U-Boot 系列之一： S3C2410 内存映射结构

作者：yeshi
QQ：251059619
Blog:http://blog.chinaunix.net/u1/33990/

这片文章主要介绍 S3C2410 的内存映射，主要参考 2410 的用户手册。

我用的板子是基于三星的 2410，三星的内存片选有 8 个 bank，这里所谓的 bank 我开始也不是很清楚，在网上搜了一通也不知所云，但是当我看了 2410 的用户手册后才有点明白，这里的 bank 就是片选，一个片选就是一个 bank，在 U-Boot 中，配制的时候要配制 SDRAM 和 FLASH 的 bank 数，那么如果你的 SDRAM 或者 FLASH 就接了一个片选的时候，就定义为 1 就可以了，其他的类推。

下面是 2410 的内存映射图，2410 和其他的大部分的处理器一样，支持 NorFlash 和 NANDFlash 启动，而这两种启动方式内存所映射的地址不怎么相同，我的板子没有 NANDFlash，所以就以从 NorFlash 启动为例子了：

```
==============================================<-------0xFFFF_FFFF
|      NOT USED
==============================================<-------0x6000_0000
|      SFR Area  （各个接口的控制寄存器）
==============================================<-------0x4800_0000
==============================================<-------0x4000_0FFF
|      BootSRAM         (4KBytes)
==============================================<-------0x4000_0000
|      SROM/SDRAM     nGCS7    (bank7)
==============================================<-------0x3800_0000
|      SROM/SDRAM     nGCS6    (bank6)
==============================================<-------0x3000_0000
|      SROM   nGCS5    (bank5)
==============================================<-------0x2800_0000
|      SROM   nGCS4    (bank4)
==============================================<-------0x2000_0000
|      SROM   nGCS3    (bank3)
==============================================<-------0x1800_0000
|      SROM   nGCS2    (bank2)
==============================================<-------0x1000_0000
|      SROM   nGCS1    (bank1)
==============================================<-------0x0800_0000
|      SROM   nGCS0    (bank0)
==============================================<-------0x0000_0000
```

上面的每个 bank 最大支持 128M，除了 bank0，其它的每个 bank 都支持 8/16/32 位操作，bank0 只支

持 16/32 位操作，在我的板子上，Flash 接 bank0，一共 8M，16 位，SDRAM 接 bank6，一个 32M，32 位。所以启动的时候，CPU 从 0x0000_0000 开始执行，而在这上面的 NorFlash，存放的是 U-Boot，用于启动 Linux 的。

而我们目前所做的工作就是要把 U-Boot 移植成功，然后烧写到该地址，假设你的板子的供应商已经提供了少些的工具，等到我们少些成功，我们就可以通过网络来加载 Linux，通过串口来调试了，至于怎么做的，我们慢慢来，以后的帖子会涉及到。。。

