

廈門大學



软件学院

《实用操作系统》实验报告

题 目 实验五 串口移植

姓 名 陈澄

学 号 32420212202930

班 级 软工三班

实验时间 2023/11/08

2023 年 11 月 08 日

1 实验目的

我们的目标是：让最小系统启动。那么对于串口，不需要考虑得很全面：

- 不需要初始化串口：u-boot 已经初始化串口了
- 不需要动态配置串口：固定使用某个波特率等配置就可以（在 u-boot 里设置过了）

移植工作只需要实现以下几点：

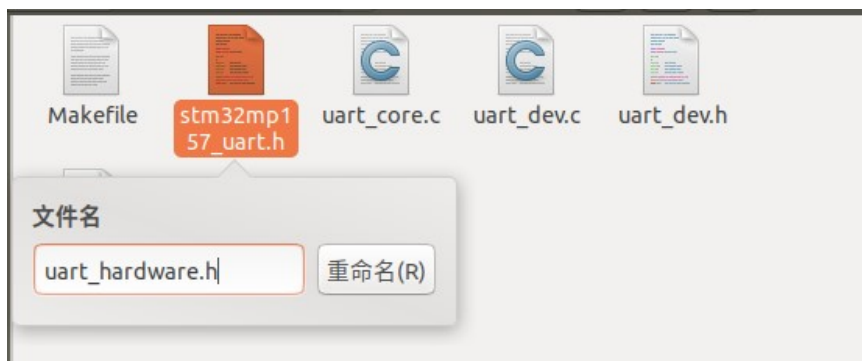
- 串口发送单个字符
- 注册串口接收中断函数：确定中断号、使能中断、在中断函数中读取数据

2 实验步骤

一、修改 Makefile

1. 进入/openharmony/vendor/democom/demochip/driver

/uart(原来为 stm32mp157_uart) 修改文件名如下，另一个 uart_stm32mp157.c 同理



修改同目录下的 makefile 文件

```
Makefile
~/openharmy/vendor/democom/demochip/driver/uart
保存(S)

include $(LITEOSTOPDIR)/config.mk

MODULE_NAME := $(notdir $(shell pwd))

LOCAL_SRCS :=  uart_core.c uart_dev.c uart hardware.c

include $(MODULE)
```

2. 通过 grep 指令搜索 “stm32mp157-uart” 得到需要修改的文件

```
book@100ask:~/openharmy/vendor$ grep "stm32mp157-uart" * -nr
democom/demochip/board/Makefile:7:LOCAL_FLAGS := -I$(LITEOSTOPDIR)/../../vendor/s
t/stm32mp157/driver/stm32mp157-uart
democom/demochip/demochip.mk:18:LIB_SUBDIRS += $(DEMOCHIP_BASE_DIR)/dr
iver/stm32mp157-uart
democom/demochip/demochip.mk:19:LITEOS_BASELIB += -lstm32mp157-uart
st/stm32mp157/board/Makefile:7:LOCAL_FLAGS := -I$(LITEOSTOPDIR)/../../vendor/st/s
tm32mp157/driver/stm32mp157-uart
st/stm32mp157/stm32mp157.mk:18:LIB_SUBDIRS += $(STM32MP157_BASE_DIR)/d
river/stm32mp157-uart
st/stm32mp157/stm32mp157.mk:19:LITEOS_BASELIB += -lstm32mp157-uart
```

进入/openharmy/vendor/democom/demochip/board/Makefile 修改

```
Makefile
~/openharmy/vendor/democom/demochip/board
保存(S)

include $(LITEOSTOPDIR)/config.mk

MODULE_NAME := $(notdir $(shell pwd))

LOCAL_SRCS :=  board.c bsd_board.c

LOCAL_FLAGS := -I$(LITEOSTOPDIR)/../../vendor//democom/demochip/driver/uart

include $(MODULE)
```

进入/openharmy/vendor/democom/demochip/demochip.mk 修改

```
ifeq ($(LOSCFG_DRIVERS_VIDEO), y)
LIB_SUBDIRS += $(DEMOCHIP_BASE_DIR)/driver/stm32mp157-fb
LITEOS_BASELIB += -lstm32mp157-fb
endif

LIB_SUBDIRS += $(DEMOCHIP_BASE_DIR)/driver/uart
LITEOS_BASELIB += -luart
```

