONFIG_RMII
		Enable RMII mode for all FECs.
		Note that this is a global option, we can't
		have one FEC in standard MII mode and another in RMII mode.

- CONFIG_CRC32_VERIFY
		Add a verify option to the crc32 command.
		The syntax is:
		添加一个验证选项到crc32命令
		=> crc32 -v <address> <count> <crc32>

		Where address/count indicate a memory area
		and crc32 is the correct crc32 which the
		area should have.
		上面address/count指示内存区域，crc32是修正crc32应该在那个区域。

- CONFIG_LOOPW
		Add the "loopw" memory command. This only takes effect if
		the memory commands are activated globally (CFG_CMD_MEM).
		添加loopw内存命令。这只是如果定义了全局变量CFG_CMD_MEM才有作用。

- CONFIG_MX_CYCLIC
		Add the "mdc" and "mwc" memory commands. These are cyclic
		"md/mw" commands.
		Examples:
		添加mdc和mwc内存命令。这是周期性的md，mw命令。
		=> mdc.b 10 4 500
		This command will print 4 bytes (10,11,12,13) each 500 ms.
		这个命令会每500ms一个周期打印4字节数据。

		=> mwc.l 100 12345678 10
		This command will write 12345678 to address 100 all 10 ms.
		这个命令在10ms内对地址100写入12345678
		This only takes effect if the memory commands are activated
		globally (CFG_CMD_MEM).
		这只是如果定义了全局变量CFG_CMD_MEM才有作用。

- CONFIG_SKIP_LOWLEVEL_INIT
- CONFIG_SKIP_RELOCATE_UBOOT

		[ARM only] If these variables are defined, then
		certain low level initializations (like setting up
		the memory controller) are omitted and/or U-Boot does
		not relocate itself into RAM.
		Normally these variables MUST NOT be defined. The
		only exception is when U-Boot is loaded (to RAM) by
		some other boot loader or by a debugger which
		performs these intializations itself.
		ARM适用。如果定义了这些变量，某些底层的初始化比如设置内存控制器会跳过或者uboot不会把自身重定位到RAM。通
常这两个
		变量不应该设置，除非当uboot是被其他bootloader加载到RAM或者某些调试器自己执行了这些初始化。


Building the Software:
======================
构建uboot

Building U-Boot has been tested in native PPC environments (on a
PowerBook G3 running LinuxPPC 2000) and in cross environments
(running RedHat 6.x and 7.x Linux on x86, Solaris 2.6 on a SPARC, and
NetBSD 1.5 on x86).
构建uboot已经在本地的ppc环境以及在交叉环境中测试过了。
```

If you are not using a native PPC environment, it is assumed that you
have the GNU cross compiling tools available in your path and named
with a prefix of "powerpc-linux-". If this is not the case, (e.g. if
you are using Monta Vista's Hard Hat Linux CDK 1.2) you must change
the definition of CROSS_COMPILE in Makefile. For HHL on a 4xx CPU,
change it to:
如果你不是用本地的ppc环境，假设你已经在你的路径中安装了一个前缀为powerpc-linux-的GNU交叉编译工具。或者你使用的是MONTA
VISTA的CDK工具，你必须在Makefile里面定义CROSS_COMPILE。对于HHL 的4xx CPU，做以下更改

```
	CROSS_COMPILE = ppc_4xx-
```

U-Boot is intended to be simple to build. After installing   the
sources   you must configure U-Boot for one specific board type. This
is done by typing:
uboot被设计成简单的就可以构建。在安装源码后，你必须配置uboot为一个定义的board类型，这可以简单的输入make
NAME_config.

```
	make NAME_config
```

where "NAME_config" is the name of one of the existing
configurations; the following names are supported:
这里"NAME_config就是一个已经存在的配置的名字。支持以下配置:

```
	ADCIOP_config		FPS860L_config		omap730p2_config
	ADS860_config		GEN860T_config		pcu_e_config
	Alaska8220_config
	AR405_config		GENIETV_config		PIP405_config
	at91rm9200dk_config	GTH_config		QS823_config
	CANBT_config		hermes_config		QS850_config
	cmi_mpc5xx_config	hymod_config		QS860T_config
	cogent_common_config	IP860_config		RPXlite_config
	cogent_mpc8260_config	IVML24_config		RPXlite_DW_config
	cogent_mpc8xx_config	IVMS8_config		RPXsuper_config
	CPCI405_config		JSE_config		rsdproto_config
	CPCIISER4_config	LANTEC_config		Sandpoint8240_config
	csb272_config		lwmon_config		sbc8260_config
	CU824_config		MBX860T_config		sbc8560_33_config
	DUET_ADS_config		MBX_config		sbc8560_66_config
	EBONY_config		MPC8260ADS_config	SM850_config
	ELPT860_config		MPC8540ADS_config	SPD823TS_config
	ESTEEM192E_config	MPC8540EVAL_config	stxgp3_config
	ETX094_config		MPC8560ADS_config	SXNI855T_config
	FADS823_config		NETVIA_config		TQM823L_config
	FADS850SAR_config	omap1510inn_config	TQM850L_config
	FADS860T_config		omap1610h2_config	TQM855L_config
	FPS850L_config		omap1610inn_config	TQM860L_config
				omap5912osk_config	walnut_config
				omap2420h4_config	Yukon8220_config
							ZPC1900_config
```

Note: for some board special configuration names may exist; check if
      additional information is available from the board vendor; for
      instance, the TQM823L systems are available without (standard)
      or with LCD support. You can select such additional "features"
      when chosing the configuration, i. e.
注意：对于某些board可能存在特定的配置名字；从供应商那里确认是否有额外的信息存在；比如，TQM823L系统可以选择是否有LCD支持
，你可以在选择配置的时候选择额外的特性。

```
      make TQM823L_config
       - will configure for a plain TQM823L, i. e. no LCD support
             配置成简单的没有LCD支持的TQM823L
      make TQM823L_LCD_config
       - will configure for a TQM823L with U-Boot console on LCD
             配置TQM823L带LCD控制台的支持。
      etc.
```

Finally, type "make all", and you should get some working U-Boot
images ready for download to / installation on your system:
最后，输入make all，然后你将得到可以下载安装到你的系统的uboot映像。

```
- "u-boot.bin" is a raw binary image
       uboot.bin是raw格式映像
- "u-boot" is an image in ELF binary format
       u-boot是ELF二进制格式的映像
- "u-boot.srec" is in Motorola S-Record format
       uboot.srec是motorola s-record格式的文件

By default the build is performed locally and the objects are saved
in the source directory. One of the two methods can be used to change
this behavior and build U-Boot to some external directory:
默认情况下构建和目标文件的保存都是在当前源码目录。有两个方法可以改变这种情况把uboot构建在外部其他目录中。

1. Add O= to the make command line invocations:
1.     在命令行执行make命令后面添加O＝目录名称
       make O=/tmp/build distclean
       make O=/tmp/build NAME_config
       make O=/tmp/build all

2. Set environment variable BUILD_DIR to point to the desired location:
2.     设置环境变量BUILD_DIR 指向你期望的目录：
       export BUILD_DIR=/tmp/build
       make distclean
       make NAME_config
       make all

Note that the command line "O=" setting overrides the BUILD_DIR environment
variable.
注意命令行上的 "O="设定会覆盖BUILD_DIR环境变量。

Please be aware that the Makefiles assume you are using GNU make, so
for instance on NetBSD you might need to use "gmake" instead of
native "make".
请注意，Makefiles假设你用的是GNU make，假设在NetBSD上你可能需要用gmake代替make。

If the system board that you have is not listed, then you will need
to port U-Boot to your hardware platform. To do this, follow these
steps:
如果你需要的board没有在系统的列表中，那你可能需要移植uboot到你的硬件平台上。按以下步骤进行：
1.  Add a new configuration option for your board to the toplevel
    "Makefile" and to the "MAKEALL" script, using the existing
    entries as examples. Note that here and at many other places
    boards and other names are listed in alphabetical sort order. Please
    keep this order.
1.            添加一个你的board配置选项到顶层的Makefile和MAKEALL脚本，用里面的其他条目作参考。注意这些boards的名字是按字母
              顺序排列的，请保持这个顺序。

2.  Create a new directory to hold your board specific code. Add any
    files you need. In your board directory, you will need at least
    the "Makefile", a "<board>.c", "flash.c" and "u-boot.lds".
2.            创建一个新目录来保存你的board相关代码。添加必要的文件。在此目录中，你至少需要一个Makefile，一个"<board>.c",
        "flash.c" 和 "u-boot.lds"。

3.  Create a new configuration file "include/configs/<board>.h" for
    your board
3.            在"include/configs/<board>.h"里面创建你的board配置文件。

3.  If you're porting U-Boot to a new CPU, then also create a new
    directory to hold your CPU specific code. Add any files you need.
3.            如果你是在移植uboot到一个新的cpu，那么也应该创建一个新目录保存你的cpu相关代码。添加必要的文件到里面。

4.  Run "make <board>_config" with your new name.
4.            运行你的新的"make <board>_config"

5.  Type "make", and you should get a working "u-boot.srec" file
    to be installed on your target system.
5.            输入make，一会你将得到可以安装到你目标系统中的"u-boot.srec"。