在上面的存储地址布局中，SFR Area 就是各个接口控制器的地址，我们可以把它定义如下，来自华恒 PPCboot 上面的源代码中的头文件 s3c2410.h：

```
/* Memory control */
#define rBWSCON          (*(volatile unsigned *)0x48000000)
#define rBANKCON0        (*(volatile unsigned *)0x48000004)
#define rBANKCON1        (*(volatile unsigned *)0x48000008)
#define rBANKCON2        (*(volatile unsigned *)0x4800000C)
#define rBANKCON3        (*(volatile unsigned *)0x48000010)
#define rBANKCON4        (*(volatile unsigned *)0x48000014)
#define rBANKCON5        (*(volatile unsigned *)0x48000018)
#define rBANKCON6        (*(volatile unsigned *)0x4800001C)
#define rBANKCON7        (*(volatile unsigned *)0x48000020)
#define rREFRESH         (*(volatile unsigned *)0x48000024)
#define rBANKSIZE        (*(volatile unsigned *)0x48000028)
#define rMRSRB6          (*(volatile unsigned *)0x4800002C)
#define rMRSRB7          (*(volatile unsigned *)0x48000030)


/* USB HOST */
#define rHcRevision            (*(volatile unsigned *)0x49000000)
#define rHcControl             (*(volatile unsigned *)0x49000004)
#define rHcCommonStatus        (*(volatile unsigned *)0x49000008)
#define rHcInterruptStatus     (*(volatile unsigned *)0x4900000C)
#define rHcInterruptEnable     (*(volatile unsigned *)0x49000010)
#define rHcInterruptDisable    (*(volatile unsigned *)0x49000014)
#define rHcHCCA                (*(volatile unsigned *)0x49000018)
#define rHcPeriodCuttendED     (*(volatile unsigned *)0x4900001C)
#define rHcControlHeadED       (*(volatile unsigned *)0x49000020)
#define rHcControlCurrentED    (*(volatile unsigned *)0x49000024)
#define rHcBulkHeadED          (*(volatile unsigned *)0x49000028)
#define rHcBuldCurrentED       (*(volatile unsigned *)0x4900002C)
#define rHcDoneHead            (*(volatile unsigned *)0x49000030)
#define rHcRmInterval          (*(volatile unsigned *)0x49000034)
#define rHcFmRemaining         (*(volatile unsigned *)0x49000038)
#define rHcFmNumber            (*(volatile unsigned *)0x4900003C)
#define rHcPeriodicStart       (*(volatile unsigned *)0x49000040)
#define rHcLSThreshold         (*(volatile unsigned *)0x49000044)
```

```
#define rHcRhDescriptorA       (*(volatile unsigned *)0x49000048)
#define rHcRhDescriptorB       (*(volatile unsigned *)0x4900004C)
#define rHcRhStatus            (*(volatile unsigned *)0x49000050)
#define rHcRhPortStatus1       (*(volatile unsigned *)0x49000054)
#define rHcRhPortStatus2       (*(volatile unsigned *)0x49000058)


/* INTERRUPT */
#define rSRCPND            (*(volatile unsigned *)0x4A000000)
#define rINTMOD            (*(volatile unsigned *)0x4A000004)
#define rINTMSK            (*(volatile unsigned *)0x4A000008)
#define rPRIORITY          (*(volatile unsigned *)0x4A00000C)
#define rINTPND            (*(volatile unsigned *)0x4A000010)
#define rINTOFFSET         (*(volatile unsigned *)0x4A000014)
#define rSUBSRCPND         (*(volatile unsigned *)0x4A000018)
#define rINTSUBMSK         (*(volatile unsigned *)0x4A00001C)


/* DMA */
#define rDISRC0            (*(volatile unsigned *)0x4B000000)
#define rDISRCC0           (*(volatile unsigned *)0x4B000004)
#define rDIDST0            (*(volatile unsigned *)0x4B000008)
#define rDIDSTC0           (*(volatile unsigned *)0x4B00000C)
#define rDCON0             (*(volatile unsigned *)0x4B000010)
#define rDSTAT0            (*(volatile unsigned *)0x4B000014)
#define rDCSRC0            (*(volatile unsigned *)0x4B000018)
#define rDCDST0            (*(volatile unsigned *)0x4B00001C)
#define rDMASKTRIG0        (*(volatile unsigned *)0x4B000020)
#define rDISRC1            (*(volatile unsigned *)0x4B000040)
#define rDISRCC1           (*(volatile unsigned *)0x4B000044)
#define rDIDST1            (*(volatile unsigned *)0x4B000048)
#define rDIDSTC1           (*(volatile unsigned *)0x4B00004C)
#define rDCON1             (*(volatile unsigned *)0x4B000050)
#define rDSTAT1            (*(volatile unsigned *)0x4B000054)
#define rDCSRC1            (*(volatile unsigned *)0x4B000058)
#define rDCDST1            (*(volatile unsigned *)0x4B00005C)
#define rDMASKTRIG1        (*(volatile unsigned *)0x4B000060)
#define rDISRC2            (*(volatile unsigned *)0x4B000080)
#define rDISRCC2           (*(volatile unsigned *)0x4B000084)
#define rDIDST2            (*(volatile unsigned *)0x4B000088)
#define rDIDSTC2           (*(volatile unsigned *)0x4B00008C)
#define rDCON2             (*(volatile unsigned *)0x4B000090)
#define rDSTAT2            (*(volatile unsigned *)0x4B000094)
#define rDCSRC2            (*(volatile unsigned *)0x4B000098)
```

```
#define rDCDST2            (*(volatile unsigned *)0x4B00009C)
#define rDMASKTRIG2        (*(volatile unsigned *)0x4B0000A0)
#define rDISRC3            (*(volatile unsigned *)0x4B0000C0)
#define rDISRCC3           (*(volatile unsigned *)0x4B0000C4)
#define rDIDST3            (*(volatile unsigned *)0x4B0000C8)
#define rDIDSTC3           (*(volatile unsigned *)0x4B0000CC)
#define rDCON3             (*(volatile unsigned *)0x4B0000D0)
#define rDSTAT3            (*(volatile unsigned *)0x4B0000D4)
#define rDCSRC3            (*(volatile unsigned *)0x4B0000D8)
#define rDCDST3            (*(volatile unsigned *)0x4B0000DC)
#define rDMASKTRIG3        (*(volatile unsigned *)0x4B0000E0)


/* CLOCK & POWER MANAGEMENT */
#define rLOCKTIME        (*(volatile unsigned *)0x4C000000)
#define rMPLLCON         (*(volatile unsigned *)0x4C000004)
#define rUPLLCON         (*(volatile unsigned *)0x4C000008)
#define rCLKCON          (*(volatile unsigned *)0x4C00000C)
#define rCLKSLOW         (*(volatile unsigned *)0x4C000010)
#define rCLKDIVN         (*(volatile unsigned *)0x4C000014)


/* LCD CONTROLLER */
#define rLCDCON1        (*(volatile unsigned *)0x4D000000)
#define rLCDCON2        (*(volatile unsigned *)0x4D000004)
#define rLCDCON3        (*(volatile unsigned *)0x4D000008)
#define rLCDCON4        (*(volatile unsigned *)0x4D00000C)
#define rLCDCON5        (*(volatile unsigned *)0x4D000010)
#define rLCDSADDR1        (*(volatile unsigned *)0x4D000014)
#define rLCDSADDR2        (*(volatile unsigned *)0x4D000018)
#define rLCDSADDR3        (*(volatile unsigned *)0x4D00001C)
#define rREDLUT           (*(volatile unsigned *)0x4D000020)
#define rGREENLUT         (*(volatile unsigned *)0x4D000024)
#define rBLUELUT          (*(volatile unsigned *)0x4D000028)
#define rDITHMODE         (*(volatile unsigned *)0x4D00004C)
#define rTPAL             (*(volatile unsigned *)0x4D000050)
#define rLCDINTPND        (*(volatile unsigned *)0x4D000054)
#define rLCDSRCPND        (*(volatile unsigned *)0x4D000058)
#define rLCDINTMSK        (*(volatile unsigned *)0x4D00005C)


/* NAND FLASH */
#define rNFCONF            (*(volatile unsigned *)0x4E000000)
#define rNFCMD             (*(volatile unsigned *)0x4E000004)
```

```
#define rNFADDR             (*(volatile unsigned *)0x4E000008)
#define rNFDATA             (*(volatile unsigned *)0x4E00000C)
#define rNFSTAT             (*(volatile unsigned *)0x4E000010)
#define rNFECC              (*(volatile unsigned *)0x4E000014)


/* UART */
#define rULCON0             (*(volatile unsigned *)0x50000000)
#define rUCON0              (*(volatile unsigned *)0x50000004)
#define rUFCON0             (*(volatile unsigned *)0x50000008)
#define rUMCON0             (*(volatile unsigned *)0x5000000C)
#define rUTRSTAT0           (*(volatile unsigned *)0x50000010)
#define rUERSTAT0           (*(volatile unsigned *)0x50000014)
#define rUFSTAT0            (*(volatile unsigned *)0x50000018)
#define rUMSTAT0            (*(volatile unsigned *)0x5000001C)
#define rUBRDIV0            (*(volatile unsigned *)0x50000028)

#define rULCON1             (*(volatile unsigned *)0x50004000)
#define rUCON1              (*(volatile unsigned *)0x50004004)
#define rUFCON1             (*(volatile unsigned *)0x50004008)
#define rUMCON1             (*(volatile unsigned *)0x5000400C)
#define rUTRSTAT1           (*(volatile unsigned *)0x50004010)
#define rUERSTAT1           (*(volatile unsigned *)0x50004014)
#define rUFSTAT1         (*(volatile unsigned *)0x50004018)
#define rUMSTAT1         (*(volatile unsigned *)0x5000401C)
#define rUBRDIV1         (*(volatile unsigned *)0x50004028)

#define rULCON2             (*(volatile unsigned *)0x50008000)
#define rUCON2              (*(volatile unsigned *)0x50008004)
#define rUFCON2             (*(volatile unsigned *)0x50008008)
#define rUTRSTAT2         (*(volatile unsigned *)0x50008010)
#define rUERSTAT2         (*(volatile unsigned *)0x50008014)
#define rUFSTAT2          (*(volatile unsigned *)0x50008018)
#define rUBRDIV2          (*(volatile unsigned *)0x50008028)

#ifdef __BIG_ENDIAN
#define rUTXH0             (*(volatile unsigned char *)0x50000023)
#define rURXH0             (*(volatile unsigned char *)0x50000027)
#define rUTXH1             (*(volatile unsigned char *)0x50004023)
#define rURXH1             (*(volatile unsigned char *)0x50004027)
#define rUTXH2             (*(volatile unsigned char *)0x50008023)
#define rURXH2             (*(volatile unsigned char *)0x50008027)

#define WrUTXH0(ch)         (*(volatile unsigned char *)0x50000023)=(unsigned char)(ch)
```

```
#define RdURXH0()           (*(volatile unsigned char *)0x50000027)
#define WrUTXH1(ch)         (*(volatile unsigned char *)0x50004023)=(unsigned char)(ch)
#define RdURXH1()           (*(volatile unsigned char *)0x50004027)
#define WrUTXH2(ch)         (*(volatile unsigned char *)0x50008023)=(unsigned char)(ch)
#define RdURXH2()           (*(volatile unsigned char *)0x50008027)

#define UTXH0             (0x50000020+3)  /* byte_access address by DMA */
#define URXH0             (0x50000024+3)
#define UTXH1             (0x50004020+3)
#define URXH1             (0x50004024+3)
#define UTXH2             (0x50008020+3)
#define URXH2             (0x50008024+3)

#else /* Little Endian */
#define rUTXH0             (*(volatile unsigned char *)0x50000020)
#define rURXH0             (*(volatile unsigned char *)0x50000024)
