在kernel初始化阶段的前期，printk是无法输出的，debug string都被buffer起来了。

比如在arch/arm/setup.c中，printk是无法用于debug的。

void __init early_print(const char *str, ...)

```c
void __init early_print(const char *str, ...)
{
        extern void printascii(const char *);
        char buf[256];
        va_list ap;

        va_start(ap, str);
        vsnprintf(buf, sizeof(buf), str, ap);
        va_end(ap);

#ifdef CONFIG_DEBUG_LL
        printascii(buf);
#endif
        printk("%s", buf);
}
```

CONFIG_DEBUG_LL需要enable

CONFIG_DEBUG_LL=y

真正output依赖于void printascii(const char *)

arch/arm/kernel/debug.S

```
ENTRY(printascii)
		addruart_current r3, r1, r2
		b	2f
1:		waituart r2, r3
		senduart r1, r3
		busyuart r2, r3
		teq	r1, #'\n'
		moveq	r1, #'\r'
		beq	1b
2:		teq	r0, #0
		ldrneb	r1, [r0], #1
		teqne	r1, #0
		bne	1b
		ret	lr
ENDPROC(printascii)
```

而uart的具体实现则依赖与platform

在Gr2 and Gs2中

```
CONFIG_DEBUG_LL_UART_8250=y		# 8250兼容UART
CONFIG_DEBUG_LL_INCLUDE="debug/8250.S"
```

printascii中的uart macro的是现在arch/arm/include/debug/8250.S

其中最关键的是

```
        .macro    addruart, rp, rv, tmp
        ldr   \rp, =CONFIG_DEBUG_UART_PHYS
        ldr   \rv, =CONFIG_DEBUG_UART_VIRT
        .endm
```

如果platform的UART是8250兼容的（绝大部分UART应该都是的），那只要提供用于early_print()的UART的地址就可以了（这个肯定是不同的SoC有不通的配置）

在arch/arm/config/pegmatite_defconfig中

```
CONFIG_DEBUG_UART_PHYS=0xd4030000
CONFIG_DEBUG_UART_VIRT=0xfe030000
```

AP::AP_APB::UART1::UART_THR0xD4030000

CONFIG_DEBUG_UART_VIRT是virtual address，怎么定的？

in arch/arm/mach-pegmatite/pegmatite.c

```
DT_MACHINE_START(PEGMATITE_DT, "Marvell Pegmatite (Device Tree)")
#ifdef CONFIG_SMP
    .smp            = smp_ops(pegmatite_smp_ops),
#endif
    .init_machine   = pegmatite_dt_init,
        .map_io          = pegmatite_map_io,
        .init_early = pegmatite_init_early,
        .init_irq    = pegmatite_init_irq,
        .init_time  = pegmatite_timer_and_clk_init,
        .restart     = pegmatite_restart,
        .dt_compat       = pegmatite_dt_compat,
#ifdef CONFIG_ZONE_DMA
        .dma_zone_size      = SZ_256M,
#endif
MACHINE_END


void __init pegmatite_map_io(void)
{
iotable_init(pegmatite_io_desc, ARRAY_SIZE(pegmatite_io_desc));
}


static struct map_desc pegmatite_io_desc[] __initdata = {
        {
                .virtual     = (unsigned long) PEGMATITE_REGS_VIRT_BASE,
                .pfn         = __phys_to_pfn(PEGMATITE_REGS_PHYS_BASE),
                .length            = PEGMATITE_REGS_SIZE,
                .type        = MT_DEVICE,
},
        {
                .virtual     = (unsigned long) PEGMATITE_UPC_VIRT_BASE,
		.pfn		= __phys_to_pfn(PEGMATITE_UPC_PHYS_BASE),
		.length	= 0x000C0000,
		.type		= MT_DEVICE
	},
};
```

in arch/arm/mach-pegmatite/pegmatite.h

```
#define PEGMATITE_REGS_PHYS_BASE          0xd4030000
#define PEGMATITE_REGS_VIRT_BASE             IOMEM(0xfe030000)
#define PEGMATITE_REGS_SIZE                     0x00001000
```

上面0xd4030000是Gr2 / Gs2 SoC的UART1的UART_THR register的physical address,而0xfe030000是该UART的UART_THR register的virtual address。

iotable_init()负责建立这种认为指定的mapping。0xfe030000位于"vmalloc"中。

```
    vmalloc : 0xf0000000 - 0xff000000   ( 240 MB)
```

由于运行pegmatite_map_io()之时，vmalloc空间实际上还是空的，所以认为指定的0xfe030000必然成功。

```
root@granite2:~# cat /proc/dma-mappingsvmallocinfo | grep iotable_init
0xf9800000-0xf98c0000  786432 iotable_init+0x0/0xc phys=f9800000 ioremap
0xfe030000-0xfe031000    4096 iotable_init+0x0/0xc phys=d4030000 ioremap  <-- 这就是early_print()所用到的UART
```

同时，setup_arch() / setup.c

```
              |

              |

             \|/

     paging_init() / arch/arm/mm/mmu.c

              |
              |
             \|/

         devicemaps_init(mdesc)

              3:46
```

有如下code

```
    /*

     * Ask the machine support to map in the statically mapped devices.

     */

    if (mdesc->map_io)

        mdesc->map_io();

    else

        debug_ll_io_init();
```

即如果在custmize的mdesc->map_io function中mapping UART的virtual address，debug_ll_io_init()也会这么做。

in arch/arm/mm/mmu.c

```
#ifdef CONFIG_DEBUG_LL
void __init debug_ll_io_init(void)
{
        struct map_desc map;

        debug_ll_addr(&map.pfn, &map.virtual);
        if (!map.pfn || !map.virtual)
                return;
        map.pfn = __phys_to_pfn(map.pfn);
        map.virtual &= PAGE_MASK;
        map.length = PAGE_SIZE;
        map.type = MT_DEVICE;
        iotable_init(&map, 1);
}
#endif
```

由此看出,early_print()也是在start_kernel() --> setup_arch() --> paging_init() 以后在能工作。有时候这可能还不是足够"early"。

要使得Linux支持early_print，总结如下：

1. config中enable如下value

CONFIG_DEBUG_LL=y

CONFIG_DEBUG_LL_UART_8250=y   # 8250兼容UART

CONFIG_DEBUG_LL_INCLUDE="debug/8250.S"

2. 在machine descriptor的map_io callback function中把UART的physical address mapping to virtual address (virtual address是用户选择的)