6.  Debug and solve any problems that might arise.
```

    [Of course, this last step is much harder than it sounds.]
6.　　　　调试解决可能遇到的任何问题。当然这个步骤要比想象中困难。

Testing of U-Boot Modifications, Ports to New Hardware, etc.:
==
测试修改uboot，移植到新的硬件

If you have modified U-Boot sources (for instance added a newboard
or support for new devices, a new CPU, etc.) you are expected to
provide feedback to the other developers. The feedback normally takes
the form of a "patch", i. e. a context diff against a certain (latest
official or latest in CVS) version of U-Boot sources.
如果你修改了uboot的源码（比如给新的设备添加board级支持，移植到新的cpu等），我们期望你能给其他的开发者提供反馈。反馈的形式通常是提供一个补丁patch，或者针对正式或者最新CVS的uboot源代码的diff文本。

But before you submit such a patch, please verify that your modifi-
cation did not break existing code. At least make sure that *ALL* of
the supported boards compile WITHOUT ANY compiler warnings. To do so,
just run the "MAKEALL" script, which will configure and build U-Boot
for ALL supported system. Be warned, this will take a while. You can
select which (cross) compiler to use by passing a `CROSS_COMPILE'
environment variable to the script, i. e. to use the cross tools from
MontaVista's Hard Hat Linux you can type
但是在你提交这样的patch之前，请确认你的修改不会破坏现行的代码。至少确定所有支持的boards编译时候不会有任何编译警告。要做这个测试，运行MAKEALL脚本，它会为uboot支持的所有系统配置和构建。当然这会很耗时间。你可以选择哪一个交叉编译器传递给脚本的CROSS_COMPILE环境变量。对于使用MontaVista's Hard Hat Linux的交叉工具的，可以输入

```
	CROSS_COMPILE=ppc_8xx- MAKEALL
```

or to build on a native PowerPC system you can type
或者构建本地powerpc系统你可以输入

```
	CROSS_COMPILE=' ' MAKEALL
```

When using the MAKEALL script, the default behaviour is to build U-Boot
in the source directory. This location can be changed by setting the
BUILD_DIR environment variable. Also, for each target built, the MAKEALL
script saves two log files (<target>.ERR and <target>.MAKEALL) in the
<source dir>/LOG directory. This default location can be changed by
setting the MAKEALL_LOGDIR environment variable. For example:
当使用MAKEALL脚本时候，默认是在当前源码目录构建uboot。这可以通过设置BUILD_DIR环境变量来改变构建目录。当然，对于每一个目标构建，MAKEALL脚本会在<source dir>/LOG 目录保存两个日记文件(<target>.ERR and <target>.MAKEALL)。这也可以通过设置环境变量MAKEALL_LOGDIR来改变保存目录。

```
	export BUILD_DIR=/tmp/build
	export MAKEALL_LOGDIR=/tmp/log
	CROSS_COMPILE=ppc_8xx- MAKEALL
```

With the above settings build objects are saved in the /tmp/build, log
files are saved in the /tmp/log and the source tree remains clean during
the whole build process.
上面的设定中，构建目标保存在 /tmp/build，日记文件保存在 /tmp/log，源代码树在整个构建过程中依然保持干净。

See also "U-Boot Porting Guide" below.

Monitor Commands - Overview:
============================
Monitor命令－概述

```
go     - start application at address 'addr'
             运行addr处的应用程序
run    - run commands in an environment variable
             在环境变量里面运行命令
bootm  - boot application image from memory
             从内存启动映像
bootp  - boot image via network using BootP/TFTP protocol
             用BootP/TFTP协议通过网络启动映像
tftpboot- boot image via network using TFTP protocol
             and env variables "ipaddr" and "serverip"
             (and eventually "gatewayip")
             使用TFTP协议和ipaddr以及serverip环境变量通过网络启动映像
```

```
rarpboot- boot image via network using RARP/TFTP protocol
             使用RARP/TFTP协议通过网络启动映像
diskboot- boot from IDE devicebootd   - boot default, i.e., run 'bootcmd'
             从IDE设备启动
loads   - load S-Record file over serial line
             通过串口加载S-Record文件
loadb   - load binary file over serial line (kermit mode)
             通过串口使用kermit模式加载二进制文件
md      - memory display
             显示内存
mm      - memory modify (auto-incrementing)
             修改内存，自动递增
nm      - memory modify (constant address)
             修改内存，固定地址
mw      - memory write (fill)
             写内存，填充
cp      - memory copy
             内存拷贝
cmp     - memory compare
             内存比较
crc32   - checksum calculation
             校验和计算
imd     - i2c memory display
             i2c内存显示
imm     - i2c memory modify (auto-incrementing)
             i2c内存修改，自动递增
inm     - i2c memory modify (constant address)
             i2c内存修改，固定地址
imw     - i2c memory write (fill)
             i2c写内存，填充
icrc32 - i2c checksum calculation
             i2c校验和计算
iprobe - probe to discover valid I2C chip addresses
             探测是否存在i2c芯片地址
iloop   - infinite loop on address range
             在地址范围内的无限循环?
isdram - print SDRAM configuration information
             打印SDRAM配置信息
sspi    - SPI utility commands
             SPI应用命令
base    - print or set address offset
             打印设置地址偏移
printenv- print environment variables
             打印环境变量
setenv - set environment variables
             设置环境变量
saveenv - save environment variables to persistent storage
             保存环境变量到非易失存储器
protect - enable or disable FLASH write protection
             开启或者禁止FLASH写保护
erase   - erase FLASH memory
flinfo - print FLASH memory information
             打印flash内存信息
bdinfo - print Board Info structure
             打印board的结构信息
iminfo - print header information for application image
             打印映像头信息
coninfo - print console devices and informations
             打印控制台设备和信息
ide     - IDE sub-system
             IDE子系统
loop    - infinite loop on address range
             地址范围内的无限循环?
loopw   - infinite write loop on address range
             在地址范围内无限循环写?
mtest   - simple RAM test
             简单RAM测试
icache - enable or disable instruction cache
             开启或者禁止指令cache
dcache - enable or disable data cache
             开启或者禁止数据cache
```

```
reset   - Perform RESET of the CPU
              复位cpu
echo    - echo args to console
              回显参数到控制台
version - print monitor version
              打印monitor版本信息
help    - print online help
              打印在线帮助
?       - alias for 'help'
              帮助


Monitor Commands - Detailed Description:
========================================

TODO.

For now: just type "help <command>".


Environment Variables:
======================
环境变量

U-Boot supports user configuration using Environment Variables which
can be made persistent by saving to Flash memory.
uboot支持用户使用环境变量来配置，它可以永久保存在flash内存中。

Environment Variables are set using "setenv", printed using
"printenv", and saved to Flash using "saveenv". Using "setenv"
without a value can be used to delete a variable from the
environment. As long as you don't save the environment you are
working with an in-memory copy. In case the Flash area containing the
environment is erased by accident, a default environment is provided.
环境变量使用setenv来设置，printenv打印出来，saveenv保存到flash。使用不带值的setenv命令删除一个环境变量。在你保存环
境
变量之前你都是在使用内存的拷贝的环境变量。当flash中保存环境变量的区域不慎被擦除了，uboot提供一个默认的环境变量。

Some configuration options can be set using Environment Variables:
用环境变量可以设定某些配置选项

  baudrate    - see CONFIG_BAUDRATE
        波特率
  bootdelay   - see CONFIG_BOOTDELAY
        启动延时
  bootcmd     - see CONFIG_BOOTCOMMAND
        启动命令
  bootargs    - Boot arguments when booting an RTOS image
        启动参数
  bootfile    - Name of the image to load with TFTP
        启动文件
  autoload    - if set to "no" (any string beginning with 'n'),
                "bootp" will just load perform a lookup of the
                configuration from the BOOTP server, but not try to
                load any image using TFTP
        自动加载，如果设置为no，bootp仅会执行从bootp服务器查询配置，但不会用TFTP加载任何映像。

  autostart   - if set to "yes", an image loaded using the "bootp",
                "rarpboot", "tftpboot" or "diskboot" commands will
                be automatically started (by internally calling
                "bootm")
        自动启动，如果设置为yes，使用"bootp"或者"rarpboot", "tftpboot" or "diskboot"会自动加载一个映像。
        如果设置为no，一个独立的映像会传递给bootm命令然后拷贝到加载地址甚至解压，但是不会启动。这可以被用来加载解压任意
数据。
                If set to "no", a standalone image passed to the
                "bootm" command will be copied to the load address
                (and eventually uncompressed), but NOT be started.
                This can be used to load and uncompress arbitrary
                data.

  i2cfast     - (PPC405GP|PPC405EP only)
```

```
                if set to 'y' configures Linux I2C driver for fast
                mode (400kHZ). This environment variable is used in
                initialization code. So, for changes to be effective
                it must be saved and board must be reset.

  initrd_high - restrict positioning of initrd images:
                If this variable is not set, initrd images will be
                copied to the highest possible address in RAM; this
                is usually what you want since it allows for
                maximum initrd size. If for some reason you want to
                make sure that the initrd image is loaded below the
                CFG_BOOTMAPSZ limit, you can set this environment
                variable to a value of "no" or "off" or "0".
                Alternatively, you can set it to a maximum upper
                address to use (U-Boot will still check that it
                does not overwrite the U-Boot stack and data).
```

限制了initrd映像的位置：如果没有设置这个变量，initrd映像会尽可能拷贝到RAM的最高地址；这可能在你想让initrd映像可以保持最大尺寸的时候有用。出于某种原因你可能想一定要initrd映像加载到CFG_BOOTMAPSZ 限制之下，你可以设置此环境变量为no或者off或者0。或者，你可以设置它到一个最高的地址来使用，uboot仍然会检查它是否覆盖了uboot堆栈和数据。

