am335x的cpu上电后，执行流程：ROM->MLO(SPL)->u-boot.img

**第一级bootloader:**引导加载程序，板子上电后会自动执行这些代码，如启动方式（SDcard、SPI、NOR），然后跳转至第二级bootloader。这些代码应该是存放在 176KB 的 ROM 中

第二级bootloader:MLO(SPL)，用于硬件初始化，关看门狗，关中断，设置CPU时钟频率、速度、加载uboot.img、dts等操作，MLO文件应该会被映射到 64 KB的 Internal SRAM 中。

第三级bootloader:u-boot.img, C代码的入口

MLO与u-boot执行流程如下：

reset //(arch/arm/cpu/armv7/start.S)  
save\_boot\_params\_ret //(arch/arm/cpu/armv7/start.S)  
  |- disable interrupts   
  |- cpu\_init\_cp15 //(arch/arm/cpu/armv7/start.S)  
  |   |- Invalidate L1 I/D  
  |   |- disable MMU stuff and caches  
  |- cpu\_init\_crit //(arch/arm/cpu/armv7/start.S)  
  |   |- lowlevel\_init //(arch/arm/cpu/armv7/lowlevel\_init.S)  
  |       |- Setup a temporary stack  
  |       |- Set up global data   
  |       |- s\_init //(arch/arm/cpu/armv7/am33xx/board.c)  
  |           |- watchdog\_disable  
  |           |- set\_uart\_mux\_conf  
  |           |- setup\_clocks\_for\_console  
  |           |- uart\_soft\_reset  
  |- \_main //(arch/arm/lib/crt0.S)  
      |(MLO)如果是 MLO 文件  
      |- board\_init\_f //(arch/arm/cpu/armv7/am33xx/board.c)  
      |   |- board\_early\_init\_f //(arch/arm/cpu/armv7/am33xx/board.c)  
      |   |   |- prcm\_init

      |   |   | |-scale\_vcores();

      |   |   | |- setup\_dplls(); //修改\*get\_dpll\_ddr\_params  
      |   |   |- set\_mux\_conf\_regs

      |   |   | |- enable\_board\_pin\_mux(); **//修改初始化引脚**  
      |   |- sdram\_init //(board/ti/am335x/board.c) //初始化 DDR,**修改为DDR3**  
      |- spl\_relocate\_stack\_gd  
      |- board\_init\_r //(common/spl/spl.c)      【完成u-boot.img  dts加载】  
          |- ...

          |- spl\_board\_init(); //spl段板载初始化，需修改为mmc0支持

          |   |-save\_omap\_boot\_params(); //保存启动参数boot params

|   |-am33xx\_spl\_board\_init() //设置PMIC,设置CORE Frequencies(do\_setup\_dpll)

          |-board\_boot\_order(spl\_boot\_list);

          | |-spl\_boot\_list[0] = spl\_boot\_device();

          |-announce\_boot\_device(spl\_boot\_list[i]);

          |-if (!spl\_load\_image(spl\_boot\_list[i])) //  
          |- spl\_load\_image //根据不同的启动方式加载 u-boot 镜像，**需要修改**  
          |- jump\_to\_image\_no\_args //进入u-boot代码运行  
        
    
      |(U-Boot)如果是U-Boot 镜像  
      |- board\_init\_f //(common/board\_f.c)  
      |   |- ...  
      |   |- initcall\_run\_list(init\_sequence\_f)     
      |   |- ...     
      |     
      |- relocate\_code //(arch/arm/lib/relocate.S) 代码重定位  
      |- relocate\_vectors //(arch/arm/lib/relocate.S) 向量表重定义  
      |- Set up final (full) environment   
      |- board\_init\_r //(common/board\_r.c)  
          |- initcall\_run\_list(init\_sequence\_r)//初始化各种外设  
              |- main\_loop()

1、修改board/wsdv/am335x/board.c文件

将函数static int read\_eeprom(struct am335x\_baseboard\_id \*header)修改成如下

|  |
| --- |
| static int read\_eeprom(struct am335x\_baseboard\_id \*header){  header->magic = 0xEE3355AA;  strcpy(header->name, "iRTK2");  strcpy(header->version, "V01");  strcpy(header->serial, "LYJ20180812");  return 0;  } |