#define rUTXH1             (*(volatile unsigned char *)0x50004020)
#define rURXH1             (*(volatile unsigned char *)0x50004024)
#define rUTXH2             (*(volatile unsigned char *)0x50008020)
#define rURXH2             (*(volatile unsigned char *)0x50008024)

#define WrUTXH0(ch)         (*(volatile unsigned char *)0x50000020)=(unsigned char)(ch)
#define RdURXH0()           (*(volatile unsigned char *)0x50000024)
#define WrUTXH1(ch)         (*(volatile unsigned char *)0x50004020)=(unsigned char)(ch)
#define RdURXH1()           (*(volatile unsigned char *)0x50004024)
#define WrUTXH2(ch)         (*(volatile unsigned char *)0x50008020)=(unsigned char)(ch)
#define RdURXH2()           (*(volatile unsigned char *)0x50008024)

#define UTXH0             (0x50000020)    /* byte_access address by DMA */
#define URXH0             (0x50000024)
#define UTXH1             (0x50004020)
#define URXH1             (0x50004024)
#define UTXH2             (0x50008020)
#define URXH2             (0x50008024)
#endif


/* PWM TIMER */
#define rTCFG0              (*(volatile unsigned *)0x51000000)
#define rTCFG1              (*(volatile unsigned *)0x51000004)
#define rTCON               (*(volatile unsigned *)0x51000008)
#define rTCNTB0             (*(volatile unsigned *)0x5100000C)
#define rTCMPB0             (*(volatile unsigned *)0x51000010)
#define rTCNTO0             (*(volatile unsigned *)0x51000014)
```

```
#define rTCNTB1            (*(volatile unsigned *)0x51000018)
#define rTCMPB1            (*(volatile unsigned *)0x5100001C)
#define rTCNTO1            (*(volatile unsigned *)0x51000020)
#define rTCNTB2            (*(volatile unsigned *)0x51000024)
#define rTCMPB2            (*(volatile unsigned *)0x51000028)
#define rTCNTO2            (*(volatile unsigned *)0x5100002C)
#define rTCNTB3            (*(volatile unsigned *)0x51000030)
#define rTCMPB3            (*(volatile unsigned *)0x51000034)
#define rTCNTO3            (*(volatile unsigned *)0x51000038)
#define rTCNTB4            (*(volatile unsigned *)0x5100003C)
#define rTCNTO4            (*(volatile unsigned *)0x51000040)


/* USB DEVICE */
#ifdef __BIG_ENDIAN
#define rFUNC_ADDR_REG        (*(volatile unsigned char *)0x52000143)
#define rPWR_REG              (*(volatile unsigned char *)0x52000147)
#define rEP_INT_REG           (*(volatile unsigned char *)0x5200014B)
#define rUSB_INT_REG          (*(volatile unsigned char *)0x5200015B)
#define rEP_INT_EN_REG        (*(volatile unsigned char *)0x5200015F)
#define rUSB_INT_EN_REG        (*(volatile unsigned char *)0x5200016F)
#define rFRAME_NUM1_REG        (*(volatile unsigned char *)0x52000173)
#define rFRAME_NUM2_REG        (*(volatile unsigned char *)0x52000177)
#define rINDEX_REG        (*(volatile unsigned char *)0x5200017B)
#define rMAXP_REG         (*(volatile unsigned char *)0x52000183)
#define rEP0_CSR          (*(volatile unsigned char *)0x52000187)
#define rIN_CSR1_REG         (*(volatile unsigned char *)0x52000187)
#define rIN_CSR2_REG         (*(volatile unsigned char *)0x5200018B)
#define rOUT_CSR1_REG        (*(volatile unsigned char *)0x52000193)
#define rOUT_CSR2_REG        (*(volatile unsigned char *)0x52000197)
#define rOUT_FIFO_CNT1_REG    (*(volatile unsigned char *)0x5200019B)
#define rOUT_FIFO_CNT2_REG    (*(volatile unsigned char *)0x5200019F)
#define rEP0_FIFO        (*(volatile unsigned char *)0x520001C3)
#define rEP1_FIFO        (*(volatile unsigned char *)0x520001C7)
#define rEP2_FIFO        (*(volatile unsigned char *)0x520001CB)
#define rEP3_FIFO        (*(volatile unsigned char *)0x520001CF)
#define rEP4_FIFO        (*(volatile unsigned char *)0x520001D3)
#define rEP1_DMA_CON          (*(volatile unsigned char *)0x52000203)
#define rEP1_DMA_UNIT         (*(volatile unsigned char *)0x52000207)
#define rEP1_DMA_FIFO         (*(volatile unsigned char *)0x5200020B)
#define rEP1_DMA_TX_LO        (*(volatile unsigned char *)0x5200020F)
#define rEP1_DMA_TX_MD        (*(volatile unsigned char *)0x52000213)
#define rEP1_DMA_TX_HI        (*(volatile unsigned char *)0x52000217)
#define rEP2_DMA_CON          (*(volatile unsigned char *)0x5200021B)
```

```
#define rEP2_DMA_UNIT         (*(volatile unsigned char *)0x5200021F)
#define rEP2_DMA_FIFO         (*(volatile unsigned char *)0x52000223)
#define rEP2_DMA_TX_LO        (*(volatile unsigned char *)0x52000227)
#define rEP2_DMA_TX_MD        (*(volatile unsigned char *)0x5200022B)
#define rEP2_DMA_TX_HI        (*(volatile unsigned char *)0x5200022F)
#define rEP3_DMA_CON          (*(volatile unsigned char *)0x52000243)
#define rEP3_DMA_UNIT         (*(volatile unsigned char *)0x52000247)
#define rEP3_DMA_FIFO         (*(volatile unsigned char *)0x5200024B)
#define rEP3_DMA_TX_LO        (*(volatile unsigned char *)0x5200024F)
#define rEP3_DMA_TX_MD        (*(volatile unsigned char *)0x52000253)
#define rEP3_DMA_TX_HI        (*(volatile unsigned char *)0x52000257)
#define rEP4_DMA_CON          (*(volatile unsigned char *)0x5200025B)
#define rEP4_DMA_UNIT         (*(volatile unsigned char *)0x5200025F)
#define rEP4_DMA_FIFO         (*(volatile unsigned char *)0x52000263)
#define rEP4_DMA_TX_LO        (*(volatile unsigned char *)0x52000267)
#define rEP4_DMA_TX_MD        (*(volatile unsigned char *)0x5200026B)
#define rEP4_DMA_TX_HI        (*(volatile unsigned char *)0x5200026F)
#else /*  little endian */
#define rFUNC_ADDR_REG        (*(volatile unsigned char *)0x52000140)
#define rPWR_REG              (*(volatile unsigned char *)0x52000144)
#define rEP_INT_REG           (*(volatile unsigned char *)0x52000148)
#define rUSB_INT_REG          (*(volatile unsigned char *)0x52000158)
#define rEP_INT_EN_REG        (*(volatile unsigned char *)0x5200015C)
#define rUSB_INT_EN_REG        (*(volatile unsigned char *)0x5200016C)
#define rFRAME_NUM1_REG        (*(volatile unsigned char *)0x52000170)
#define rFRAME_NUM2_REG        (*(volatile unsigned char *)0x52000174)
#define rINDEX_REG        (*(volatile unsigned char *)0x52000178)
#define rMAXP_REG         (*(volatile unsigned char *)0x52000180)
#define rEP0_CSR          (*(volatile unsigned char *)0x52000184)
#define rIN_CSR1_REG        (*(volatile unsigned char *)0x52000184)
#define rIN_CSR2_REG        (*(volatile unsigned char *)0x52000188)
#define rOUT_CSR1_REG        (*(volatile unsigned char *)0x52000190)
#define rOUT_CSR2_REG        (*(volatile unsigned char *)0x52000194)
#define rOUT_FIFO_CNT1_REG    (*(volatile unsigned char *)0x52000198)
#define rOUT_FIFO_CNT2_REG    (*(volatile unsigned char *)0x5200019C)
#define rEP0_FIFO        (*(volatile unsigned char *)0x520001C0)
#define rEP1_FIFO        (*(volatile unsigned char *)0x520001C4)
#define rEP2_FIFO        (*(volatile unsigned char *)0x520001C8)
#define rEP3_FIFO        (*(volatile unsigned char *)0x520001CC)
#define rEP4_FIFO        (*(volatile unsigned char *)0x520001D0)
#define rEP1_DMA_CON         (*(volatile unsigned char *)0x52000200)
#define rEP1_DMA_UNIT        (*(volatile unsigned char *)0x52000204)
#define rEP1_DMA_FIFO        (*(volatile unsigned char *)0x52000208)
#define rEP1_DMA_TX_LO        (*(volatile unsigned char *)0x5200020C)
```

```
#define rEP1_DMA_TX_MD        (*(volatile unsigned char *)0x52000210)
#define rEP1_DMA_TX_HI        (*(volatile unsigned char *)0x52000214)
#define rEP2_DMA_CON         (*(volatile unsigned char *)0x52000218)
#define rEP2_DMA_UNIT        (*(volatile unsigned char *)0x5200021C)
#define rEP2_DMA_FIFO        (*(volatile unsigned char *)0x52000220)
#define rEP2_DMA_TX_LO        (*(volatile unsigned char *)0x52000224)
#define rEP2_DMA_TX_MD        (*(volatile unsigned char *)0x52000228)
#define rEP2_DMA_TX_HI        (*(volatile unsigned char *)0x5200022C)
#define rEP3_DMA_CON         (*(volatile unsigned char *)0x52000240)
#define rEP3_DMA_UNIT        (*(volatile unsigned char *)0x52000244)
#define rEP3_DMA_FIFO        (*(volatile unsigned char *)0x52000248)
#define rEP3_DMA_TX_LO        (*(volatile unsigned char *)0x5200024C)
#define rEP3_DMA_TX_MD        (*(volatile unsigned char *)0x52000250)
#define rEP3_DMA_TX_HI        (*(volatile unsigned char *)0x52000254)
#define rEP4_DMA_CON         (*(volatile unsigned char *)0x52000258)
#define rEP4_DMA_UNIT        (*(volatile unsigned char *)0x5200025C)
#define rEP4_DMA_FIFO        (*(volatile unsigned char *)0x52000260)
#define rEP4_DMA_TX_LO        (*(volatile unsigned char *)0x52000264)
#define rEP4_DMA_TX_MD        (*(volatile unsigned char *)0x52000268)
#define rEP4_DMA_TX_HI        (*(volatile unsigned char *)0x5200026C)
#endif /*  __BIG_ENDIAN */

/* WATCH DOG TIMER */
#define rWTCON            (*(volatile unsigned *)0x53000000)
#define rWTDAT            (*(volatile unsigned *)0x53000004)
#define rWTCNT            (*(volatile unsigned *)0x53000008)



/* IIC */
#define rIICCON            (*(volatile unsigned *)0x54000000)
#define rIICSTAT           (*(volatile unsigned *)0x54000004)
#define rIICADD            (*(volatile unsigned *)0x54000008)
#define rIICDS             (*(volatile unsigned *)0x5400000C)



/* IIS */
#define rIISCON            (*(volatile unsigned *)0x55000000)
#define rIISMOD            (*(volatile unsigned *)0x55000004)
#define rIISPSR            (*(volatile unsigned *)0x55000008)
#define rIISFCON        (*(volatile unsigned *)0x5500000C)

#ifdef __BIG_ENDIAN
#define IISFIF            ((volatile unsigned short *)0x55000012)
#else /*  little endian */
```

```
#define IISFIF           ((volatile unsigned short *)0x55000010)
#endif


/* I/O PORT */
#define rGPACON           (*(volatile unsigned *)0x56000000)
#define rGPADAT           (*(volatile unsigned *)0x56000004)

#define rGPBCON           (*(volatile unsigned *)0x56000010)
#define rGPBDAT           (*(volatile unsigned *)0x56000014)
#define rGPBUP           (*(volatile unsigned *)0x56000018)

#define rGPCCON           (*(volatile unsigned *)0x56000020)
#define rGPCDAT           (*(volatile unsigned *)0x56000024)