```
                For instance, when you have a system with 16 MB
                RAM, and want to reserve 4 MB from use by Linux,
                you can do this by adding "mem=12M" to the value of
                the "bootargs" variable. However, now you must make
                sure that the initrd image is placed in the first
                12 MB as well - this can be done with
```

比如，你的系统有16MB的RAM，想保留其中4MB给linux，你可以添加mem＝12M到bootargs参数中。但是你必须确定initrd映像是保存在12MB地址空间里面，这可以通过设置环境变量initrd_high来实现（0x00c00000＝12M）

```
                setenv initrd_high 00c00000
```

如果你设置initrd_high是0xffffffff，这告诉uboot所有的地址对于linux kernel来说都是合法的，包括flash内存。这种情况下uboot不会拷贝任何ramdisk的。这可能可以节省启动时间，但是需要你的linux kernel支持这个特性。

```
                If you set initrd_high to 0xFFFFFFFF, this is an
                indication to U-Boot that all addresses are legal
                for the Linux kernel, including addresses in flash
                memory. In this case U-Boot will NOT COPY the
                ramdisk at all. This may be useful to reduce the
                boot time on your system, but requires that this
                feature is supported by your Linux kernel.

  ipaddr      - IP address; needed for tftpboot command
                           ip地址；tftpboot命令需要

  loadaddr    - Default load address for commands like "bootp",
                "rarpboot", "tftpboot", "loadb" or "diskboot"
                           "bootp""rarpboot""tftpboot""loadb""diskboot"默认加载地址

  loads_echo  - see CONFIG_LOADS_ECHO
                                          加载回显
  serverip    - TFTP server IP address; needed for tftpboot command
                                  TFTP服务器IP地址；tftpboot命令需要

  bootretry   - see CONFIG_BOOT_RETRY_TIME
                           启动重试次数
  bootdelaykey      - see CONFIG_AUTOBOOT_DELAY_STR

  bootstopkey - see CONFIG_AUTOBOOT_STOP_STR

  ethprime    - When CONFIG_NET_MULTI is enabled controls which
                interface is used first.
                     那个eth最先使用，       CONFIG_NET_MULTI要有效。
  ethact      - When CONFIG_NET_MULTI is enabled controls which
                interface is currently active. For example you
                can do the following
                     激活那个eth为当前接口，前提是CONFIG_NET_MULTI要有效。
                => setenv ethact FEC ETHERNET
                => ping 192.168.0.1 # traffic sent on FEC ETHERNET
                => setenv ethact SCC ETHERNET
                => ping 10.0.0.1 # traffic sent on SCC ETHERNET

   netretry   - When set to "no" each network operation will
                either succeed or fail without retrying.
                When set to "once" the network operation will
                fail when all the available network interfaces
                are tried once without success.
                Useful on scripts which control the retry operation
                themselves.
                     当设置为no的时候，网络操作成功或者失败都不会重试。当设置为once时候，所有网络接口在失败时候会尝试
重操作一次。
  tftpsrcport - If this is set, the value is used for TFTP's
                UDP source port.
                     TFTP的UDP源端口号
  tftpdstport - If this is set, the value is used for TFTP's UDP
                destination port instead of the Well Know Port 69.
                     TFTP的UDP目标端口号，设置这个以代替已知的69号端口
   vlan              - When set to a value < 4095 the traffic over
                ethernet is encapsulated/received over 802.1q
                VLAN tagged frames.
                     当设置一个<4095的值，以太网传输通过802.lq VLAN桢格式封装和接收。

The following environment variables may be used and automatically
updated by the network boot commands ("bootp" and "rarpboot"),
depending the information provided by your boot server:
可以使用的环境变量如下，它们可以自动通过网络启动命令bootp and rarpboot来更新，依赖于你的启动服务器提供的信息：

  bootfile    - see above
  dnsip              - IP address of your Domain Name Server
  dnsip2      - IP address of your secondary Domain Name Server
  gatewayip   - IP address of the Gateway (Router) to use
  hostname    - Target hostname
  ipaddr      - see above
  netmask     - Subnet Mask
  rootpath    - Pathname of the root filesystem on the NFS server
  serverip    - see above


There are two special Environment Variables:
这里有两个特别的环境变量
  serial#     - contains hardware identification information such
                as type string and/or serial number
                                   包含硬件标志信息如类型字串或者序列号
  ethaddr     - Ethernet address
                                   以太网地址
These variables can be set only once (usually during manufacturing of
the board). U-Boot refuses to delete or overwrite these variables
once they have been set once.
这些变量只能设置一次（通常在制造board的时候）。当设置后，uboot拒绝删除或者覆盖这些变量。

Further special Environment Variables:
只读的环境变量
  ver         - Contains the U-Boot version string as printed
                with the "version" command. This variable is
                readonly (see CONFIG_VERSION_VARIABLE).
                             包含uboot版本，使用ersion命令打印。只读。

Please note that changes to some configuration parameters may take
only effect after the next boot (yes, that's just like Windoze :-).
请注意某些配置参数可能重启过后才起作用（yes，就像windoze一样）；

Command Line Parsing:
=====================
命令行解析器

There are two different command line parsers available with U-Boot:
```

```
the old "simple" one, and the much more powerful "hush" shell:
uboot存在两种不同的命令行解析器：老旧简单的，强大的hush shell：

Old, simple command line parser:
--------------------------------
旧的简单的命令行解析器

- supports environment variables (through setenv / saveenv commands)
       支持环境变量操作
- several commands on one line, separated by ';'
       一行可以有好几个命令，用；隔开
- variable substitution using "... ${name} ..." syntax
       变量引用
- special characters ('$', ';') can be escaped by prefixing with '\',
       特殊转义字符
  for example:
       setenv bootcmd bootm \${address}
- You can also escape text by enclosing in single apostrophes, for example:
       setenv addip 'setenv bootargs $bootargs ip=$ipaddr:$serverip:$gatewayip:$netmask:$hostname::off'
       你可以用单撇号封闭文字串

Hush shell:
-----------

- similar to Bourne shell, with control structures like
  if...then...else...fi, for...do...done; while...do...done,
  until...do...done, ...
       和Bourne shell类似，可以带控制结构如if...then...else...fi，for...do...done; while...do...done,
  until...do...done, ...
- supports environment ("global") variables (through setenv / saveenv
  commands) and local shell variables (through standard shell syntax
  "name=value"); only environment variables can be used with "run"
  command
       支持环境变量（全局）和本地局部shell变量（name＝value），仅环境变量可以用run命令。

General rules:
--------------

(1) If a command line (or an environment variable executed by a "run"
    command) contains several commands separated by semicolon, and
    one of these commands fails, then the remaining commands will be
    executed anyway.
              如果一个命令行包含几个用分号分隔的命令，当其中一个命令失败了，其余的命令不受影响。

(2) If you execute several variables with one call to run (i. e.
    calling run with a list af variables as arguments), any failing
    command will cause "run" to terminate, i. e. the remaining
    variables are not executed.
              如果你用run调用了几个变量，任何失败的命令都会导致run命令终止。

Note for Redundant Ethernet Interfaces:
=======================================
冗余以太网接口注意事项

Some boards come with redundant ethernet interfaces; U-Boot supports
such configurations and is capable of automatic selection of a
"working" interface when needed. MAC assignment works as follows:
某些board有冗余以太网接口；uboot支持这样的配置，而且它可以在需要时候选择一个工作的接口。MAC分配工作如下：网络接口的编号
为eth0，eth1，eth2...对应的MAC地址可以存储在环境变量如ethaddr(=>eth0),"eth1addr" (=>eth1), "eth2addr", ...
Network interfaces are numbered eth0, eth1, eth2, ... Corresponding
MAC addresses can be stored in the environment as "ethaddr" (=>eth0),
"eth1addr" (=>eth1), "eth2addr", ...

