#boot from flash

```
cat spl/u-boot-spl-pad.bin u-boot.bin > u-boot-with-spl.bin
```

1.spl/u-boot-spl.bin---->spl/u-boot-spl.lds

```
_start()->arch/mips/cpu/xburst/x1000/start.S
board_init_f()->arch/mips/cpu/xburst/x1000/soc.c
board_init_r()->common/spl/spl.c                          #read u-boot
jump_to_image_no_args()-> arch/mips/cpu/xburst/x1000/soc.c        #jump u-boot
```

2.u-boot------>u-boot.lds

```
_start()->arch/mips/cpu/xburst/x1000/start.S          #u-boot run
board_init_f()->arch/mips/lib/board.c
board_init_r()->arch/mips/lib/board.c
main_loop()->common/main.c
[
       process_boot_delay(); ->common/main.c
       [
             s = getenv ("bootcmd");
             if(bootdelay != -1 && s && !abortboot(bootdelay))
                     run_command_list(s, -1, 0);->common/main.c
                    [
                           builtin_run_command_list(buff, flag);->common/main.c
                           builtin_run_command(line, 0)->common/main.c
                           cmd_process(flag, argc, argv, &repeatable, NULL)->common/command.c
                    ]
             else
                    return ;
        ]
        for (;;) {
        len = readline ("halley2" "-sfcnor# ");
        flag = 0;
        if (len > 0)
         strcpy (lastcommand, console_buffer);
        else if (len == 0)
         flag |= 0x0001;
        if (len == -1)
         puts ("<INTERRUPT>\n");
        else
         rc = run_command(lastcommand, flag);
        if (rc <= 0) {
         lastcommand[0] = 0;
        }
        }
]
```

#boot from usb

1.read ginfo from usb

2.read spl from usb

3.run spl:

```
_start()->arch/mips/cpu/xburst/x1000/start.S
board_init_f()->arch/mips/cpu/xburst/x1000/soc.c
```

4.return to bootroom from spl

5.read u-boot from usb

6.run u-boot:

```
_start()->arch/mips/cpu/xburst/x1000/start.S
board_init_f()->arch/mips/lib/board.c
board_init_r()->arch/mips/lib/board.c
main_loop()->common/main.c
[
       process_boot_delay(); ->common/main.c
       [
             s = getenv ("bootcmd");#bootcmd = “burn”
             if(bootdelay != -1 && s && !abortboot(bootdelay))
                     run_command_list(s, -1, 0);->common/main.c
                    [
                           builtin_run_command_list(buff, flag);->common/main.c
                           builtin_run_command(line, 0)->common/main.c
                           cmd_process(flag, argc, argv, &repeatable, NULL)->common/command.c
                           do_burn()->common/cmd_burn.c #只做了 usb 的部分初始化工作，没有传输任何数据
                    ]
             else
                    return ;
        ]
        for (;;) {
        len = readline ("halley2" "-sfcnor# ");
        flag = 0;
        if (len > 0)
         strcpy (lastcommand, console_buffer);
        else if (len == 0)
         flag |= 0x0001;
        if (len == -1)
         puts ("<INTERRUPT>\n");
        else
         rc = run_command(lastcommand, flag);
        if (rc <= 0) {
         lastcommand[0] = 0;
        }
        }
]
```

readline()函数实现：

```
int readline_into_buffer(const char *const prompt, char *buffer, int timeout)
{
 char *p = buffer;
 char * p_buf = p;
 int n = 0;
 int plen = 0;
 int col;
 char c;
 if (prompt) {
  plen = strlen (prompt);
  puts (prompt);                #在这里打印命令行输入前缀
 }
 col = plen;
 for (;;) {
#下面红色代码 burner 中有，uboot 中没有
  while (!tstc()) {
   int usb_gadget_handle_interrupts(void);
   usb_gadget_handle_interrupts();#从 boot-args 到所有 policies 的烧录都在这个中断中完成
  }
  c = getc();
  switch (c) {
  case '\r':
  case '\n':
   *p = '\0';
   puts ("\r\n");
   return p - p_buf;
  case '\0':
   continue;
  case 0x03:
   p_buf[0] = '\0';
   return -1;
  case 0x15:
   while (col > plen) {
    puts (erase_seq);
    --col;
   }
   p = p_buf;
   n = 0;
   continue;
  case 0x17:
   p=delete_char(p_buf, p, &col, &n, plen);
   while ((n > 0) && (*p != ' ')) {
    p=delete_char(p_buf, p, &col, &n, plen);
   }
   continue;
  case 0x08:
  case 0x7F:
   p=delete_char(p_buf, p, &col, &n, plen);
   continue;
  default:
   if (n < 1024 -2) {
    if (c == '\t') {
     puts (tab_seq+(col&07));
     col += 8 - (col&07);
    } else {
     char buf[2];
     ++col;
     buf[0] = c;
     buf[1] = '\0';
     puts(buf);
    }
    *p++ = c;
    ++n;
   } else {
    putc ('\a');
   }
  }
 }
}
```