二、修改代码

1. 进入/openharmony/vendor/democom/demochip/board/include
/asm/platform.h

注释掉 stm32mp157 所使用的 4 号串口，手动添加一行 imx6ull 使用的
1 号串口

```
#define I2C1_REG_BASE          IO_DEVICE_ADDR(0x12061000)
#define I2C2_REG_BASE          IO_DEVICE_ADDR(0x12062000)

// #define UART4_REG_PBASE      0x40010000 /* stm32mp157 uart4 */
#define UART1_REG_PBASE        0x02020000 /* imx6ull uart1 */

#define UART4_REG_BASE          IO_DEVICE_ADDR(UART4_REG_PBASE)
```

把后面这行注释掉

```
// #define UART0_REG_PBASE      0x12040000
// #define UART1_REG_PBASE      0x12041000
#define UART2_REG_PBASE        0x12042000
```

将物理地址映射为虚拟地址

```
#define UART0_REG_BASE          IO_DEVICE_ADDR(UART0_REG_PBASE)
#define UART1_REG_BASE          IO_DEVICE_ADDR(UART1_REG_PBASE)
#define UART2_REG_BASE          IO_DEVICE_ADDR(0x12042000)
```

注释掉不需要的代码，将 UART0 改为 UART

```
// #if (CONSOLE_UART == UART0)
    #define UART_BASE            UART_REG_BASE
    #define UART0_INT_NUM        NUM_HAL_INTERRUPT_UART
// #elif (CONSOLE_UART == UART1)
    #define UART_BASE            UART1_REG_BASE
    #define UART0_INT_NUM        NUM_HAL_INTERRUPT_UART1
// #elif (CONSOLE_UART == UART2)
    #define UART_BASE            UART2_REG_BASE
    #define UART0_INT_NUM        NUM_HAL_INTERRUPT_UART2
// #endif
```

2. 进入/openharmony/vendor/democom/demochip/board/include

/asm/hal_platform_ints.h

修改串口 1 的中断

```
~  
~  
#define NUM_HAL_INTERRUPT_TIMER3 38  
  
#define NUM_HAL_INTERRUPT_UART0 58  
#define NUM_HAL_INTERRUPT_UART1 (32+26) /* IMX6ULL uart1 */  
#define NUM_HAL_INTERRUPT_UART2 41  
  
#define NUM_HAL_INTERRUPT_UART4 84  
  
#define NUM_HAL_INTERRUPT_GPIO0 48  
#define NUM_HAL_INTERRUPT_GPIO1 49
```

3. 进入/openharmony/vendor/democom/demochip/board

/bsd_board.c

将 uart_add_device 方法内的串口参数全部改为 1 号串口

```
// callback never be null pointer  
static void uart_add_device(add_res_callback_t callback)  
{  
    device_t uart_dev;  
    UART_ADD_DEVICE(uart_dev, 0);  
    callback("uart", SYS_RES_MEMORY, 0, UART1_REG_PBASE,  
            UART1_REG_PBASE + UART_IOMEM_COUNT, UART_IOMEM_COUNT);  
    callback("uart", SYS_RES_IRQ, 0, NUM_HAL_INTERRUPT_UART1,  
            NUM_HAL_INTERRUPT_UART1, 1);  
}  
//#endif
```