If the network interface stores some valid MAC address (for instance
in SROM), this is used as default address if there is NO correspon-
ding setting in the environment; if the corresponding environment
variable is set, this overrides the settings in the card; that means:
如果网络接口保存了有效的MAC地址（如在SROM），这会用作默认值如果没有在环境中设置对应的设置；如果设置了，那么会覆盖card的
设定；这意味着：

o If the SROM has a valid MAC address, and there is no address in the
```

```
  environment, the SROM's address is used.
       如果SROM有有效的MAC地址，环境中没有对应的地址设定，那么会用SROM的。

o If there is no valid address in the SROM, and a definition in the
  environment exists, then the value from the environment variable is
  used.
       如果SROM中没有保存有效的地址，而且在环境中存在定义，那么会用环境中的。

o If both the SROM and the environment contain a MAC address, and
  both addresses are the same, this MAC address is used.
       如果SROM和环境中都定义了MAC地址，而且一样，那就使用这个MAC地址。

o If both the SROM and the environment contain a MAC address, and the
  addresses differ, the value from the environment is used and a
  warning is printed.
       如果SROM和环境中都定义了MAC地址，而且是不同的，那么环境中的会用到，并且会打印警告。

o If neither SROM nor the environment contain a MAC address, an error
  is raised.
       如果SROM和环境中都没有MAC地址，会引发一个错误。


Image Formats:
==============
映像格式

The "boot" commands of this monitor operate on "image" files which
can be basicly anything, preceeded by a special header; see the
definitions in include/image.h for details; basicly, the header
defines the following image properties:
monitor的boot命令可以对任何前面加了特殊文件头的映像文件操作；可以查看include/image.h了解详细的定义；一般来说，文件头定义了如下映像属性：

* Target Operating System (Provisions for OpenBSD, NetBSD, FreeBSD,
  4.4BSD, Linux, SVR4, Esix, Solaris, Irix, SCO, Dell, NCR, VxWorks,
  LynxOS, pSOS, QNX, RTEMS, ARTOS;
  Currently supported: Linux, NetBSD, VxWorks, QNX, RTEMS, ARTOS, LynxOS).
       目标操作系统（目前支持Linux, NetBSD, VxWorks, QNX, RTEMS, ARTOS, LynxOS)

* Target CPU Architecture (Provisions for Alpha, ARM, AVR32, Intel x86,
  IA64, MIPS, NIOS, PowerPC, IBM S390, SuperH, Sparc, Sparc 64 Bit;
  Currently supported: ARM, AVR32, Intel x86, MIPS, NIOS, PowerPC).
       目标cpu体系（目前支持ARM, AVR32, Intel x86, MIPS, NIOS, PowerPC)

* Compression Type (uncompressed, gzip, bzip2)
       压缩格式
* Load Address
       加载地址
* Entry Point
       入口
* Image Name
       映像名字
* Image Timestamp
       映像时间戳

The header is marked by a special Magic Number, and both the header
and the data portions of the image are secured against corruption by
CRC32 checksums.
文件头被特殊的魔数标志，而且文件头和映像的数据都被加上了对应的crc32校验和。


Linux Support:
==============
linux支持

Although U-Boot should support any OS or standalone application
easily, the main focus has always been on Linux during the design of
U-Boot.
虽然uboot应该很容易支持任何os或者独立的应用程序，设计uboot的主要的焦点总是集中在linux上

U-Boot includes many features that so far have been part of some
```

special "boot loader" code within the Linux kernel. Also, any
"initrd" images to be used are no longer part of one big Linux image;
instead, kernel and "initrd" are separate images. This implementation
serves several purposes:
因为uboot包含很多功能，目前为止已经成为linux kernel部分特殊的boot loader了。当然，任何使用的initrd映像已经不是庞大的linux映像的一部分了；取而代之的是kernel和initrd是独立的映像。这是有几个目的的：

- the same features can be used for other OS or standalone
  applications (for instance: using compressed images to reduce the
  Flash memory footprint)
  同样的功能可以用给其他os或者独立应用程序（比如使用压缩映像减少flash需求）

- it becomes much easier to port new Linux kernel versions because
  lots of low-level, hardware dependent stuff are done by U-Boot
  移植新版本的linux kernel变得很容易，因为很多底层的硬件相关的工作已经交给uboot了。

- the same Linux kernel image can now be used with different "initrd"
  images; of course this also means that different kernel images can
  be run with the same "initrd". This makes testing easier (you don't
  have to build a new "zImage.initrd" Linux image when you just
  change a file in your "initrd"). Also, a field-upgrade of the
  software is easier now.
  同样的linux kernel映像可以使用不同的initrd映像；这同样意味着不同的linux kernel可以用一样的initrd来运行。这也使得测试更容易些（当你仅仅是更改initrd的一个文件时候你不用重新构建一个新的linux 映像）。当然，文件层次的软件更新也很容易了。

Linux HOWTO:
============

Porting Linux to U-Boot based systems:

移植linux到基于boot的系统：

U-Boot cannot save you from doing all the necessary modifications to
configure the Linux device drivers for use with your target hardware
(no, we don't intend to provide a full virtual machine interface to
Linux :-).
uboot不会节省你使用你的目标硬件而进行的一切必要的配置修改linux设备驱动的工作量（no，我们不想为linux提供完全的虚拟机器接口）。

But now you can ignore ALL boot loader code (in arch/ppc/mbxboot).
但是现在你可以忽略所有的底层启动代码。

Just make sure your machine specific header file (for instance
include/asm-ppc/tqm8xx.h) includes the same definition of the Board
Information structure as we define in include/u-boot.h, and make
sure that your definition of IMAP_ADDR uses the same value as your
U-Boot configuration in CFG_IMMR.
仅需确认你的机器特定的头文件包含和我们在include/u-boot.h一样的板级信息结构定义，确认你的IMAP_ADDR和uboot中配置一样。

Configuring the Linux kernel:
----
linux kernel 配置

No specific requirements for U-Boot. Make sure you have some root
device (initial ramdisk, NFS) for your target system.
对uboot 来说没有特殊的要求。请确认你的目标系统有一个可以启动的设备（initial ramdisk，NFS）。

Building a Linux Image:

构建linux 映像

With U-Boot, "normal" build targets like "zImage" or "bzImage" are
not used. If you use recent kernel source, a new build target
"uImage" will exist which automatically builds an image usable by
U-Boot. Most older kernels also have support for a "pImage" target,
which was introduced for our predecessor project PPCBoot and uses a
100% compatible format.
对于uboot，正常的构建目标比如zimage，bzimage是没用的。如果你使用的是近来的内核源码，一个叫uimage的新的构建目标会自动

为uboot构建一个映像。

Example:

```
	make TQM850L_config
	make oldconfig
	make dep
	make uImage
```

The "uImage" build target uses a special tool (in 'tools/mkimage') to
encapsulate a compressed Linux kernel image with header	 information,
CRC32 checksum etc. for use with U-Boot. This is what we are doing:
uimage构建目标使用特殊的工具(in 'tools/mkimage')对linux kernel映像封装了文件头信息，crc32校验和信息。为了能使用uboot这是我们的步骤:

```
* build a standard "vmlinux" kernel image (in ELF binary format):
	构建标准的vmlinux内核映像（elf二进制格式）
* convert the kernel into a raw binary image:
	转换内核到raw的二进制格式映像
	${CROSS_COMPILE}-objcopy -O binary \
				 -R .note -R .comment \
				 -S vmlinux linux.bin

* compress the binary image:
	压缩二进制映像
	gzip -9 linux.bin

* package compressed binary image for U-Boot:
为uboot打包压缩的二进制映像
	mkimage -A ppc -O linux -T kernel -C gzip \
		-a 0 -e 0 -n "Linux Kernel Image" \
		-d linux.bin.gz uImage
```

The "mkimage" tool can also be used to create ramdisk images for use
with U-Boot, either separated from the Linux kernel image, or
combined into one file. "mkimage" encapsulates the images with a 64
byte header containing information about target architecture,
operating system, image type, compression method, entry points, time
stamp, CRC32 checksums, etc.
"mkimage"工具也可以用来创建ramdisk映像给uboot使用，不管是和linux kernel映像分开还是复合成一个文件。"mkimage"把映像和64字节文件头封装在一起。

"mkimage" can be called in two ways: to verify existing images and
print the header information, or to build new images.
"mkimage"可以用在两个方面：验证已有的映像并打印文件头信息，或者构建一个新的映像。

In the first form (with "-l" option) mkimage lists the information
contained in the header of an existing U-Boot image; this includes
checksum verification:
第一个方面（－l选项）的mkimage列举了已有的uboot映像的文件头信息，包括校验和

```
	tools/mkimage -l image
	  -l ==> list image header information
```

The second form (with "-d" option) is used to build a U-Boot image
from a "data file" which is used as image payload:
第二个方面（－d选项）用来构建uboot映像

```
	tools/mkimage -A arch -O os -T type -C comp -a addr -e ep \
		      -n name -d data_file image
	  -A ==> set architecture to 'arch'
			  设置体系结构
	  -O ==> set operating system to 'os'
			  设置操作系统
	  -T ==> set image type to 'type'
			  设置映像类型
	  -C ==> set compression type 'comp'
			  设置压缩类型
	  -a ==> set load address to 'addr' (hex)
			  设置加载地址（hex）
	  -e ==> set entry point to 'ep' (hex)
```

```
                            设置入口点
         -n ==> set image name to 'name'
                            设置映像名字
         -d ==> use image data from 'datafile'
                            用来做映像文件的datafile
```

Right now, all Linux kernels for PowerPC systems use the same load
address (0x00000000), but the entry point address depends on the
kernel version:
目前，所有的powerpc系统使用的linux kernel都是相同的加载地址（0x00000000），但是入口地址取决于不同的linux版本

- 2.2.x kernels have the entry point at 0x0000000C,
- 2.3.x and later kernels have the entry point at 0x00000000.

So a typical call to build a U-Boot image would read:
下面是一个典型的构建uboot用的映像步骤

```
       -> tools/mkimage -n '2.4.4 kernel for TQM850L' \
       > -A ppc -O linux -T kernel -C gzip -a 0 -e 0 \
       > -d /opt/elsk/ppc_8xx/usr/src/linux-2.4.4/arch/ppc/coffboot/vmlinux.gz \
       > examples/uImage.TQM850L
       Image Name:   2.4.4 kernel for TQM850L
       Created:      Wed Jul 19 02:34:59 2000
       Image Type:   PowerPC Linux Kernel Image (gzip compressed)
       Data Size:    335725 Bytes = 327.86 kB = 0.32 MB
       Load Address: 0x00000000
       Entry Point:  0x00000000
```

To verify the contents of the image (or check for corruption):
校验映像内容