int abortboot(int bootdelay)函数实现：

```
int abortboot_normal(int bootdelay)
{
 int abort = 0;
 unsigned long ts;
 if (bootdelay >= 0)
  printf("Hit any key to stop autoboot: %2d ", bootdelay);
 while ((bootdelay > 0) && (!abort)) {
  --bootdelay;
  ts = get_timer(0);
  do {
   if (tstc()) {
    abort = 1;
    bootdelay = 0;
    (void) getc();
    break;
   }
   udelay(10000);
  } while (!abort && get_timer(ts) < 1000);
  printf("\b\b\b%2d ", bootdelay);#"\b" 在 printf()函数里就是退格的意思，也就是控制光标前移一个字符
 }
 putc('\n');
 return abort;
}
```

usb_gadget_handle_interrupts()函数实现：

```
{
    if (usb_poll_active == true)
        udc_irq();
    return 0;
}
int udc_irq(void)
{
    struct dwc2_udc *dev = the_controller;
    u32 intsts = udc_read_reg(GINT_STS);
    u32 gintmsk = udc_read_reg(GINT_MASK);
    u32 pending = intsts & gintmsk;
    if (pending & GINTSTS_USB_EARLYSUSPEND)
        handle_early_suspend_intr(dev);
    if (pending & GINTSTS_USB_RESET)
        handle_reset_intr(dev);
    if (pending & GINTSTS_ENUM_DONE)
        handle_enum_done_intr(dev);
    if (pending & GINTSTS_IEP_INTR)
        handle_inep_intr(dev);
    if (pending & GINTSTS_OEP_INTR)
        handle_outep_intr(dev);
    if (pending & GINTSTS_RXFIFO_NEMPTY)
        handle_rxfifo_nempty(dev, 0);
    return IRQ_HANDLED;
}

int handle_outep_intr(struct dwc2_udc *dev)
{
        u32 ep_intr, intr;
       u32 ep_msk;
       u32 ep_pending;
        int epnum;
       struct dwc2_ep *dep = NULL;
       for (epnum = 0, intr = (udc_read_reg(OTG_DAINT)&
DAINT_OUT_MASK)>>DAINT_OUT_BIT;
                    intr != 0 && epnum <= DWC2_MAX_OUT_ENDPOINTS;
                    intr &= ~(0x1 << epnum), epnum++) {
               if (!(intr & (0x1 << epnum)))
                    continue;
               else
                    dep = dev->ep_out_attr[epnum];
               ep_intr = udc_read_reg(DOEP_INT(epnum));
               pr_info("===== epnum %d out intr %x ======\n",epnum,ep_intr);
               ep_msk = udc_read_reg(DOEP_MASK);
               ep_pending = (ep_intr&ep_msk);
               if (ep_pending & DEP_XFER_COMP) {
                    if (!epnum) {
                         outep0_transfer_complete(dep);#会调 handle_cmd()
                    } else {
                         outepx_transfer_complete(dep);
                    }
                    udc_write_reg(DEP_XFER_COMP, DOEP_INT(epnum));
               }
               if (ep_pending & DEP_STATUS_PHASE_RECV) {
                    if (!epnum && udc_read_reg(DOEP_INT(epnum)) & DEP_STATUS_PHASE_RECV) {
                         pr_info("DEP_STATUS_PHASE_RECV\n");
                         udc_write_reg(DEP_STATUS_PHASE_RECV, DOEP_INT(0));
```

```
                        }
                                pr_err("back to back received \n");
                                udc_write_reg(DEP_B2B_SETUP_RECV, DOEP_INT(epnum));
                        }
                        parse_setup(dep);#会调 f_cloner_setup_handle()
                }
        }
        return 0;
}


#ifndef CONFIG_BURNER
#include <generated/ddr_reg_values.h>
struct global_info ginfo __attribute__ ((section(".data"))) = {
        .extal          = CONFIG_SYS_EXTAL,
        .cpufreq        = CONFIG_SYS_CPU_FREQ,
        .ddrfreq        = CONFIG_SYS_MEM_FREQ,
        .uart_idx       = CONFIG_SYS_UART_INDEX,
        .baud_rate      = CONFIG_BAUDRATE,

        .ddr_change_param = {
                DDRC_CFG_VALUE,
                DDRC_MMAP0_VALUE,
                DDRC_MMAP1_VALUE,
                DDRC_TIMING4_VALUE,
                DDRC_AUTOSR_EN_VALUE,
                .ddr_remap_array = REMMAP_ARRAY
        }
};
#endif
#define DECLARE_GLOBAL_DATA_PTR     register volatile gd_t *gd asm ("k0")
gd_t gdata __attribute__ ((section(".data")));
void board_init_f(ulong dummy)
{
        /* Set global data pointer */
        gd = &gdata;
        /* Setup global info */
#ifndef CONFIG_BURNER
        gd->arch.gi = &ginfo;
#else
        burner_param_info();#从烧录工具中获得 ginfo
#endif
….
….
….
….
}
```