2.修改引脚配置

|  |
| --- |
| 修改void enable\_board\_pin\_mux(void)  void enable\_board\_pin\_mux(void)  {  /\* Do board-specific muxes. \*/  if(board\_is\_iRTK2()){  }  else if (board\_is\_bone()) {  /\* Beaglebone pinmux \*/  configure\_module\_pin\_mux(mii1\_pin\_mux);  configure\_module\_pin\_mux(mmc0\_pin\_mux);  #if defined(CONFIG\_NAND)  configure\_module\_pin\_mux(nand\_pin\_mux);  #elif defined(CONFIG\_NOR)  configure\_module\_pin\_mux(bone\_norcape\_pin\_mux);  #else  configure\_module\_pin\_mux(mmc1\_pin\_mux);  #endif  } else if (board\_is\_gp\_evm()) {  ……………………………….  }  修改SD卡CD引脚：  static struct module\_pin\_mux mmc0\_pin\_mux[] = {  {OFFSET(mmc0\_dat3), (MODE(0) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_DAT3 \*/  {OFFSET(mmc0\_dat2), (MODE(0) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_DAT2 \*/  {OFFSET(mmc0\_dat1), (MODE(0) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_DAT1 \*/  {OFFSET(mmc0\_dat0), (MODE(0) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_DAT0 \*/  {OFFSET(mmc0\_clk), (MODE(0) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_CLK \*/  {OFFSET(mmc0\_cmd), (MODE(0) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_CMD \*/  /\*{OFFSET(mcasp0\_aclkr), (MODE(4) | RXACTIVE)},\*/ /\* MMC0\_WP \*/  {OFFSET(spi0\_cs1), (MODE(5) | RXACTIVE | PULLUP\_EN)}, /\* MMC0\_CD \*/  {-1},  }; |

修改spl fit支持

|  |
| --- |
| #ifdef CONFIG\_SPL\_LOAD\_FIT  int board\_fit\_config\_name\_match(const char \*name)  {  **if ( (board\_is\_gp\_evm() board\_is\_iRTK2() ) && !strcmp(name, "am335x-evm"))**  return 0;  else if (board\_is\_bone() && !strcmp(name, "am335x-bone"))  return 0;  else if (board\_is\_bone\_lt() && !strcmp(name, "am335x-boneblack"))  return 0;  else if ( (board\_is\_evm\_sk() || board\_is\_iRTK2() ) && !strcmp(name, "am335x-evmsk"))  return 0;  else if (board\_is\_bbg1() && !strcmp(name, "am335x-bonegreen"))  return 0;  else if (board\_is\_icev2() && !strcmp(name, "am335x-icev2"))  return 0;  else  return -1;  }  #endif |

**#if defined(CONFIG\_TI\_SECURE\_DEVICE)**

#define CONFIG\_SPL\_OS\_BOOT

#endif

修改环境变量：

/\*lyj: add ORIGINAL CODE\*/

//original: "loadfdt=load ${devtype} ${bootpart} ${fdtaddr} ${bootdir}/${fdtfile}\0" \

//modify: "loadfdt=load mmc ${mmcdev} ${fdtaddr} ${fdtfile}\0" \

//"loadbootscript=load mmc ${mmcdev} ${loadaddr} boot.scr\0" \

// "envboot=mmc dev ${mmcdev}; " \

// "if mmc rescan; then " \

// "echo SD/MMC found on device ${mmcdev};" \

// "if run loadbootscript; then " \

// "run bootscript;" \

// "else " \

// "if run loadbootenv; then " \

// "echo Loaded env from ${bootenvfile};" \

// "run importbootenv;" \

// "fi;" \

// "if test -n $uenvcmd; then " \

// "echo Running uenvcmd ...;" \

// "run uenvcmd;" \

// "fi;" \

// "fi;" \

// "fi;\0" \

// "mmcloados=run args\_mmc; " \

"mmcloados=run iRTK2\_mmc\_args; " \

"if test ${boot\_fdt} = yes || test ${boot\_fdt} = try; then " \

"if run loadfdt; then " \

"bootz ${loadaddr} - ${fdtaddr}; " \

"else " \

"if test ${boot\_fdt} = try; then " \

"bootz; " \

"else " \

"echo WARN: Cannot load the DT; " \

"fi; " \

"fi; " \

"else " \

"bootz; " \

"fi;\0" \

"bootargs\_defaults=setenv bootargs " \

"console=${console} " \

"${optargs}\0" \

"ramdisk=cramfs.img\0"\

"mmc\_load\_uimage=fatload mmc ${mmcdev} ${kernel\_addr\_r} ${bootfile};fatload mmc ${mmcdev} ${ramdisk\_addr\_r} ${ramdisk}\0"\

"mmc\_root=/dev/ram rw\0" \

"mmc\_root\_fs\_type=cramfs \0" \

"ip\_method=none\0" \

"mmc\_args=run bootargs\_defaults;" \

"setenv bootargs ${bootargs} " \

"root=${mmc\_root} " \

"initrd=${ramdisk\_addr\_r},32MB "\

"rootfstype=${mmc\_root\_fs\_type} ip=${ip\_method}\0" \

修改128M内存支持

|  |
| --- |
| 在ti\_armv7\_common.h 修改环境变量为：  #define DEFAULT\_LINUX\_BOOT\_ENV \  "loadaddr=0x82000000\0" \  "kernel\_addr\_r=0x82000000\0" \  **"fdtaddr=0x86000000\0" \**  **"fdt\_addr\_r=0x86000000\0" \**  "rdaddr=0x88080000\0" \  "ramdisk\_addr\_r=0x81600000\0" \  "scriptaddr=0x80000000\0" \  "pxefile\_addr\_r=0x80100000\0" \  **"bootm\_size=0x8000000\0" \**  "boot\_fdt=try\0"  修改设备树：  memory {  device\_type = "memory";  reg = <0x80000000 **0x8000000**>; /\* 0x10000000 256 MB --> 0x8000000 128 MB\*/  }; |