```
       -> tools/mkimage -l examples/uImage.TQM850L
       Image Name:   2.4.4 kernel for TQM850L
       Created:      Wed Jul 19 02:34:59 2000
       Image Type:   PowerPC Linux Kernel Image (gzip compressed)
       Data Size:    335725 Bytes = 327.86 kB = 0.32 MB
       Load Address: 0x00000000
       Entry Point:  0x00000000
```

NOTE: for embedded systems where boot time is critical you can trade
speed for memory and install an UNCOMPRESSED image instead: this
needs more space in Flash, but boots much faster since it does not
need to be uncompressed:
注意：对于那些启动时间很严格的嵌入式系统，你可以加快内存速度然后安装没有压缩的映像：这可能需要更多的flash空间，但是因为不用解压所以启动可能会快很多。

```
       -> gunzip /opt/elsk/ppc_8xx/usr/src/linux-2.4.4/arch/ppc/coffboot/vmlinux.gz
       -> tools/mkimage -n '2.4.4 kernel for TQM850L' \
       > -A ppc -O linux -T kernel -C none -a 0 -e 0 \
       > -d /opt/elsk/ppc_8xx/usr/src/linux-2.4.4/arch/ppc/coffboot/vmlinux \
       > examples/uImage.TQM850L-uncompressed
       Image Name:   2.4.4 kernel for TQM850L
       Created:      Wed Jul 19 02:34:59 2000
       Image Type:   PowerPC Linux Kernel Image (uncompressed)
       Data Size:    792160 Bytes = 773.59 kB = 0.76 MB
       Load Address: 0x00000000
       Entry Point:  0x00000000
```

Similar you can build U-Boot images from a 'ramdisk.image.gz' file
when your kernel is intended to use an initial ramdisk:
类似的你可以构建uboot用的'ramdisk.image.gz'文件，当你的kernel使用初始化ramdisk时

```
       -> tools/mkimage -n 'Simple Ramdisk Image' \
       > -A ppc -O linux -T ramdisk -C gzip \
       > -d /LinuxPPC/images/SIMPLE-ramdisk.image.gz examples/simple-initrd
       Image Name:   Simple Ramdisk Image
       Created:      Wed Jan 12 14:01:50 2000
       Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
       Data Size:    566530 Bytes = 553.25 kB = 0.54 MB
       Load Address: 0x00000000
       Entry Point:  0x00000000
```

```
Installing a Linux Image:
-------------------------
```
安装linux 映像

```
To downloading a U-Boot image over the serial (console) interface,
you must convert the image to S-Record format:
```
如果要通过串口控制台接口下载uboot映像，你需要把image转换成S－Record格式：

```
	objcopy -I binary -O srec examples/image examples/image.srec
```

```
The 'objcopy' does not understand the information in the U-Boot
image header, so the resulting S-Record file will be relative to
address 0x00000000. To load it to a given address, you need to
specify the target address as 'offset' parameter with the 'loads'
command.
```
'objcopy'不清楚uboot映像文件头的信息，所以S-Record文件的参考地址会是0x00000000。为了加载到指定地址，你需要使用带offset参数指定目标地址的loads命令。

```
Example: install the image to address 0x40100000 (which on the
TQM8xxL is in the first Flash bank):

	=> erase 40100000 401FFFFF

	.......... done
	Erased 8 sectors

	=> loads 40100000
	## Ready for S-Record download ...
	~>examples/image.srec
	1 2 3 4 5 6 7 8 9 10 11 12 13 ...
	...
	15989 15990 15991 15992
	[file transfer complete]
	[connected]
	## Start Addr = 0x00000000


You can check the success of the download using the 'iminfo' command;
this includes a checksum verification so you  can  be  sure  no	 data
corruption happened:
```
你可以用iminfo命令检查那些成功下载的映像；这包含了校验和信息这样你可以确定数据完整性。

```
	=> imi 40100000

	## Checking Image at 40100000 ...
	   Image Name:	 2.2.13 for initrd on TQM850L
	   Image Type:	 PowerPC Linux Kernel Image (gzip compressed)
	   Data Size:  335725 Bytes = 327 kB = 0 MB
	   Load Address: 00000000
	   Entry Point:	 0000000c
	   Verifying Checksum ... OK


Boot Linux:
-----------
```
启动linux

```
The "bootm" command is used to boot an application that is stored in
memory (RAM or Flash). In case of a Linux kernel image, the contents
of the "bootargs" environment variable is passed to the kernel as
parameters. You can check and modify this variable using the
"printenv" and "setenv" commands:
```
bootm命令用来启动存储在内存中的应用程序。对于linux 内核映像，bootargs环境变量会作为参数传递给kernel，你可以用"printenv" and "setenv"检查和修改这些变量。

```
	=> printenv bootargs
	bootargs=root=/dev/ram
	=> setenv bootargs root=/dev/nfs rw nfsroot=10.0.0.2:/LinuxPPC nfsaddrs=10.0.0.99:10.0.0.2

	=> printenv bootargs
	bootargs=root=/dev/nfs rw nfsroot=10.0.0.2:/LinuxPPC nfsaddrs=10.0.0.99:10.0.0.2

	=> bootm 40020000
	## Booting Linux kernel at 40020000 ...
	   Image Name:	  2.2.13 for NFS on TQM850L
	   Image Type:	  PowerPC Linux Kernel Image (gzip compressed)
	   Data Size:  381681 Bytes = 372 kB = 0 MB
	   Load Address: 00000000
	   Entry Point:	  0000000c
	   Verifying Checksum ... OK
	   Uncompressing Kernel Image ... OK
	Linux version 2.2.13 (wd@denx.local.net) (gcc version 2.95.2 19991024 (release)) #1 Wed Jul 19
02:35:17 MEST 2000
	Boot arguments: root=/dev/nfs rw nfsroot=10.0.0.2:/LinuxPPC nfsaddrs=10.0.0.99:10.0.0.2
	time_init: decrementer frequency = 187500000/60
	Calibrating delay loop... 49.77 BogoMIPS
	Memory: 15208k available (700k kernel code, 444k data, 32k init) [c0000000,c1000000]
	…
```

If you want to boot a Linux kernel with initial ram disk, you pass
the memory addresses of both the kernel and the initrd image (PPBCOOT
format!) to the "bootm" command:
如果你需要initial ram disk的linux kernel启动，你可以传递两者的内存地址给bootm命令：

```
	=> imi 40100000 40200000

	## Checking Image at 40100000 ...
	   Image Name:	  2.2.13 for initrd on TQM850L
	   Image Type:	  PowerPC Linux Kernel Image (gzip compressed)
	   Data Size:  335725 Bytes = 327 kB = 0 MB
	   Load Address: 00000000
	   Entry Point:	  0000000c
	   Verifying Checksum ... OK

	## Checking Image at 40200000 ...
	   Image Name:	  Simple Ramdisk Image
	   Image Type:	  PowerPC Linux RAMDisk Image (gzip compressed)
	   Data Size:  566530 Bytes = 553 kB = 0 MB
	   Load Address: 00000000
	   Entry Point:	  00000000
	   Verifying Checksum ... OK

	=> bootm 40100000 40200000
	## Booting Linux kernel at 40100000 ...
	   Image Name:	  2.2.13 for initrd on TQM850L
	   Image Type:	  PowerPC Linux Kernel Image (gzip compressed)
	   Data Size:  335725 Bytes = 327 kB = 0 MB
	   Load Address: 00000000
	   Entry Point:	  0000000c
	   Verifying Checksum ... OK
	   Uncompressing Kernel Image ... OK
	## Loading RAMDisk Image at 40200000 ...
	   Image Name:	  Simple Ramdisk Image
	   Image Type:	  PowerPC Linux RAMDisk Image (gzip compressed)
	   Data Size:  566530 Bytes = 553 kB = 0 MB
	   Load Address: 00000000
	   Entry Point:	  00000000
	   Verifying Checksum ... OK
	   Loading Ramdisk ... OK
	Linux version 2.2.13 (wd@denx.local.net) (gcc version 2.95.2 19991024 (release)) #1 Wed Jul 19
02:32:08 MEST 2000
	Boot arguments: root=/dev/ram
	time_init: decrementer frequency = 187500000/60
	Calibrating delay loop... 49.77 BogoMIPS
	…
	RAMDISK: Compressed image found at block 0
	VFS: Mounted root (ext2 filesystem).

	bash#
```

Boot Linux and pass a flat device tree:

启动linux，传递一个flat设备树

First, U-Boot must be compiled with the appropriate defines. See the section
titled "Linux Kernel Interface" above for a more in depth explanation. The
following is an example of how to start a kernel and pass an updated
flat device tree:
首先，uboot必须配置相应的定义。见上面的"Linux Kernel Interface"或者更多信息。下面这个例子展示了怎样启动一个kernel和传递一个更新的flat设备树