4. 进入/openharmony/vendor/democom/demochip/board/include

/uart.h

修改串口中断号的宏为串口 1

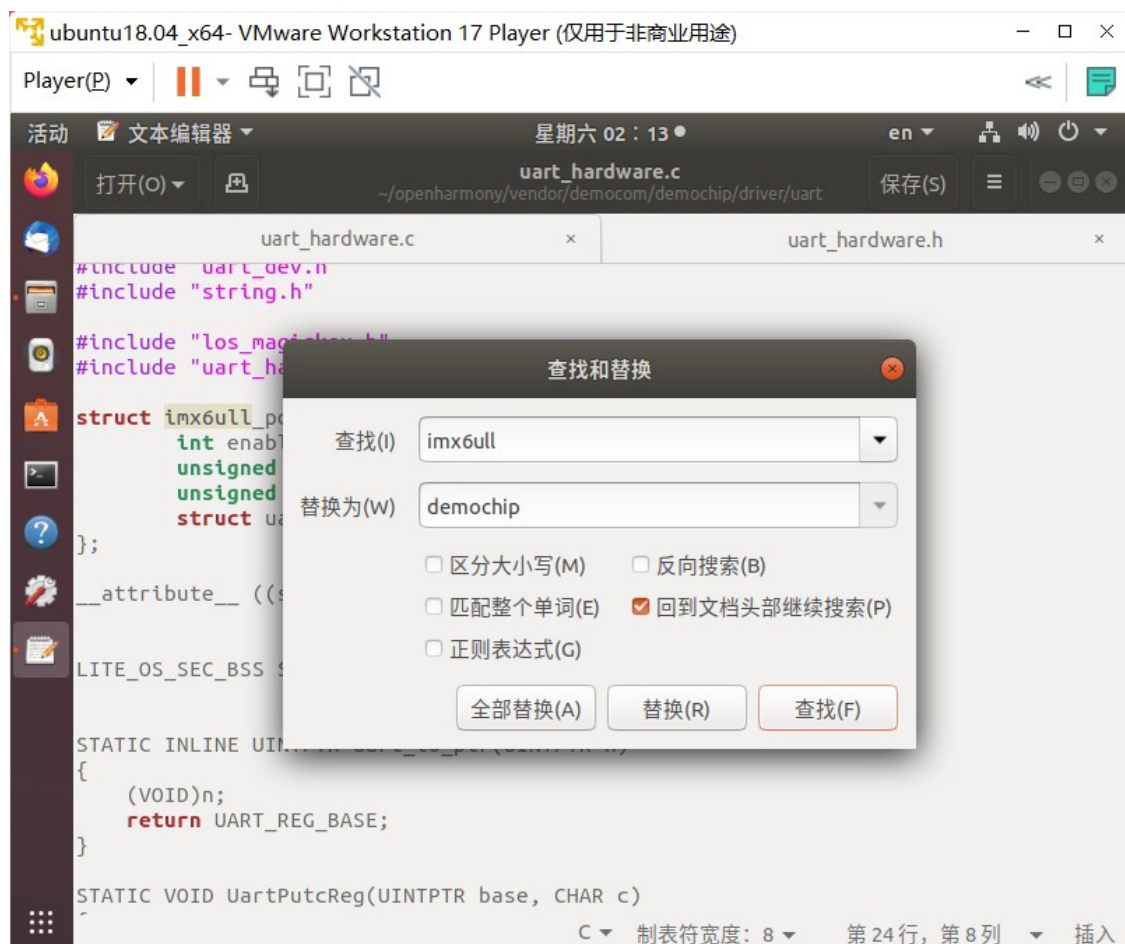
```
#define TTY_DEVICE "/dev/uartdev-0"  
#define UART_REG_BASE UART1_REG_BASE  
#define NUM_HAL_INTERRUPT_UART NUM_HAL_INTERRUPT_UART1 /* TODO,100ask */
```

5. 将 /openharmoney/vendor/nxp/imx6ull/driver/imx6ull-uart 下的 uart_imx6ull.c 和 imx6ull_uart.h 的内容复制到 /openharmoney/vendor/democom/demochip/driver/uart 下的 uart_hardware.c 和 uart_hardware.h

在 uart_hardware.c 中修改引用的头文件

```
#include "los_magickey.h"  
#include "uart_hardware.h"
```