#define rGPCUP           (*(volatile unsigned *)0x56000028)

#define rGPDCON           (*(volatile unsigned *)0x56000030)
#define rGPDDAT           (*(volatile unsigned *)0x56000034)
#define rGPDUP           (*(volatile unsigned *)0x56000038)

#define rGPECON           (*(volatile unsigned *)0x56000040)
#define rGPEDAT           (*(volatile unsigned *)0x56000044)
#define rGPEUP           (*(volatile unsigned *)0x56000048)

#define rGPFCON           (*(volatile unsigned *)0x56000050)
#define rGPFDAT           (*(volatile unsigned *)0x56000054)
#define rGPFUP           (*(volatile unsigned *)0x56000058)

#define rGPGCON           (*(volatile unsigned *)0x56000060)
#define rGPGDAT           (*(volatile unsigned *)0x56000064)
#define rGPGUP           (*(volatile unsigned *)0x56000068)

#define rGPHCON           (*(volatile unsigned *)0x56000070)
#define rGPHDAT           (*(volatile unsigned *)0x56000074)
#define rGPHUP           (*(volatile unsigned *)0x56000078)

#define rMISCCR           (*(volatile unsigned *)0x56000080)
#define rDCLKCON        (*(volatile unsigned *)0x56000084)
#define rEXTINT0        (*(volatile unsigned *)0x56000088)
#define rEXTINT1        (*(volatile unsigned *)0x5600008C)
#define rEXTINT2        (*(volatile unsigned *)0x56000090)
#define rEINTFLT0        (*(volatile unsigned *)0x56000094)
#define rEINTFLT1        (*(volatile unsigned *)0x56000098)
#define rEINTFLT2        (*(volatile unsigned *)0x5600009C)
```

```
#define rEINTFLT3        (*(volatile unsigned *)0x560000A0)
#define rEINTMASK        (*(volatile unsigned *)0x560000A4)
#define rEINTPEND        (*(volatile unsigned *)0x560000A8)
#define rGSTATUS0        (*(volatile unsigned *)0x560000AC)
#define rGSTATUS1        (*(volatile unsigned *)0x560000B0)


/* RTC */
#ifdef __BIG_ENDIAN
#define rRTCCON            (*(volatile unsigned char *)0x57000043)
#define rTICNT            (*(volatile unsigned char *)0x57000047)
#define rRTCALM            (*(volatile unsigned char *)0x57000053)
#define rALMSEC            (*(volatile unsigned char *)0x57000057)
#define rALMMIN            (*(volatile unsigned char *)0x5700005B)
#define rALMHOUR        (*(volatile unsigned char *)0x5700005F)
#define rALMDATE        (*(volatile unsigned char *)0x57000063)
#define rALMMON            (*(volatile unsigned char *)0x57000067)
#define rALMYEAR        (*(volatile unsigned char *)0x5700006B)
#define rRTCRST            (*(volatile unsigned char *)0x5700006F)
#define rBCDSEC            (*(volatile unsigned char *)0x57000073)
#define rBCDMIN            (*(volatile unsigned char *)0x57000077)
#define rBCDHOUR        (*(volatile unsigned char *)0x5700007B)
#define rBCDDATE        (*(volatile unsigned char *)0x5700007F)
#define rBCDDAY            (*(volatile unsigned char *)0x57000083)
#define rBCDMON            (*(volatile unsigned char *)0x57000087)
#define rBCDYEAR        (*(volatile unsigned char *)0x5700008B)
#else /*  little endian */
#define rRTCCON            (*(volatile unsigned char *)0x57000040)
#define rTICNT            (*(volatile unsigned char *)0x57000044)
#define rRTCALM            (*(volatile unsigned char *)0x57000050)
#define rALMSEC            (*(volatile unsigned char *)0x57000054)
#define rALMMIN            (*(volatile unsigned char *)0x57000058)
#define rALMHOUR        (*(volatile unsigned char *)0x5700005C)
#define rALMDATE        (*(volatile unsigned char *)0x57000060)
#define rALMMON            (*(volatile unsigned char *)0x57000064)
#define rALMYEAR        (*(volatile unsigned char *)0x57000068)
#define rRTCRST            (*(volatile unsigned char *)0x5700006C)
#define rBCDSEC            (*(volatile unsigned char *)0x57000070)
#define rBCDMIN            (*(volatile unsigned char *)0x57000074)
#define rBCDHOUR        (*(volatile unsigned char *)0x57000078)
#define rBCDDATE        (*(volatile unsigned char *)0x5700007C)
#define rBCDDAY            (*(volatile unsigned char *)0x57000080)
#define rBCDMON            (*(volatile unsigned char *)0x57000084)
#define rBCDYEAR        (*(volatile unsigned char *)0x57000088)
```

```
#endif


/* ADC */
#define rADCCON            (*(volatile unsigned *)0x58000000)
#define rADCTSC            (*(volatile unsigned *)0x58000004)
#define rADCDLY            (*(volatile unsigned *)0x58000008)
#define rADCDAT0        (*(volatile unsigned *)0x5800000C)
#define rADCDAT1        (*(volatile unsigned *)0x58000010)


/* SPI */
#define rSPCON0            (*(volatile unsigned *)0x59000000)
#define rSPSTA0            (*(volatile unsigned *)0x59000004)
#define rSPPIN0            (*(volatile unsigned *)0x59000008)
#define rSPPRE0            (*(volatile unsigned *)0x5900000C)
#define rSPTDAT0        (*(volatile unsigned *)0x59000010)
#define rSPRDAT0        (*(volatile unsigned *)0x59000014)
#define rSPCON1            (*(volatile unsigned *)0x59000020)
#define rSPSTA1            (*(volatile unsigned *)0x59000024)
#define rSPPIN1            (*(volatile unsigned *)0x59000028)
#define rSPPRE1            (*(volatile unsigned *)0x5900002C)
#define rSPTDAT1        (*(volatile unsigned *)0x59000030)
#define rSPRDAT1        (*(volatile unsigned *)0x59000034)


/* SD INTERFACE */
#define rSDICON            (*(volatile unsigned *)0x5A000000)
#define rSDIPRE            (*(volatile unsigned *)0x5A000004)
#define rSDICmdArg        (*(volatile unsigned *)0x5A000008)
#define rSDICmdCon        (*(volatile unsigned *)0x5A00000C)
#define rSDICmdSta        (*(volatile unsigned *)0x5A000010)
#define rSDIRSP0        (*(volatile unsigned *)0x5A000014)
#define rSDIRSP1        (*(volatile unsigned *)0x5A000018)
#define rSDIRSP2        (*(volatile unsigned *)0x5A00001C)
#define rSDIRSP3        (*(volatile unsigned *)0x5A000020)
#define rSDIDTimer        (*(volatile unsigned *)0x5A000024)
#define rSDIBSize        (*(volatile unsigned *)0x5A000028)
#define rSDIDatCon        (*(volatile unsigned *)0x5A00002C)
#define rSDIDatCnt        (*(volatile unsigned *)0x5A000030)
#define rSDIDatSta        (*(volatile unsigned *)0x5A000034)
#define rSDIFSTA        (*(volatile unsigned *)0x5A000038)
#ifdef __BIG_ENDIAN
#define rSDIDAT            (*(volatile unsigned char *)0x5A00003F)
```

```
#else
#define rSDIDAT           (*(volatile unsigned char *)0x5A00003C)
#endif
#define rSDIIntMsk        (*(volatile unsigned *)0x5A000040)

/* ISR */
#define pISR_RESET        (*(unsigned *)(_ISR_STARTADDRESS+0x0))
#define pISR_UNDEF        (*(unsigned *)(_ISR_STARTADDRESS+0x4))
#define pISR_SWI        (*(unsigned *)(_ISR_STARTADDRESS+0x8))
#define pISR_PABORT        (*(unsigned *)(_ISR_STARTADDRESS+0xC))
#define pISR_DABORT        (*(unsigned *)(_ISR_STARTADDRESS+0x10))
#define pISR_RESERVED        (*(unsigned *)(_ISR_STARTADDRESS+0x14))
#define pISR_IRQ        (*(unsigned *)(_ISR_STARTADDRESS+0x18))
#define pISR_FIQ        (*(unsigned *)(_ISR_STARTADDRESS+0x1C))

#define pISR_EINT0        (*(unsigned *)(_ISR_STARTADDRESS+0x20))
#define pISR_EINT1        (*(unsigned *)(_ISR_STARTADDRESS+0x24))
#define pISR_EINT2        (*(unsigned *)(_ISR_STARTADDRESS+0x28))
#define pISR_EINT3        (*(unsigned *)(_ISR_STARTADDRESS+0x2C))
#define pISR_EINT4_7        (*(unsigned *)(_ISR_STARTADDRESS+0x30))
#define pISR_EINT8_23        (*(unsigned *)(_ISR_STARTADDRESS+0x34))
#define pISR_BAT_FLT        (*(unsigned *)(_ISR_STARTADDRESS+0x3C))
#define pISR_TICK        (*(unsigned *)(_ISR_STARTADDRESS+0x40))
#define pISR_WDT        (*(unsigned *)(_ISR_STARTADDRESS+0x44))
#define pISR_TIMER0        (*(unsigned *)(_ISR_STARTADDRESS+0x48))
#define pISR_TIMER1        (*(unsigned *)(_ISR_STARTADDRESS+0x4C))
#define pISR_TIMER2        (*(unsigned *)(_ISR_STARTADDRESS+0x50))
#define pISR_TIMER3        (*(unsigned *)(_ISR_STARTADDRESS+0x54))
#define pISR_TIMER4        (*(unsigned *)(_ISR_STARTADDRESS+0x58))
#define pISR_UART2        (*(unsigned *)(_ISR_STARTADDRESS+0x5C))
#define pISR_NOTUSED        (*(unsigned *)(_ISR_STARTADDRESS+0x60))
#define pISR_DMA0        (*(unsigned *)(_ISR_STARTADDRESS+0x64))
#define pISR_DMA1        (*(unsigned *)(_ISR_STARTADDRESS+0x68))
#define pISR_DMA2        (*(unsigned *)(_ISR_STARTADDRESS+0x6C))
#define pISR_DMA3        (*(unsigned *)(_ISR_STARTADDRESS+0x70))
#define pISR_SDI        (*(unsigned *)(_ISR_STARTADDRESS+0x74))
#define pISR_SPI0        (*(unsigned *)(_ISR_STARTADDRESS+0x78))
#define pISR_UART1        (*(unsigned *)(_ISR_STARTADDRESS+0x7C))
#define pISR_USBD        (*(unsigned *)(_ISR_STARTADDRESS+0x84))
#define pISR_USBH        (*(unsigned *)(_ISR_STARTADDRESS+0x88))
#define pISR_IIC        (*(unsigned *)(_ISR_STARTADDRESS+0x8C))
#define pISR_UART0        (*(unsigned *)(_ISR_STARTADDRESS+0x90))
#define pISR_SPI1        (*(unsigned *)(_ISR_STARTADDRESS+0x94))
#define pISR_RTC        (*(unsigned *)(_ISR_STARTADDRESS+0x98))
```

```
#define pISR_ADC        (*(unsigned *)(_ISR_STARTADDRESS+0xA0))


/* PENDING BIT */
#define BIT_EINT0        (0x1)
#define BIT_EINT1        (0x1<<1)
#define BIT_EINT2        (0x1<<2)
#define BIT_EINT3        (0x1<<3)
#define BIT_EINT4_7        (0x1<<4)
#define BIT_EINT8_23        (0x1<<5)
#define BIT_BAT_FLT        (0x1<<7)
#define BIT_TICK        (0x1<<8)
#define BIT_WDT           (0x1<<9)
#define BIT_TIMER0        (0x1<<10)
#define BIT_TIMER1        (0x1<<11)
#define BIT_TIMER2        (0x1<<12)
#define BIT_TIMER3        (0x1<<13)
#define BIT_TIMER4        (0x1<<14)
#define BIT_UART2        (0x1<<15)
#define BIT_LCD           (0x1<<16)
#define BIT_DMA0        (0x1<<17)
#define BIT_DMA1        (0x1<<18)
#define BIT_DMA2        (0x1<<19)
#define BIT_DMA3        (0x1<<20)
#define BIT_SDI           (0x1<<21)
#define BIT_SPI0        (0x1<<22)
#define BIT_UART1        (0x1<<23)
#define BIT_USBD        (0x1<<25)
#define BIT_USBH        (0x1<<26)
#define BIT_IIC           (0x1<<27)
#define BIT_UART0        (0x1<<28)
#define BIT_SPI1        (0x1<<29)
#define BIT_RTC           (0x1<<30)
#define BIT_ADC           (0x1<<31)
#define BIT_ALLMSK        (0xFFFFFFFF)
```