```
=> print oftaddr
oftaddr=0x300000
=> print oft
oft=oftrees/mpc8540ads.dtb
=> tftp $oftaddr $oft
Speed: 1000, full duplex
Using TSEC0 device
TFTP from server 192.168.1.1; our IP address is 192.168.1.101
Filename 'oftrees/mpc8540ads.dtb'.
Load address: 0x300000
Loading: #
done
Bytes transferred = 4106 (100a hex)
=> tftp $loadaddr $bootfile
Speed: 1000, full duplex
Using TSEC0 device
TFTP from server 192.168.1.1; our IP address is 192.168.1.2
Filename 'uImage'.
Load address: 0x200000
Loading:############
done
Bytes transferred = 1029407 (fb51f hex)
=> print loadaddr
loadaddr=200000
=> print oftaddr
oftaddr=0x300000
=> bootm $loadaddr - $oftaddr
## Booting image at 00200000 ...
   Image Name:   Linux-2.6.17-dirty
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    1029343 Bytes = 1005.2 kB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
Booting using flat device tree at 0x300000
Using MPC85xx ADS machine description
Memory CAM mapping: CAM0=256Mb, CAM1=256Mb, CAM2=0Mb residual: 0Mb
[snip]
```

More About U-Boot Image Types:

更多uboot映像的类型信息

U-Boot supports the following image types:
uboot支持一下映像类型

   "Standalone Programs" are directly runnable in the environment
       provided by U-Boot; it is expected that (if they behave
       well) you can continue to work in U-Boot after return from
       the Standalone Program.
       标准裸机程序可以在uboot环境中直接运行；如果可能uboot可以在运行完应用程序后返回。

   "OS Kernel Images" are usually images of some Embedded OS which
       will take over control completely. Usually these programs
       will install their own set of exception handlers, device
       drivers, set up the MMU, etc. - this means, that you cannot
       expect to re-enter U-Boot except by resetting the CPU.

os内核映像通常是某些嵌入式OS，可以取得完全的控制权。通常这些程序会安装他们自己的异常句柄，设备驱动，设置MMU等，这意味
着你不可能返回uboot，除非重启。

"RAMDisk Images" are more or less just data blocks, and their
parameters (address, size) are passed to an OS kernel that is
being started.
RAMDisk映像多多少少仅仅是数据块，他们的参数（地址，尺寸）都会在开始时候传递给kernel

"Multi-File Images" contain several images, typically an OS
(Linux) kernel image and one or more data images like
RAMDisks. This construct is useful for instance when you want
to boot over the network using BOOTP etc., where the boot
server provides just a single image file, but you want to get
for instance an OS kernel and a RAMDisk image.
多文件格式的映像包含了几个映像，典型的是linux内核映像和一个或者几个数据映像如ramdisk。这种结构很有用当你需要通过网络
用bootp协议启动的时候，boot服务器仅能提供一个映像文件，而你需要得到kernel和ramdisk的映像。

"Multi-File Images" start with a list of image sizes, each
image size (in bytes) specified by an "uint32_t" in network
byte order. This list is terminated by an "(uint32_t)0".
Immediately after the terminating 0 follow the images, one by
one, all aligned on "uint32_t" boundaries (size rounded up to
a multiple of 4 bytes).
多映像格式的映像，开头通常是一个映像大小的列表，每个映像大小用一个"uint32_t"形式的网络字节顺序表示。列表
最后"(uint32_t)0"表示列表结束。紧接着是映像，一个接一个，都对齐到"uint32_t"的边界。

"Firmware Images" are binary images containing firmware (like
U-Boot or FPGA images) which usually will be programmed to
flash memory.
固件映像是包含固件的二进制映像，通常会写入flash内存

"Script files" are command sequences that will be executed by
U-Boot's command interpreter; this feature is especially
useful when you configure U-Boot to use a real shell (hush)
as command interpreter.
脚本文件是会被uboot命令行解析执行的命令序列；这特别有用当你配置uboot使用hush为命令行解析器时。

Standalone HOWTO:

裸奔程序howto

One of the features of U-Boot is that you can dynamically load and
run "standalone" applications, which can use some resources of
U-Boot like console I/O functions or interrupt services.
uboot的其中一个功能是你可以动态的加载运行应用程序，这些程序可以使用某些uboot的资源比如控制IO和中断服务功能。

Two simple examples are included with the sources:
源码中包含两个简单的程序

"Hello World" Demo:

'examples/hello_world.c' contains a small "Hello World" Demo
application; it is automatically compiled when you build U-Boot.
It's configured to run at address 0x00040004, so you can play with it
like that:
'examples/hello_world.c'包含一个hello world的小演示程序；当你构建uboot时候他自动编译进去了。它被配置成运行地址是0x00040004，你可以这样来运行他：

```
        => loads
        ## Ready for S-Record download ...
        ~>examples/hello_world.srec
        1 2 3 4 5 6 7 8 9 10 11 ...
        [file transfer complete]
        [connected]
        ## Start Addr = 0x00040004

        => go 40004 Hello World! This is a test.
        ## Starting application at 0x00040004 ...
        Hello World
        argc = 7
        argv[0] = "40004"
        argv[1] = "Hello"
        argv[2] = "World!"
        argv[3] = "This"
        argv[4] = "is"
        argv[5] = "a"
        argv[6] = "test."
        argv[7] = "<NULL>"
        Hit any key to exit ...

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

Another example, which demonstrates how to register a CPM interrupt
handler with the U-Boot code, can be found in 'examples/timer.c'.
Here, a CPM timer is set up to generate an interrupt every second.
The interrupt service routine is trivial, just printing a '.'
character, but this is just a demo program. The application can be
controlled by the following keys:
另外一个，展示了如何用uboot代码注册CPM中断句柄，可以在'examples/timer.c'找到他。这里，CPM定时器设置成每秒产生一个中断，中断服务程序很简单，仅仅是打印一个.字符，但这仅是演示用程序。这个程序可以用下面key控制

```
        ? - print current values og the CPM Timer registers
        b - enable interrupts and start timer
        e - stop timer and disable interrupts
        q - quit application

        => loads
        ## Ready for S-Record download ...
        ~>examples/timer.srec
        1 2 3 4 5 6 7 8 9 10 11 ...
        [file transfer complete]
        [connected]
        ## Start Addr = 0x00040004

        => go 40004
        ## Starting application at 0x00040004 ...
        TIMERS=0xfff00980
        Using timer 1
          tgcr @ 0xfff00980, tmr @ 0xfff00990, trr @ 0xfff00994, tcr @ 0xfff00998, tcn @ 0xfff0099c,
ter @ 0xfff009b0

Hit 'b':
        [q, b, e, ?] Set interval 1000000 us
        Enabling timer
Hit '?':
        [q, b, e, ?] ........
        tgcr=0x1, tmr=0xff1c, trr=0x3d09, tcr=0x0, tcn=0xef6, ter=0x0
Hit '?':
        [q, b, e, ?] .
        tgcr=0x1, tmr=0xff1c, trr=0x3d09, tcr=0x0, tcn=0x2ad4, ter=0x0
Hit '?':
        [q, b, e, ?] .
        tgcr=0x1, tmr=0xff1c, trr=0x3d09, tcr=0x0, tcn=0x1efc, ter=0x0
Hit '?':
        [q, b, e, ?] .
        tgcr=0x1, tmr=0xff1c, trr=0x3d09, tcr=0x0, tcn=0x169d, ter=0x0
Hit 'e':
        [q, b, e, ?] ...Stopping timer
Hit 'q':
        [q, b, e, ?] ## Application terminated, rc = 0x0
```

Minicom warning:
minicom注意事项

Over time, many people have reported problems when trying to use the
"minicom" terminal emulation program for serial download. I (wd)

consider minicom to be broken, and recommend not to use it. Under
Unix, I recommend to use C-Kermit for general purpose use (and
especially for kermit binary protocol download ("loadb" command), and
use "cu" for S-Record download ("loads" command).
在过去，很多人在尝试使用minicom终端模拟器的串口下载功能的时候都报告有问题，我认为是minicom的问题，并且建议不要使用它。在unix中，我建议使用c－kermit作为一般应用，特别是在kermit二进制协议下载和使用cu下载s－record格式文件的时候。

Nevertheless, if you absolutely want to use it try adding this
configuration to your "File transfer protocols" section:
如果你坚持一定要使用minicom，你可以尝试添加这个配置到你的"File transfer protocols"段

```
          Name          Program                  Name U/D FullScr IO-Red. Multi
       X  kermit  /usr/bin/kermit -i -l %l -s    Y    U          Y      N     N
       Y  kermit  /usr/bin/kermit -i -l %l -r    N    D          Y      N     N
```

NetBSD Notes:
=============
NetBSD注意：

Starting at version 0.9.2, U-Boot supports NetBSD both as host
(build U-Boot) and target system (boots NetBSD/mpc8xx).

Building requires a cross environment; it is known to work on
NetBSD/i386 with the cross-powerpc-netbsd-1.3 package (you will also
need gmake since the Makefiles are not compatible with BSD make).
Note that the cross-powerpc package does not install include files;
attempting to build U-Boot will fail because <machine/ansi.h> is
missing. This file has to be installed and patched manually:

```
       # cd /usr/pkg/cross/powerpc-netbsd/include
       # mkdir powerpc
       # ln -s powerpc machine
       # cp /usr/src/sys/arch/powerpc/include/ansi.h powerpc/ansi.h
       # ${EDIT} powerpc/ansi.h   ## must remove __va_list, _BSD_VA_LIST
```

Native builds *don't* work due to incompatibilities between native
and U-Boot include files.