将 uart_hardware.c 的 imx6ull 全部替换为 demochip



三、是否编译通过

1. 成功编译通过

```
ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)
Player(P)  终端
book@100ask: ~/openharmy/kernel/liteos_a
Makefile:158: recipe for target 'liteos' failed
make: *** [liteos] Error 1
book@100ask:~/openharmy/kernel/liteos_a$ make -j 16
make[1]: 进入目录"/home/book/openharmy/kernel/liteos_a/arch/arm/arm"
make[1]: 对"all"无需做任何事。
make[1]: 离开目录"/home/book/openharmy/kernel/liteos_a/arch/arm/arm"
make[1]: 进入目录"/home/book/openharmy/kernel/liteos_a/platform"
make[1]: 离开目录"/home/book/openharmy/kernel/liteos_a/platform"
make[1]: 进入目录"/home/book/openharmy/kernel/liteos_a/kernel/common"
make[1]: 对"all"无需做任何事。
make[1]: 离开目录"/home/book/openharmy/kernel/liteos_a/kernel/common"
make[1]: 进入目录"/home/book/openharmy/kernel/liteos_a/kernel/base"
make[1]: 对"all"无需做任何事。
make[1]: 离开目录"/home/book/openharmy/kernel/liteos_a/kernel/base"
make[1]: 进入目录"/home/book/openharmy/vendor/democom/demochip/board"
make[1]: 离开目录"/home/book/openharmy/vendor/democom/demochip/board"
make[1]: 进入目录"/home/book/openharmy/vendor/democom/demochip/driver/mtd/common"
make[1]: 离开目录"/home/book/openharmy/vendor/democom/demochip/driver/mtd/common"
make[1]: 进入目录"/home/book/openharmy/vendor/democom/demochip/driver/mtd/spi_no
r"
make[1]: 离开目录"/home/book/openharmy/vendor/democom/demochip/driver/mtd/spi_no
r"
make[1]: 进入目录"/home/book/openharmy/vendor/democom/demochip/driver/stm32mp157
-fb"
make[1]: 离开目录"/home/book/openharmy/vendor/democom/demochip/driver/stm32mp157
-fb"
make[1]: 进入目录"/home/book/openharmy/vendor/democom/demochip/driver/usb"
make[1]: 离开目录"/home/book/openharmy/vendor/democom/demochip/driver/usb"
```

```
ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)
Player(P)  终端
book@100ask: ~/openharmy/kernel/liteos_a
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
100_init -uregulator_machine_init -uhisimeidia_regulator_init -ucpufreq_init -uhis
ilicon_cpufreq_init -ucpufreq_machine_init -udevfreq_init -umedia_devfreq_init -ud
evfreq_machine_init -uhieth_machine_init -uhigmac_machine_init -umachine_init -Ma
p=/home/book/openharmy/kernel/liteos_a/out/demochip/liteos.map -o /home/book/ope
nharmy/kernel/liteos_a/out/demochip/liteos --start-group -lclang_rt.builtins -l
unwind --no-dependent-libraries -lcortex-a7 -lbsp -lrootfs -lbase -lboard -lmtl_co
mmon -lspinor_flash -lstm32mp157-fb -luart -lcup -ldynload -lvds -ltickless -lli
teip -lpipes -lc -lsec -lscrew -lc++ -lc++abi -lc++support -lz -lposix -lbsd -lli
nuxkpi -lvfs -lmulti_partition -lbch -lfat -lvirpart -ldisk -lbcache -lramfs -lnfs
-lproc -ljffs2 -llwp --whole-archive -lhdf -lhdf_config -lhello --no-whole-archi
ve -lhievent -lmem -lmtl_common -lvideo -lhilog -lshell -ltelnet -lsyscall -lsecu
rity --end-group
/home/book/llvm/bin/./bin/llvm-objcopy -R .bss -O binary /home/book/openharmy/
kernel/liteos_a/out/demochip/liteos /home/book/openharmy/kernel/liteos_a/out/dem
ochip/liteos.bin
/home/book/llvm/bin/./bin/llvm-objdump -t /home/book/openharmy/kernel/liteos_a
/out/demochip/liteos |sort >/home/book/openharmy/kernel/liteos_a/out/demochip/li
teos.sym.sorted
/home/book/llvm/bin/./bin/llvm-objdump -d /home/book/openharmy/kernel/liteos_a
/out/demochip/liteos >/home/book/openharmy/kernel/liteos_a/out/demochip/liteos.a
sm
make[1]: 进入目录"/home/book/openharmy/kernel/liteos_a/apps"
make[2]: 进入目录"/home/book/openharmy/kernel/liteos_a/apps/shell"
make[2]: 离开目录"/home/book/openharmy/kernel/liteos_a/apps/shell"
make[2]: 进入目录"/home/book/openharmy/kernel/liteos_a/apps/init"
make[2]: 离开目录"/home/book/openharmy/kernel/liteos_a/apps/init"
make[1]: 离开目录"/home/book/openharmy/kernel/liteos_a/apps"
book@100ask:~/openharmy/kernel/liteos_a$
```

