EmSyS

Rev-1.0/14/Nov/2006

U-Boot, Redboot & Linux Kernel Porting Guide

By:

Rupesh Sugathan
(Rupesh.Sugathan@LnTEmSyS.com)

U-Boot Porting Guide

Capabilities of U-Boot

- ✓ Supports ARM, PPC, MIPS, X86 baords
- ✓ Contains bootup code
- ✓ Executes from either Flash or from Flash & RAM
- ✓ TCP/IP Network capable
- ✓ Load & Boot linux kernel & or Ramdisk (bootm command)
- ✓ Execute standalone applications (Go command)
- ✓ Pass Command line args to linux
- ✓ File download from serial, Ethernet, Flash
- ✓ Parameter storage on Flash sector
- ✓ Flash write, init, erase commands
- ✓ Boot compressed images
- ✓ Run scripts (made using mkimage)
- ✓ Boot from disk (IDE, CF card)
- ✓ Preboot command
- ✓ Display bmp images
- ✓ Option for altboot on many powerrecycle (spec for Carrier Grade Linux)
- ✓ Run cmds in an env variable
- ✓ Bootcmd, bootdelay
- ✓ Test commands
    - o Mem display
    - o Mem modify
    - o Memtest
    - o Many more…

Affected Dirs while porting U-Boot

- ✓ Include
    - ✓ CPU configuration header file
        - o Processor specific definitions for u-boot to access CPU
        - o Register definitions for both ANSI-C & PowerPC assemply.
    - ✓ Board configuration file under Configs/ dir.
        - o Hardware Config
        - o Memory map
        - o Peripheral definitions/configuration
- ✓ Board
    - ✓ External (off-chip) Flash driver/ initializations
    - ✓ External SDRAM Configurations
    - ✓ External device Configurations (FPGA, PCI etc)
- ✓ Cpu

- ✓ Chip select Configurations
- ✓ Internal Serial / USB drivers
- ✓ Internal Flash driver/ initializations

✓ Lib-xxx (lib-ppc, lib-arm etc)
- ✓ Contains a file board.c, which performs some of the generic board specific functionalities.

Porting procedure of U-Boot

1. Create a board specific Config file under Configs directory.
2. Create a new target in the main Makefile to load this include as the build configurations.
3. Configure u-boot by modifying the board configuration file in include/Configs/ dir
   - ✓ Chip select configuration for the defined memory map incorporating Flash, SDRAM and other memory-mapped peripherals.
   - ✓ Memory base addresses
   - ✓ Flash memory parameters (sector size, sector number etc)
   - ✓ Register Space address
   - ✓ SDRAM configurations
   - ✓ Default command line arguments to Linux kernel.
   - ✓ And many other configurations specific to your board.

4. Define the Linker memory Map for
   - ✓ CFG_MONITOR_BASE – address of u-boot in SDRAM
   - ✓ CFG_GBL_DATA_SIZE – global data size
   - ✓ CFG_MALLOC_LEN – heap size
   - ✓ CFG_ENV_ADDR, CFG_ENV_SIZE – Env space to store u-boot configs.
   - ✓ CFG_BOOTMAPSZ – operating system space.

5. Define all driver functions that are needed for the new board by placing the appropriate driver files under a newly created directory with the board name under the board/ directory.

6. Modify the CPU specific file and the lib-cpu specific file if required.

7. make <new board Config target>

8. make all

9. u-boot.bin (the executable)

Redboot Porting Guide

Capabilities of Redboot

- ✓ Supports ARM, PPC, SuperH, NEC, MIPS, X86
- ✓ Contains bootup code
- ✓ Executes from either Flash or Flash & RAM
- ✓ Supports TCP/IP Network capability
- ✓ Boot linux kernel & or Ramdisk (exec command)
- ✓ Execute standalone applications (go command)
- ✓ Pass Command line args to linux (through exec command)
- ✓ File download from serial, Ethernet, Flash (load)
- ✓ Parameter storage on Flash sector (with a filesystem)
- ✓ Flash write, init, Config, erase commands
- ✓ Boot compressed images
- ✓ Bootcmd, bootdelay
- ✓ Test commands
    - o Mem display
    - o Mem modify
    - o Memtest
    - o Many more tests..