Booting assumes that (the first part of) the image booted is a
stage-2 loader which in turn loads and then invokes the kernel
proper. Loader sources will eventually appear in the NetBSD source
tree (probably in sys/arc/mpc8xx/stand/u-boot_stage2/); in the
meantime, see ftp://ftp.denx.de/pub/u-boot/ppcboot_stage2.tar.gz

Implementation Internals:
=========================
内部实现原理

The following is not intended to be a complete description of every
implementation detail. However, it should help to understand the
inner workings of U-Boot and make it easier to port it to custom
hardware.
接下来不可能完全描述每一个实现细节。但是，它对明白uboot内部是如何工作的是有帮助的，对移植到定制的硬件的工作也更容易些。

Initial Stack, Global Data:

内部堆栈，全局数据

The implementation of U-Boot is complicated by the fact that U-Boot
starts running out of ROM (flash memory), usually without access to
system RAM (because the memory controller is not initialized yet).
This means that we don't have writable Data or BSS segments, and BSS
is not initialized as zero. To be able to get a C environment working
at all, we have to allocate at least a minimal stack. Implementation
options for this are defined and restricted by the CPU used: Some CPU
models provide on-chip memory (like the IMMR area on MPC8xx and
MPC826x processors), on others (parts of) the data cache can be
locked as (mis-) used as memory, etc.

uboot执行的复杂事实是uboot在flash ROM里面开始执行，通常访问不了系统RAM，因为此时内存控制器还没有初始化。这意味着我们没有可写的数据段和BSS段，而且BSS段也没有初始化为0。为了得到完整的可以工作的c环境，我们必须重定位到一个最小化的堆栈。这通常通过使用CPU来定义和限制：某些CPU提供芯片内置的内存（比如MPC8XX处理器的IMMR区域），另外数据cache也可以锁定或者复用为内存。

```
	Chris Hallinan posted a good summary of   these issues to  the
	u-boot-users mailing list:
```
Chris Hallinan发给uboot users 邮件列表上的邮件很好的总结了这个问题：

```
	Subject: RE: [U-Boot-Users] RE: More On Memory Bank x (nothingness)?
	From: "Chris Hallinan" <clh@net1plus.com>
	Date: Mon, 10 Feb 2003 16:43:46 -0500 (22:43 MET)
	…

	Correct me if I'm wrong, folks, but the way I understand it
	is this: Using DCACHE as initial RAM for Stack, etc, does not
	require any physical RAM backing up the cache. The cleverness
	is that the cache is being used as a temporary supply of
	necessary storage before the SDRAM controller is setup. It's
	beyond the scope of this list to expain the details, but you
	can see how this works by studying the cache architecture and
	operation in the architecture and processor-specific manuals.
```
伙计，如果发现我错了，请纠正，但是我是这样理解它的：使用DCACHE作为堆栈的初始化RAM，不需要任何物理内存备份cache。应该清楚的是在设置SDRAM控制器之前，cache可以用作存储需要的临时供给。在这里讨论的细节之外，你可以从特定的体系和处理器手册上通过学习cache的结构和操作明白这是怎么工作的。

```
	OCM is On Chip Memory, which I believe the 405GP has 4K. It
	is another option for the system designer to use as an
	initial stack/ram area prior to SDRAM being available. Either
	option should work for you. Using CS 4 should be fine if your
	board designers haven't used it for something that would
	cause you grief during the initial boot! It is frequently not
	used.
```
OCM是芯片内内存，我相信405G里面是4K大小。它是系统设计者在SDRAM可用之前拿来做初始化堆栈ram区域的另外一条途径。对你来说这通常也是可行的。使用cs4应该没问题，如果你的board设计者还没有使用它做某用途时候，否则，在初始化启动的时候这会给你带来痛苦！不过通常cs4都没有使用到。

```
	CFG_INIT_RAM_ADDR should be somewhere that won't interfere
	with your processor/board/system design. The default value
	you will find in any recent u-boot distribution in
	walnut.h should work for you. I'd set it to a value larger
	than your SDRAM module. If you have a 64MB SDRAM module, set
	it above 400_0000. Just make sure your board has no resources
	that are supposed to respond to that address! That code in
	start.S has been around a while and should work as is when
	you get the config right.
```
CFG_INIT_RAM_ADDR应该是一个不会干扰你的处理器board系统设计的值。在比较新的uboot源码中你可以找到walnut.h里面的默认值，它应该是没问题的。我会把它设置一个比你的SDRAM模组大的值。如果你有64MB的SDRAM，设置它在400_0000之上。不过请确认你的board在对应的地址上面没有其他的资源。这个代码在start.S里面，当你配置的当那么它应该是没有问题的。

```
	-Chris Hallinan
	DS4.COM, Inc.

It is essential to remember this, since it has some impact on the C
code for the initialization procedures:
```
很有必要牢记这些，因为在初始化流程中对c代码有很大影响：

```
* Initialized global data (data segment) is read-only. Do not attempt
  to write it.
```
初始化后的全局数据段是只读的，不要试图对它写入。

```
* Do not use any unitialized global data (or implicitely initialized
  as zero data - BSS segment) at all - this is undefined, initiali-
  zation is performed later (when relocating to RAM).
```
不要使用没有初始化的全局数据或者没有完全清0的BSS段数据，这是没定义的，当重定位到RAM后会执行初始化。

```
* Stack space is very limited. Avoid big data buffers or things like
  that.
        堆栈空间非常有限。避免使用类似于很大的数据缓存之类的数据。

Having only the stack as writable memory limits means we cannot use
normal global data to share information beween the code. But it
turned out that the implementation of U-Boot can be greatly
simplified by making a global data structure (gd_t) available to all
functions. We could pass a pointer to this data as argument to _all_
functions, but this would bloat the code. Instead we use a feature of
the GCC compiler (Global Register Variables) to share the data: we
place a pointer (gd) to the global data into a register which we
reserve for this purpose.
```

只有一个可写内存的堆栈限制意味着我们不能在代码之间通过全局变量来共享信息。但是uboot却是能通过构造能对所有函数所见的全局数据结构来非常简单的突破这个限制。我们可以给所有函数的这些数据传递一个指针作为参数，但是可能这会导致代码冗肿。取而代之的我们使用了一个gcc编译器的一个特性（全局寄存器变量）来共享数据：我们在一个寄存器里面放置一个指向全局数据的指针来达到这个目的。

```
When choosing a register for such a purpose we are restricted by the
relevant  (E)ABI  specifications for the current architecture, and by
GCC's implementation.
```

当选择一个寄存器作此目的的时候，我们受当前体系的有关EABI相关定义和GCC的实现的限制。

```
For PowerPC, the following registers have specific use:
对于PowerPC，下列寄存器有特殊用途

        R1:     stack pointer
                        堆栈指针
        R2:     TOC pointer
                        TOC指针
        R3-R4: parameter passing and return values
                        参数传递和返回值
        R5-R10: parameter passing
                        参数传递
        R13:    small data area pointer
                        少量数据区域指针
        R30:    GOT pointer
                        GOT指针
        R31:    frame pointer
                        祯指针

        (U-Boot also uses R14 as internal GOT pointer.)
                uboot使用R14作内部GOT指针
    ==> U-Boot will use R29 to hold a pointer to the global data
                uboot会使用R29保存一个指向全局数据的指针
    Note: on PPC, we could use a static initializer (since the
    address of the global data structure is known at compile time),
    but it turned out that reserving a register results in somewhat
    smaller code - although the code savings are not that big (on
    average for all boards 752 bytes for the whole U-Boot image,
    624 text + 127 data).
```

注意：在PPC，我们可以使用静态的初始化（因为全局数据结构的地址在编译的时候就已经知道），但是结果是保留一个寄存器以达

到更小的代码，虽然保存的代码不是特别大（对所有的boards整个uboot映像平均是752字节，624 text + 127 data。

```
On ARM, the following registers are used:
ARM平台，下列寄存器会用作：

        R0:     function argument word/integer result
                        函数参数 字/整数 结果
        R1-R3: function argument word
                        函数参数字
        R9:     GOT pointer
                        GOT指针
        R10:    stack limit (used only if stack checking if enabled)
                        堆栈限制（仅在堆栈检查允许的时候）
        R11:    argument (frame) pointer
                        参数（祯）指针
        R12:    temporary workspace
```

```
                    临时工作空间
      R13:    stack pointer
                    堆栈指针
      R14:    link register
                    连接寄存器
      R15:    program counter
                    程序计数器

    ==> U-Boot will use R8 to hold a pointer to the global data
                    uboot会使用R8保存全局数据的指针
NOTE: DECLARE_GLOBAL_DATA_PTR must be used with file-global scope,
or current versions of GCC may "optimize" the code too much.
```

Memory Management:

内存管理

U-Boot runs in system state and uses physical addresses, i.e. the
MMU is not used either for address mapping nor for memory protection.
uboot运行在系统态，并且使用物理内存。MMU的地址转换或者内存保护都没有用到。

The available memory is mapped to fixed addresses using the memory
controller. In this process, a contiguous block is formed for each
memory type (Flash, SDRAM, SRAM), even when it consists of several
physical memory banks.
可用的内存是那些使用内存控制器布局的固定地址的内存。在这处理过程中，每一种内存（flash，sdram，sram）都要求有一个连续的块
，即使它有几个物理的内存banks。