2. 测试

(1) 进入/openharmony/kernel/liteos_a/arch/arm/arm/src/startup
/reset_vector_up.S

仿照添加一段 DEMOCHIP 的代码，当启动的是 DEMOCHIP 时输出一个字
符

```
/* Startup code which will get the machine into supervisor mode */
.global reset_vector
.type reset_vector,function
reset_vector:

#if defined(LOSCFG_PLATFORM_DEMOCHIP)
    ldr sp, =0x80000000 + 0x1000000
    mov r0, #'S'
    bl uart_putc_phy
#endif

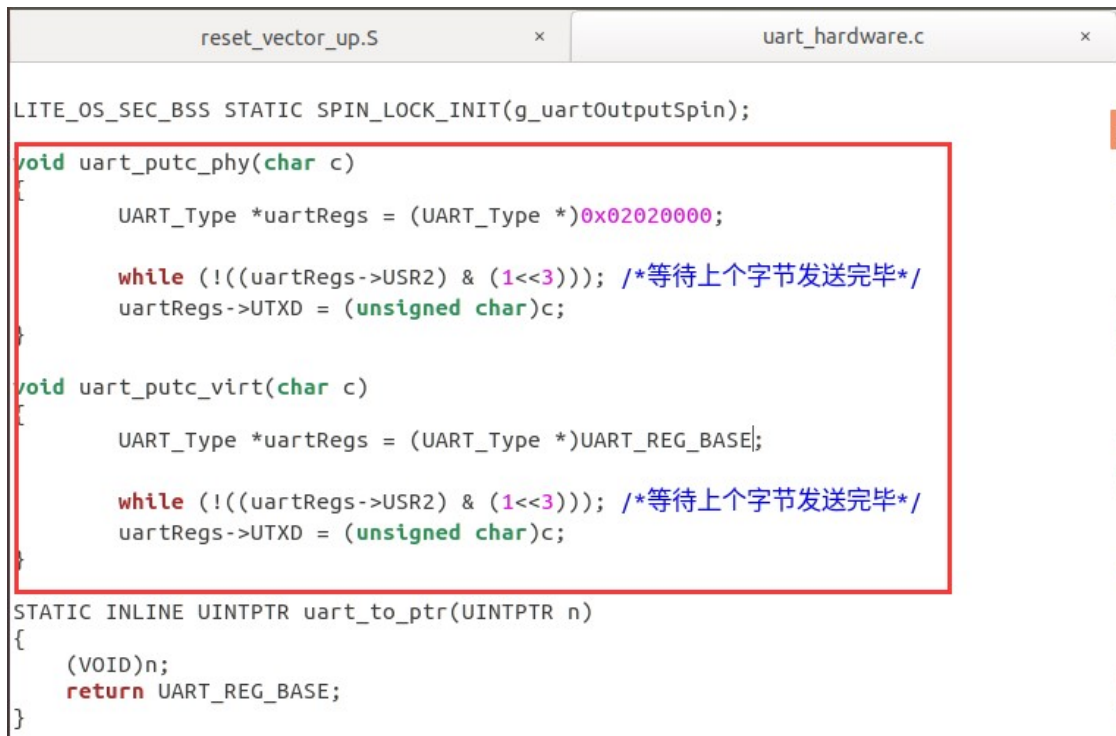
#if defined(LOSCFG_PLATFORM_STM32MP157)
    ldr sp, =0xc0000000 + 0x1000000
    mov r0, #'S'
    bl uart_putc_phy
#endif
#if 1

/* GDB_START - generate a compiled_breadk,This function will get GDB stubs
started, with a proper environment */
    bl GDB_START
    .word 0xe7ffdeff
#endif

#if defined(LOSCFG_PLATFORM_DEMOCHIP)
    mov r0, 'm'
    bl uart_putc_virt
#endif

#if defined(LOSCFG_PLATFORM_STM32MP157)
    mov r0, 'm'
    bl uart_putc_virt
#endif
```

(2) 回到 uart_hardware.c 添加一个方法实现上面的 uart_putc_phy



```
reset_vector_up.S x      uart hardware.c x
LITE_OS_SEC_BSS STATIC SPIN_LOCK_INIT(g_uartOutputSpin);
void uart_putc_phy(char c)
{
    UART_Type *uartRegs = (UART_Type *)0x02020000;

    while (!((uartRegs->USR2) & (1<<3))); /*等待上个字节发送完毕*/
    uartRegs->UTXD = (unsigned char)c;
}

void uart_putc_virt(char c)
{
    UART_Type *uartRegs = (UART_Type *)UART_REG_BASE;

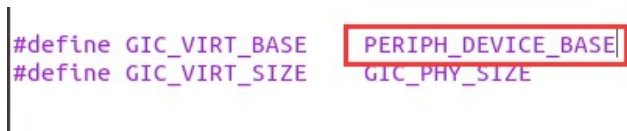
    while (!((uartRegs->USR2) & (1<<3))); /*等待上个字节发送完毕*/
    uartRegs->UTXD = (unsigned char)c;
}

STATIC INLINE UINTPTR uart_to_ptr(UINTPTR n)
{
    (VOID)n;
    return UART_REG_BASE;
}
```