一下子搬出这个多是不是有点头晕？呵呵，其实我们不必一个一个的看，看也没用，我们需要的时候会接触到的，比如串口的配制，就那么几个寄存器，会搞熟的，要根据具体的应用来学，要不很累的，哈哈，一点愚见。。。

现在明白了没，其实就那么多东西，我们控制外设接口的时候，无非就是往寄存器里面写东西，而写东西的程序就是驱动程序，应用程序只要调用驱动程序就可以了，至于各个外设怎么和处理器联系起来的，那是硬件的东西，我们现在可以不用管（当然了，能知道原理更好，可以直接看原理图，做 PCB 板子了）。

下篇文章我们将看看各个部分（启动代码，文件系统以及内核）在 flash 中的布局，以及搬运到内核后的布局。

U-Boot 系列之二： Flash 和 SDRAM 中的布局

本篇文章主要讨论 u-boot，Linux 内核以及文件系统在 Flash 以及 SDRAM 中的布局，我用的板子是华恒的爱好者学习板，基于 S3C2410，所参考的也是华恒所给的文档。

通过 u-boot 命令 flinfo，可以看出，所用的 flash 是 intel TE28F640J3C120，flash 一共有 64 个块，每个块有 128K 大小，其中 u-boot 就放在最前面的块中，下面是其中的分配布局，第二个是对应的当把内核以及文件系统搬到内存中时内存的布局。

Flash：

```
============================<-----0x0000_0000
|       U-BOOT
============================
|
============================<-----0x0004_0000
|       Linux(zImage)
============================
|
============================<-----0x0014_0000
|  文件系统 ramdisk.image.gz
============================
|
============================<-----0x0054_0000
|  文件系统 cramfs
============================
|
============================<-----0x0074_0000
|  文件系统 JFFS2
============================<-----0x0080_0000
```