U-Boot is installed in the first 128 kB of the first Flash bank (on
TQM8xxL modules this is the range 0x40000000 ... 0x4001FFFF). After
booting and sizing and initializing DRAM, the code relocates itself
to the upper end of DRAM. Immediately below the U-Boot code some
memory is reserved for use by malloc() [see CFG_MALLOC_LEN
configuration setting]. Below that, a structure with global Board
Info data is placed, followed by the stack (growing downward).
uboot被安装到第一个flash bank的开始128kb处（对TQM8XXL来说这是0x40000000 ... 0x4001FFFF）。在启动初始化DRAM后，uboot把自己复制重定位到DRAM的高处。紧接着uboot代码的下面保留给malloc（）函数使用（见CFG_MALLOC_LEN配置选项）。然后紧接着这的下面是全局board信息的结构，然后是向下生长的堆栈。

Additionally, some exception handler code is copied to the low 8 kB
of DRAM (0x00000000 ... 0x00001FFF).
另外，某些异常句柄代码被拷贝到DRAM的最低8kb处(0x00000000 ... 0x00001FFF)

So a typical memory configuration with 16 MB of DRAM could look like
this:
所以一个16MB内存大小的典型布局可能像下面这样：

```
      0x0000 0000          异常向量表
           :
      0x0000 1FFF
      0x0000 2000          应用程序自由使用的区域
           :
           :

           :
           :
      0x00FB FF20          Monitor 堆栈（往下生长）
      0x00FB FFAC          Board Info 数据 和全局数据的拷贝
      0x00FC 0000          Malloc 用
           :
      0x00FD FFFF
      0x00FE 0000          Monitor在RAM中拷贝
      ...eventually: LCD or video 祯缓存
      ...eventually: 受保护的RAM
      0x00FF FFFF          RAM结束处
```

System Initialization:

系统初始化

In the reset configuration, U-Boot starts at the reset entry point
(on most PowerPC systens at address 0x00000100). Because of the reset
configuration for CS0# this is a mirror of the onboard Flash memory.
To be able to re-map memory U-Boot then jumps to its link address.
To be able to implement the initialization code in C, a (small!)
initial stack is set up in the internal Dual Ported RAM (in case CPUs
which provide such a feature like MPC8xx or MPC8260), or in a locked
part of the data cache. After that, U-Boot initializes the CPU core,
the caches and the SIU.
在复位配置中，uboot进入复位入口点开始启动（大多数PowerPC系统在地址0x00000100）。因为在复位配置中，CS0是板载flash的映像。为了能重映射内存，uboot然后跳转到它的连接地址。为了能实现用c代码来初始化，需要一个内部在dual port RAM中设置一个非常
小的初始化堆栈（MPC8XX或者MPC8260提供此类功能），或者是一个锁定的数据cache。然后uboot开始初始化cpu核心，cache，和SIU。

Next, all (potentially) available memory banks are mapped using a
preliminary mapping. For example, we put them on 512 MB boundaries
(multiples of 0x20000000: SDRAM on 0x00000000 and 0x20000000, Flash
on 0x40000000 and 0x60000000, SRAM on 0x80000000). Then UPM A is
programmed for SDRAM access. Using the temporary configuration, a
simple memory test is run that determines the size of the SDRAM
banks.
接着，使用初步的重定位使得所有（潜在的）内存banks都被重映射。比如我们把它们放在512MB的边界（0x20000000的整数倍SDRAM 在 0x00000000和 0x20000000, Flash在 0x40000000 和 0x60000000, SRAM 在 0x80000000)。UPM A被编程以便访问SDRAM。使用临时配置，一个简单的内存测试会运行以检测SDRAM banks的大小。

When there is more than one SDRAM bank, and the banks are of
different size, the largest is mapped first. For equal size, the first
bank (CS2#) is mapped first. The first mapping is always for address
0x00000000, with any additional banks following immediately to create
contiguous memory starting from 0.
当SDRAM有超过一个数量的bank时候，并且这些bank是不同大小的，最大的那个会首先重映射。对于相等尺寸的bank，CS2会首先映射。第一个映射的地址当然是0x00000000，其他的banks接跟着，这样就构建了从0连续开始的内存。

Then, the monitor installs itself at the upper end of the SDRAM area
and allocates memory for use by malloc() and for the global Board
Info data; also, the exception vector code is copied to the low RAM
pages, and the final stack is set up.
然后，monitor会在SDRAM区域的最高端安装自己并且会分配内存给malloc函数用，以及全局board信息数据用；当然异常向量表代码会拷贝到RAM的最低端，最后设置堆栈。

Only after this relocation will you have a "normal" C environment;
until that you are restricted in several ways, mostly because you are
running from ROM, and because the code will have to be relocated to a
new address in RAM.
当这些重定位完成后你才会有一个正常的c环境；在这之前你都是被几个因素限制住的，大部分都是因为你是在rom中运行，代码需要重新定位到RAM一个新的地址。

U-Boot Porting Guide:

----
uboot移植指南

[Based on messages by Jerry Van Baren in the U-Boot-Users mailing
list, October 2002]

```
int main (int argc, char *argv[])
{
	sighandler_t no_more_time;

	signal (SIGALRM, no_more_time);
	alarm (PROJECT_DEADLINE - toSec (3 * WEEK));

	if (available_money > available_manpower) {
		pay consultant to port U-Boot;
		return 0;
	}
	Download latest U-Boot source;

	Subscribe to u-boot-users mailing list;

	if (clueless) {
		email ("Hi, I am new to U-Boot, how do I get started?");
	}

	while (learning) {
		Read the README file in the top level directory;
		Read http://www.denx.de/twiki/bin/view/DULG/Manual ;
		Read the source, Luke;
	}

	if (available_money > toLocalCurrency ($2500)) {
		Buy a BDI2000;
	} else {
		Add a lot of aggravation and time;
	}

	Create your own board support subdirectory;

	Create your own board config file;

	while (!running) {
		do {
			Add / modify source code;
		} until (compiles);
		Debug;
		if (clueless)
			email ("Hi, I am having problems...");
	}
	Send patch file to Wolfgang;

	return 0;
}

void no_more_time (int sig)
{
      hire_a_guru();
}
```

Coding Standards:

代码标准

All contributions to U-Boot should conform to the Linux kernel
coding style; see the file "Documentation/CodingStyle" and the script
"scripts/Lindent" in your Linux kernel source directory. In sources
originating from U-Boot a style corresponding to "Lindent -pcs" (adding
spaces before parameters to function calls) is actually used.

Source files originating from a different project (for example the
MTD subsystem) are generally exempt from these guidelines and are not
reformated to ease subsequent migration to newer versions of those
sources.

Please note that U-Boot is implemented in C (and to some small parts in
Assembler); no C++ is used, so please do not use C++ style comments (//)
in your code.

Please also stick to the following formatting rules:
- remove any trailing white space
- use TAB characters for indentation, not spaces
- make sure NOT to use DOS '\r\n' line feeds
- do not add more than 2 empty lines to source files
- do not add trailing empty lines to source files

Submissions which do not conform to the standards may be returned
with a request to reformat the changes.

```
Submitting Patches:
-------------------
提交补丁

Since the number of patches for U-Boot is growing, we need to
establish some rules. Submissions which do not conform to these rules
may be rejected, even when they contain important and valuable stuff.

Patches shall be sent to the u-boot-users mailing list.

When you send a patch, please include the following information with
it:

* For bug fixes: a description of the bug and how your patch fixes
  this bug. Please try to include a way of demonstrating that the
  patch actually fixes something.

* For new features: a description of the feature and your
  implementation.

* A CHANGELOG entry as plaintext (separate from the patch)

* For major contributions, your entry to the CREDITS file

* When you add support for a new board, don't forget to add this
  board to the MAKEALL script, too.

* If your patch adds new configuration options, don't forget to
  document these in the README file.

* The patch itself. If you are accessing the CVS repository use "cvs
  update; cvs diff -puRN"; else, use "diff -purN OLD NEW". If your
  version of diff does not support these options, then get the latest
  version of GNU diff.

  The current directory when running this command shall be the top
  level directory of the U-Boot source tree, or it's parent directory
  (i. e. please make sure that your patch includes sufficient
  directory information for the affected files).

  We accept patches as plain text, MIME attachments or as uuencoded
  gzipped text.

* If one logical set of modifications affects or creates several
  files, all these changes shall be submitted in a SINGLE patch file.

* Changesets that contain different, unrelated modifications shall be
  submitted as SEPARATE patches, one patch per changeset.


Notes:

* Before sending the patch, run the MAKEALL script on your patched
  source tree and make sure that no errors or warnings are reported
  for any of the boards.

* Keep your modifications to the necessary minimum: A patch
  containing several unrelated changes or arbitrary reformats will be
  returned with a request to re-formatting / split it.

* If you modify existing code, make sure that your new code does not
  add to the memory footprint of the code ;-) Small is beautiful!
  When adding new features, these should compile conditionally only
  (using #ifdef), and the resulting code with the new feature
  disabled must not need more memory than the old code without your
  modification.

* Remember that there is a size limit of 40 kB per message on the
  u-boot-users mailing list. Compression may help.
```