(3) 进入/openharmony/kernel/liteos_a/kernel/base/include

/los_vm_zone.h

修改宏 GIC_VIRT_BASE

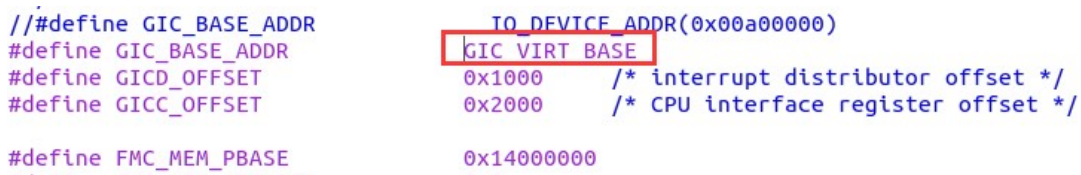


```
#define GIC_VIRT_BASE PERIPH_DEVICE_BASE
#define GIC_VIRT_SIZE GIC_PHY_SIZE
```

(4) 进入/openharmony/vendor/democom/demochip/board/include/asm

/platform.h

修改 GIC_BASE_ADDR 宏

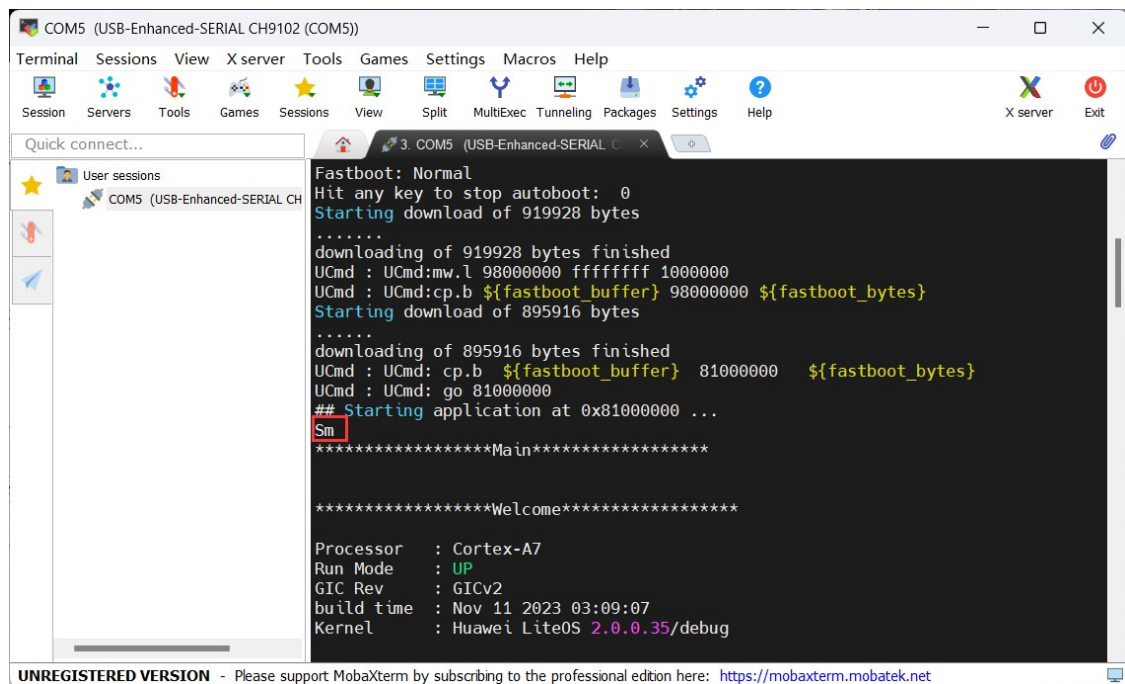


```
TO_DEVICE_ADDR(0x00a00000)
#define GIC_BASE_ADDR GIC_VIRT_BASE
#define GICD_OFFSET 0x1000 /* interrupt distributor offset */
#define GICC_OFFSET 0x2000 /* CPU interface register offset */

#define FMC_MEM_PBASE 0x14000000
```