SDRAM：

```
============================<-----0x3000_0000
|
============================<-----0x3000_8000
|       Linux(zImage)
============================
|
============================<-----0x3080_0000
|   文件系统 ramdisk
============================<-----0x3200_0000
```

现在是不是对我们将要做的工作很清楚了？

其实就是先把 u-boot 移植好，然后就用供应商提供的烧写工具把 u-boot 烧写到 flash 的前面，然后就由 u-boot 来接管一切，通过网络来下载内核和文件系统（TFTP），或者把主机上的目录挂在到目标板上（NFS），这样开发和调试将非常的方便，我以前用的板子是通过 JTAG 下载下去的，有时候很不稳定，一个不是技术上的错误都可能导致出错，郁闷至极。

好了，言归正传，下一篇文章将介绍 u-boot 的整体结构，以及启动代码（start.S 中）

U-Boot 系列之三:u-boot 整体结构及启动代码分析

作者：yeshi
QQ：251059619
Blog:http://blog.chinaunix.net/u1/33990/

本篇文章首先介绍 u-boot 的整体代码结构，移植的基本步骤，然后分析启动的代码（start.S）中其中代码结构和移植步骤是参考了下面两个连接的文章，这方面资源比较多，一搜一堆，就不自己发明车轮了阿，呵呵。

http://www.mcublog.com/blog/user1/9450/archives/2006/12887.html

和老古开发网上面的一篇文章,作者是 焦玉全 黄乡生 鲍玉军,题目是第 7540 篇:U-Boot 在 S3C2410 上的移植（具体 URL 忘了，我文章是下载的，知道的朋友告诉一声）

一、整体结构

首先下载 u-boot 的源代码（www.denx.de），解压缩，你可以看到下面的目录：

- board 目标板相关文件，主要包含 SDRAM、FLASH 驱动；
- common 独立于处理器体系结构的通用代码，如内存大小探测与故障检测；
- cpu 与处理器相关的文件。如 mpc8xx 子目录下含串口、网口、LCD 驱动及中断初始化等文件；
- driver 通用设备驱动，如 CFI FLASH 驱动（目前对 INTEL FLASH 支持较好）
- doc U-Boot 的说明文档；
- examples 可在 U-Boot 下运行的示例程序；如 hello_world.c,timer.c；
- include U-Boot 头文件；尤其 configs 子目录下与目标板相关的配置头文件是移植过程中经常要修改的文件；
- lib_xxx 处理器体系相关的文件，如 lib_ppc, lib_arm 目录分别包含与 PowerPC、ARM 体系结构相关的文件；
- net 与网络功能相关的文件目录，如 bootp,nfs,tftp；
- post 上电自检文件目录。尚有待于进一步完善；
- rtc RTC 驱动程序；
- tools 用于创建 U-Boot S-RECORD 和 BIN 镜像文件的工具；

二、移植步骤

为了使 U-Boot 支持新的开发板,一种简便的做法是在 U-Boot 已经支持的开发板中选择一种和目标板接近的,并在其基础上进行修改。代码修改的步骤如下:

1)在 board 目录下创建 smdk2410 目录,添加 smdk2410.c、flash.c、memsetup.s、u-boot.lds 和 config.mk 等;

2)在 cpu 目录下创建 arm920t 目录,主要包含 start.s、interrupts.c、cpu.c、serial.c 和 speed.c 等文件;

3)在 include/configs 目录下添加 smdk2410.h,它定义了全局的宏定义等;

4)修改 u-boot 根目录下的 Makefile 文件：

```
    smdk2410_config : unconfig@./mkconfig $(@:_config=) arm arm920t smdk2410
```

5)运行 make smdk2410_config,如果没有错误,就可以开始进行与硬件相关的代码移植工作。由于这部分代码与硬件紧密相关,所以要熟悉开发板的硬件配置,可参考各芯片的用户手册。

当然，这个是一般步骤，后面我们做的可能具体文件名还和这个不一样，等到那时候在交待，这里先介绍的目的是在开始的时候给个大概的思路，要不直接分析源代码，有点在原始森林的感觉，耐心的看吧：）

三、start.S 分析

首先介绍 start.S 中的代码的具体作用，由于该代码是系统最开始执行的，这时，u-boot 对系统一无所知，必须要初始化一些东西，比如设置异常的入口地址和异常处理函数;配置 PLLCON 寄存器,确定系统的主频;屏蔽看门狗和中断;初始化 I/O 寄存器;关闭 MMU 功能;调用 /board/smdk2410 中的 memsetup.s,初始化存储器空间,设置刷新频率;将 U-Boot 的内容复制到 SDRAM 中;设置堆栈的大小, 然后 ldr pc, _start_armboot，跳到 C 函数

代码来自华恒的板子上面的资料

```
/* CPU clcok */
/* 50.00 MHz */
#define MDIV_50                 0x5c
#define PDIV_50                 0x4
#define SDIV_50                 0x2

/* 100.00 MHz */
#define MDIV_100        0xa1
#define PDIV_100        0x3
#define SDIV_100        0x1

/* 200.00 MHz */
/*CPU clock = 202.800000 Mhz, HCLK = 101.400000 Mhz, PCLK = 50.700000 Mhz*/
/*0xa1  3  1*/
/*180Mhz 90Mhz 45Mhz  0x52  1  1*/
/*152MHZ              0x44 1   1*/
#define MDIV_200                0xa1
#define PDIV_200                0x3
#define SDIV_200                0x1


#define vMPLLCON_50             ((MDIV_50 << 12) | (PDIV_50 << 4) | (SDIV_50))
```

```
#define vMPLLCON_100        ((MDIV_100 << 12) | (PDIV_100 << 4) | (SDIV_100))
#define vMPLLCON_200        ((MDIV_200 << 12) | (PDIV_200 << 4) | (SDIV_200))

/*
 *************************************************************************
 *
 * Jump vector table as in table 3.1 in [1]
 *
 *************************************************************************
 */


.globl _start
_start:    b       reset=====================================〉程序从这里开始
    ldr    pc, _undefined_instruction
    ldr    pc, _software_interrupt
    ldr    pc, _prefetch_abort
    ldr    pc, _data_abort
    ldr    pc, _not_used
    ldr    pc, _irq
    ldr    pc, _fiq

_undefined_instruction:    .word undefined_instruction
_software_interrupt:    .word software_interrupt
_prefetch_abort:    .word prefetch_abort
_data_abort:        .word data_abort
_not_used:        .word not_used
_irq:            .word irq
_fiq:            .word fiq

    .balignl 16,0xdeadbeef


/*
 *************************************************************************
 *
 * Startup Code (reset vector)
 *
 * do important init only if we don't start from memory!
 * relocate armboot to ram
 * setup stack
 * jump to second stage
 *
 *************************************************************************
 */

/*
 * CFG_MEM_END is in the board dependent config-file (configs/config_BOARD.h)
 */
_TEXT_BASE:
    .word    TEXT_BASE

.globl _armboot_start
_armboot_start:
    .word _start

/*
 * Note: _armboot_end_data and _armboot_end are defined
 * by the (board-dependent) linker script.
 * _armboot_end_data is the first usable FLASH address after armboot
 */
.globl _armboot_end_data
_armboot_end_data:
    .word armboot_end_data
.globl _armboot_end
_armboot_end:
    .word armboot_end

/*
 * _armboot_real_end is the first usable RAM address behind armboot
 * and the various stacks
 */
.globl _armboot_real_end
_armboot_real_end:
    .word 0x0badc0de

#ifdef CONFIG_USE_IRQ
/* IRQ stack memory (calculated at run-time) */
.globl IRQ_STACK_START
IRQ_STACK_START:
    .word    0x0badc0de

/* IRQ stack memory (calculated at run-time) */
.globl FIQ_STACK_START
FIQ_STACK_START:
    .word 0x0badc0de
#endif
```

```
/*
 * the actual reset code
 */

reset:=============================================〉程序开始的时候跳转到这里

    ldr     r0, =pWTCON
    mov     r1, #0x0
    str     r1, [r0]；0x0 写到 0x53000000 地址上，其实该地址是看门狗的控制寄存器，该指令就是
关闭看门狗，具体对应位请看 2410 的用户手册。

    /*
     * mask all IRQs by setting all bits in the INTMR - default
     */
    mov    r1, #0xffffffff
    ldr    r0, =INTMSK
    str    r1, [r0]；将 0xffffffff 写到 INTMSK 代表的地址上，屏蔽中断
#if defined(CONFIG_S3C2410)
    ldr    r1, =0x7ff
    ldr    r0, =INTSUBMSK
    str    r1, [r0]；对于 2410 还要设置此寄存器
#endif

        @ initialise system clocks；下面没什么要讲的，具体参看 2410 的用户手册
        mov     r1, #CLK_CTL_BASE
        mvn     r2, #0xff000000
        str     r2, [r1, #0x0]  /*oLOCKTIME*/

        @ldr    r2, mpll_50mhz
        @str    r2, [r1, #0x4]  /*oMPLLCON*/

        mov     r1, #CLK_CTL_BASE
/*        mov     r2, #0x3*/
        mov     r2, #0x3
        str     r2, [r1, #0x14]           /*oCLKDIVN*/

        mrc     p15, 0, r1, c1, c0, 0          @ read ctrl register
        orr     r1, r1, #0xc0000000            @ Asynchronous
        mcr     p15, 0, r1, c1, c0, 0          @ write ctrl register

        @ now, CPU clock is 200 Mhz
        mov     r1, #CLK_CTL_BASE
        ldr     r2, mpll_200mhz
        str     r2, [r1, #0x4]              /*oMPLLCON*/

    /*
     * we do sys-critical inits only at reboot,
     * not when booting from ram!
     */
#ifdef CONFIG_INIT_CRITICAL
    bl     cpu_init_crit ；跳到下面的子过程
#endif

relocate:；下面代码的作用是把u-boot的后续代码搬运到内存中
    /*
     * relocate armboot to RAM
     */
    adr    r0, _start        /* r0 <- current position of code */
    ldr    r2, _armboot_start
    ldr    r3, _armboot_end
    sub    r2, r3, r2        /* r2 <- size of armboot */
    ldr    r1, _TEXT_BASE
```