Redboot porting can be categorized into three

Platform porting (custom board)
Variant porting (new processor based on the existing cpu core)
Architecture porting (new cpu core)

We will focus only on the Platform porting

Affected Dirs while porting Redboot

- ✓ Hal/Common (general interrupt configurations)
    - o Cdl
    - o Src
    - o include
- ✓ Hal/cpu/Arch (files for generic support of the cpu core)
    - o Cdl
    - o Src
    - o Misc
    - o include
- ✓ Hal/cpu/board (board specific files)

- Cdl
- Src
- Misc
- Include
  - pkgconf

Porting procedure of Redboot

1. Create a directory for your board under hal/cpu dir, where cpu is the cpu which makes up ur board
2. copy contents of a similar board under the same hal/cpu directory to this new directory,
3. Modify the following files under hal/cpu/board/cdl, to select/configure the correct configurations of the target build
   a. Hal_cpu_board.cdl
4. Modify the file ecos.db, under packages/ directory.
   a. Place the new platform's file paths
   b. Add new package & description for the new platform
   c. Add the package to the list of targets
5. Modify the following files under hal/cpu/board/include/pkgconf, to adjust the memory layout of the target
   a. *.h (memory layout for run time)
   b. *.ldi (memory layout for linker)
6. Modify the following files under hal/cpu/board/src/, to modify the HAL initialization code.
   a. Board.S (hal_hardware_init routine)
      i. Chip select
      ii. Interrupt control
      iii. Rtc
      iv. Disable cache
      v. Disable MMU (if needed)
   b. *.c
      i. serial port driver
      ii. other on chip peripheral drivers
      iii. communication interface table
      iv. virtual vector table
      v. page table
7. Modify the following files under hal/cpu/board/
   a. Hal_cpu_board.cdl
8. make <board_Config>
9. make

Linux Kernel Porting Guide

Affected directories while porting Linux kernel

- ✓ **Documentation/**
  Documentation of the kernel – good place to find a lot of useful informations
- ✓ **arch/**
  Contains architecture specific code. Each supported architecture contains a sub directory, ppc, arm etc. Each of this architecture subdirectory has four major subdirectories of interest
  - **arch/arm/kernel/**
    Architecture specific kernel code.
    - **arch/arm/kernel/head.S**
      Contains kernel entry, the entry point of the kernel. The file also contains the exception handling of the kernel. The file sets up the stack, clears BSS, loads MMU, and call the start_kernel() (located in init/main.c), which is in fact the first C function the kernel executes.
    - **arch/arm/kernel/setup.c**
      setup_arch() function in this file calls the board-specific setup function. The command line string and memory start and memory end are passed over to the caller of this function. The start and end addresses for the linked-in ramdisk image are also updated here.
  - **arch/arm/mm**
    Architecture specific memory management code
  - **arch/arm/lib**
    Architecture specific library code, vsprintf and so on.
  - **arch/arm/mach-mx3/**
    Platform specific code. Most of the changes will be here when you do a board level kernel porting.
    - **arch/arm/mach-mx3/setup.c (also named with board name)**
      Contains the platform-specific setup function. The various chip selects configuration, interrupts initialization, are done here.
    - **arch/arm/mach-mx3/irq.c**
      Contains code to handle the platform specific interrupt controller.
- ✓ **drivers/**
  Contains device driver code. You might need to modify/add code here.
- ✓ **include/asm-arm**
  Include directory that contains architecture specific includes.
- ✓ **init/**
  Contains kernel initialization code. Can modify if there are any specific activities to be done while initializing.

Porting procedure of Linux kernel

- ✓ **Create a directory for your target, (say my_arm11_board) under directory arch/arm/.**
  This directory should contain the interrupt handling, timer, initialization, and setup routines for your specific platform
- ✓ **Copy the files from the platform, which is closest to this new platform.**
- ✓ **Create a directory (my_arm11_board) for your target under include/asm-arm/ directory-**
  This directory will hold include files specific to your platform
- ✓ **Modify the files of the above mentioned directories to suit the target**
- ✓ **Modify files in the directories mentioned above under section, directories of interest, to custom scenarios.**
- ✓ **Hook the new platform to the build system, by modifying the Config files and the Makefiles under the arch/arm/ directory**
- ✓ **Make defconfig (to select default configuration of the board)**
- ✓ **Make**