四、运行结果

成功显示之前调用 `uart_putc_phy` 显示的字符 'S' 以及 `uart_putc_virt` 显示的字符 'm'，也成功进入主函数 `main`，已达到实验要求。



```
COM5 (USB-Enhanced-SERIAL CH9102 (COM5))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
User sessions
COM5 (USB-Enhanced-SERIAL CH

Fastboot: Normal
Hit any key to stop autoboot: 0
Starting download of 919928 bytes
.....
downloading of 919928 bytes finished
UCmd : UCmd:mw.l 98000000 ffffffff 1000000
UCmd : UCmd:cp.b ${fastboot_buffer} 98000000 ${fastboot_bytes}
Starting download of 895916 bytes
.....
downloading of 895916 bytes finished
UCmd : UCmd: cp.b ${fastboot_buffer} 81000000 ${fastboot_bytes}
UCmd : UCmd: go 81000000
## Starting application at 0x81000000 ...
Sm
*****Main*****

*****Welcome*****

Processor   : Cortex-A7
Run Mode   : UP
GIC Rev    : GICv2
build time : Nov 11 2023 03:09:07
Kernel     : Huawei LiteOS 2.0.0.35/debug

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
```

The screenshot shows a MobaXterm terminal window titled "COM5 (USB-Enhanced-SERIAL CH9102 (COM5))". The terminal displays the following output:

```
main core booting up...
cpu 0 entering scheduler
proc fs init ...
Mount procfs finished.
mem dev init ...
spinor_init init ...
src/common/spinor.c spinor_init 155
umx6ull fb init init ...
stm32mp157_lcd.c up_fbinitialize 82
[ERR]0sVmPageFaultHandler 357
#####excFrom: kernel#####!
data_abort fsr:0x805, far:0x00000000
Abort caused by a write instruction. Translation fault, section
excType: data_abort
processName      = KProcess
processID        = 2
process aspace   = 0x40000000 -> 0x60000000
taskName         = SystemInit
taskID           = 4
task kernel stack = 0x40282ce0 -> 0x40286ce0
pc              = 0x40011328
klr              = 0x40011ef4
ksp              = 0x40286c18
fp               = 0x40286cb8
R0               = 0x897c8000
```

At the bottom of the terminal window, a message reads: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

The screenshot shows the same MobaXterm terminal window. The output continues with memory dumps and a memcheck result:

```
0x400d9010 : 00000003 00000000 5851f42d c0b18ccf
0x400d9020 : cbb5f646 c7033129 30705b04 20fd5db4
0x400d9030 : 9a8b7f78 502959d8 ab894868 6c0356a7

dump mem around R11:0x40286cb8
0x40286c78 : 0000001d aaa00000 0000001e aaa00000
0x40286c88 : 0000001f aaa00000 00000000 40000000
0x40286c98 : 80000000 4021cc04 05050505 06060606
0x40286ca8 : 07070707 08080808 09090909 0a0a0a0a
0x40286cb8 : 40286cd8 40008b9c 00000004 01010101
0x40286cc8 : 02020202 03030303 04040404 0a0a0a0a
0x40286cd8 : 0b0b0b0b 4004d958 bfd7931f 00000080
0x40286ce8 : 40282ccc 80000410 e2055007 e1c157b4

dump mem around SP:0x40286c18
0x40286bd8 : 897c8000 00000000 0000f800 00000000
0x40286be8 : 402879cc 400a0dd0 4009d48e 00000000
0x40286bf8 : 400d9000 09090909 0a0a0a0a 40286cb8
0x40286c08 : 00000004 40286c18 40011ef4 40011328
0x40286c18 : 4009d48e 400a0dd0 00000052 aaa00000
0x40286c28 : 00000013 aaa00000 00000014 aaa00000
0x40286c38 : 00000015 aaa00000 00000016 aaa00000
0x40286c48 : 00000017 aaa00000 00000018 aaa00000
system memcheck over, all passed!
```

At the bottom of the terminal window, a message reads: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

3 实验遇到的问题及其解决方法

无

4 我的体会

通过这次实验，我了解了鸿蒙 Liteos 的内核是如何调用的控制台以及如何使用控制台输出单个字符。学会了如何编写方法修改控制台的输出字符，也学会了如何修改串口参数使最小的鸿蒙系统启动。