\board\smdk2410\config.mk中定义了_TEXT_BASE，表示把u-boot搬到内存中的相应位置，在这里是0x33F00000，相对于0x30000000来说是63MB的地方，朋友们可能要疑惑了，前面的文章不是介绍了我们用的板子是32M内存吗？那怎么是63M的地方呢，其实2410支持地址循环，这里其实就是31M的地方，哈哈，那u-boot不是把自己放在SDRAM中的最高的地方吗？的确是这样的。

```
    add    r2, r0, r2        /* r2 <- source end address */

    /*
     * r0 = source address
     * r1 = target address
     * r2 = source end address
     */
copy_loop:；开始拷贝
    ldmia    r0!, {r3-r10}
    stmia    r1!, {r3-r10}
    cmp    r0, r2
    ble    copy_loop
```

/* set up the stack */；最后我们在u-boot之后建立堆栈，为C语言的执行创建环境，否则是不允许执行C程序的，大小为128*1024

```
    ldr    r0, _armboot_end
    add    r0, r0, #c onfig_STACKSIZE 该宏在\include\configs\smdk2410.h中定义
    sub    sp, r0, #12         /* leave 3 words for abort-stack */

    ldr    pc, _start_armboot
```

_start_armboot: .word start_armboot；跳转到/lib_arm/board.c 中的 start_armboot()函数中运行了，到此我们完成了第一阶段，初始汇编代码的运行

```
/*
 *************************************************************************
 *
 * CPU_init_critical registers
 *
 * setup important registers
 * setup memory timing
 *
 *************************************************************************
 */


cpu_init_crit:
    /*
     * flush v4 I/D caches ；意思很明白不是？
     */
    mov    r0, #0
    mcr    p15, 0, r0, c7, c7, 0    // flush v3/v4 cache
    mcr    p15, 0, r0, c8, c7, 0    // flush v4 TLB

    /*
     * disable MMU stuff and caches
     */
    mrc    p15, 0, r0, c1, c0, 0
    bic    r0, r0, #0x00002300    @ clear bits 13, 9:8 (--V- --RS)
    bic    r0, r0, #0x00000087    @ clear bits 7, 2:0 (B--- -CAM)
    orr    r0, r0, #0x00000002    @ set bit 2 (A) Align
    orr    r0, r0, #0x00001000    @ set bit 12 (I) I-Cache
    mcr    p15, 0, r0, c1, c0, 0



    /*
     * before relocating, we have to setup RAM timing
     * because memory timing is board-dependend, you will
     * find a memsetup.S in your board directory.
     */
    mov    ip, lr
    bl    memsetup ==〉上面英文注释就是，在重新定位之前，要初始化RAM，以及设置刷新频率
```
我们在本文的最后插上 memsetup.S 的代码。

```
    mov    lr, ip

    mov    pc, lr 返回，并且回到上面，进行重定位


/*
 *************************************************************************
 *
 * Interrupt handling
 *
 *************************************************************************
 */

@
@ IRQ stack frame.
@
#define S_FRAME_SIZE    72

#define S_OLD_R0    68
#define S_PSR        64
#define S_PC        60
#define S_LR        56
#define S_SP        52

#define S_IP        48
#define S_FP        44
#define S_R10        40
#define S_R9        36
#define S_R8        32
#define S_R7        28
#define S_R6        24
#define S_R5        20
#define S_R4        16
#define S_R3        12
#define S_R2        8
#define S_R1        4
#define S_R0        0

#define MODE_SVC 0x13
#define I_BIT     0x80

/*
 * use bad_save_user_regs for abort/prefetch/undef/swi ...
 * use irq_save_user_regs / irq_restore_user_regs for IRQ/FIQ handling
```

```
*/

   .macro    bad_save_user_regs
   sub    sp, sp, #S_FRAME_SIZE
   stmia    sp, {r0 - r12}            @ Calling r0-r12
   add     r8, sp, #S_PC

   ldr    r2, _armboot_end
   add    r2, r2, #c onfig_STACKSIZE
   sub    r2, r2, #8
   ldmia    r2, {r2 - r4}                   @ get pc, cpsr, old_r0
   add    r0, sp, #S_FRAME_SIZE        @ restore sp_SVC

   add    r5, sp, #S_SP
   mov    r1, lr
   stmia    r5, {r0 - r4}                   @ save sp_SVC, lr_SVC, pc, cpsr, old_r
   mov    r0, sp
   .endm

   .macro    irq_save_user_regs
   sub    sp, sp, #S_FRAME_SIZE
   stmia    sp, {r0 - r12}            @ Calling r0-r12
   add     r8, sp, #S_PC
   stmdb   r8, {sp, lr}^                  @ Calling SP, LR
   str     lr, [r8, #0]                   @ Save calling PC
   mrs     r6, spsr
   str     r6, [r8, #4]                   @ Save CPSR
   str     r0, [r8, #8]                   @ Save OLD_R0
   mov    r0, sp
   .endm

   .macro    irq_restore_user_regs
   ldmia    sp, {r0 - lr}^            @ Calling r0 - lr
   mov    r0, r0
   ldr    lr, [sp, #S_PC]            @ Get PC
   add    sp, sp, #S_FRAME_SIZE
   subs     pc, lr, #4            @ return & move spsr_svc into cpsr
   .endm

   .macro get_bad_stack
   ldr    r13, _armboot_end        @ setup our mode stack
   add    r13, r13, #c onfig_STACKSIZE    @ resides at top of normal stack
   sub    r13, r13, #8
   str    lr, [r13]            @ save caller lr / spsr
```

```
    mrs    lr, spsr
    str     lr, [r13, #4]

    mov    r13, #MODE_SVC            @ prepare SVC-Mode
    @ msr    spsr_c, r13
    msr    spsr, r13
    mov    lr, pc
    movs    pc, lr
    .endm

    .macro get_irq_stack            @ setup IRQ stack
    ldr    sp, IRQ_STACK_START
    .endm

    .macro get_fiq_stack            @ setup FIQ stack
    ldr    sp, FIQ_STACK_START
    .endm

/*
 * exception handlers
 */
    .align  5
undefined_instruction:
    get_bad_stack
    bad_save_user_regs
    bl     do_undefined_instruction

    .align    5
software_interrupt:
    get_bad_stack
    bad_save_user_regs
    bl     do_software_interrupt

    .align    5
prefetch_abort:
    get_bad_stack
    bad_save_user_regs
    bl     do_prefetch_abort

    .align    5
data_abort:
    get_bad_stack
    bad_save_user_regs
    bl     do_data_abort
    .align    5
not_used:
    get_bad_stack
    bad_save_user_regs
    bl     do_not_used

#ifdef CONFIG_USE_IRQ

    .align    5
irq:
    get_irq_stack
    irq_save_user_regs
    bl     do_irq
    irq_restore_user_regs

    .align    5
fiq:
    get_fiq_stack
    /* someone ought to write a more effiction fiq_save_user_regs */
    irq_save_user_regs
    bl     do_fiq
    irq_restore_user_regs

#else

    .align    5
irq:
    get_bad_stack
    bad_save_user_regs
    bl     do_irq

    .align    5
fiq:
    get_bad_stack
    bad_save_user_regs
    bl     do_fiq

#endif
@ Processor clock values
.align 4
mpll_50mhz:
        .long   vMPLLCON_50
mpll_100mhz:
        .long   vMPLLCON_100
mpll_200mhz:
        .long   vMPLLCON_200

    .align    5
.globl reset_cpu
reset_cpu:
#ifdef CONFIG_S3C2400
    bl    disable_interrupts
    ldr    r1, _rWTCON
    ldr    r2, _rWTCNT
    /* Disable watchdog */
    mov    r3, #0x0000
    str    r3, [r1]
    /* Initialize watchdog timer count register */
    mov    r3, #0x0001
    str    r3, [r2]
    /* Enable watchdog timer; assert reset at timer timeout */
    mov    r3, #0x0021
    str    r3, [r1]
_loop_forever:
    b    _loop_forever
_rWTCON:
    .word    0x15300000
_rWTCNT:
    .word    0x15300008
#else /* ! CONFIG_S3C2400 */
    mov     ip, #0
    mcr     p15, 0, ip, c7, c7, 0          @ invalidate cache
    mcr     p15, 0, ip, c8, c7, 0          @ flush TLB (v4)
    mrc     p15, 0, ip, c1, c0, 0          @ get ctrl register
    bic     ip, ip, #0x000f                @ ............wcam
    bic     ip, ip, #0x2100                @ ..v....s........
    mcr     p15, 0, ip, c1, c0, 0          @ ctrl register
    mov     pc, r0
#endif /* CONFIG_S3C2400 */
////////////////////////////////////////////////////////////////////
```
start.S 到此结束

//下面就是初始化内存的代码

/board/smdk2410/memsetup.S

```
memsetup:
    /* memory control configuration */
    /* make r0 relative the current location so that it */
    /* reads SMRDATA out of FLASH rather than memory ! */
    ldr     r0, =SMRDATA
    ldr     r1, _TEXT_BASE
    sub     r0, r0, r1
    ldr     r1, =BWSCON    /* Bus Width Status Controller */
    add     r2, r0, #13*4
0:
    ldr     r3, [r0], #4
    str     r3, [r1], #4
    cmp     r2, r0
    bne     0b

    /* everything is fine now */
    mov     pc, lr

    .ltorg
/* the literal pools origin */

SMRDATA:
                                                                                    .word
(0+(B1_BWSCON<<4)+(B2_BWSCON<<8)+(B3_BWSCON<<12)+(B4_BWSCON<<16)+(B5_BWSCON<<20)+(B6_BWSCO
N<<24)+(B7_BWSCON<<28))
                                                                                    .word
((B0_Tacs<<13)+(B0_Tcos<<11)+(B0_Tacc<<8)+(B0_Tcoh<<6)+(B0_Tah<<4)+(B0_Tacp<<2)+(B0_PMC))
                                                                                    .word
((B1_Tacs<<13)+(B1_Tcos<<11)+(B1_Tacc<<8)+(B1_Tcoh<<6)+(B1_Tah<<4)+(B1_Tacp<<2)+(B1_PMC))
                                                                                    .word
((B2_Tacs<<13)+(B2_Tcos<<11)+(B2_Tacc<<8)+(B2_Tcoh<<6)+(B2_Tah<<4)+(B2_Tacp<<2)+(B2_PMC))
                                                                                    .word
((B3_Tacs<<13)+(B3_Tcos<<11)+(B3_Tacc<<8)+(B3_Tcoh<<6)+(B3_Tah<<4)+(B3_Tacp<<2)+(B3_PMC))
                                                                                    .word
((B4_Tacs<<13)+(B4_Tcos<<11)+(B4_Tacc<<8)+(B4_Tcoh<<6)+(B4_Tah<<4)+(B4_Tacp<<2)+(B4_PMC))
                                                                                    .word
((B5_Tacs<<13)+(B5_Tcos<<11)+(B5_Tacc<<8)+(B5_Tcoh<<6)+(B5_Tah<<4)+(B5_Tacp<<2)+(B5_PMC))
    .word ((B6_MT<<15)+(B6_Trcd<<2)+(B6_SCAN))
    .word ((B7_MT<<15)+(B7_Trcd<<2)+(B7_SCAN))
    .word ((REFEN<<23)+(TREFMD<<22)+(Trp<<20)+(Trc<<18)+(Tchr<<16)+REFCNT)
    .word 0x31
    .word 0x30
    .word 0x30
```

下篇文章，我们主要分析 start_armboot()函数的主要作用，其实该函数就是进行一系列的硬件初始化，然后进入 main_loop，等待用户的命令，或者直接默认加载 Linux 内核，启动 Linux。

U-Boot 系列之四： start_armboot()函数分析

作者：yeshi
QQ：251059619
Blog:http://blog.chinaunix.net/u1/33990/

在上一篇文章中，我们介绍了 u-boot 启动的时候汇编语言的部分，当时我们进行了一些简单的初始化，并且为 C 语言的执行建立的环境（堆栈），现在我们看看当从汇编语言转到 C 语言的时候执行的第一个函数（ start_armboot ()，在 lib_arm\board.c 中），该函数进行了一系列的外设初始化，然后调用 main_loop (),根据配置来选择是直接加载 Linux 内核还是进入等待命令模式。

在介绍该函数之前，我们需要看一看几个数据结构，这些是 u-boot 中几个重要的数据结构：

1）、gd_t 该数据结构保存了 u-boot 需要的配置信息，注释简单明了

```
typedef struct global_data {
    bd_t             *bd;            //与板子相关的结构，见下面
    unsigned long    flags;
    unsigned long    baudrate;
    unsigned long    have_console;   /* serial_init() was called */
    unsigned long    reloc_off;      /* Relocation Offset */
    unsigned long    env_addr;       /* Address  of Environment struct */
    unsigned long    env_valid;      /* Checksum of Environment valid? */
#ifdef CONFIG_VFD  //我们一般没有配置这个，这个是 frame buffer 的首地址
    unsigned long    fb_base;    /* base address of frame buffer */
#endif
} gd_t;
```

2）、bd_t 保存与板子相关的配置参数

```
typedef struct bd_info {
    int              bi_baudrate;    /* serial console baudrate */
    unsigned long    bi_ip_addr;     /* IP Address */
    unsigned char    bi_enetaddr[6]; /* Ethernet adress */
    struct environment_s           *bi_env;
    ulong            bi_arch_number;    /* unique id for this board */
    ulong            bi_boot_params;    /* where this board expects params */
    struct                /* RAM configuration */
    {
        ulong start;
        ulong size;
    }bi_dram[CONFIG_NR_DRAM_BANKS];     //在我的板子上是 1 个
} bd_t;
```

现在我贴出 start_armboot ()的源代码，然后具体的在其中解释一些代码的作用：

```
void start_armboot (void)
{
    DECLARE_GLOBAL_DATA_PTR;

    ulong size;
    gd_t gd_data;
    bd_t bd_data;
    init_fnc_t **init_fnc_ptr; //这个是函数的指针，指向一些硬件初始化的函数，见下面

    /*init_fnc_t *init_sequence[] = {
        cpu_init,           /* basic cpu dependent setup */
        board_init,         /* basic board dependent setup */
        interrupt_init,     /* set up exceptions */
        env_init,           /* initialize environment */
        init_baudrate,      /* initialze baudrate settings */
        serial_init,        /* serial communications setup */
        display_banner,
        dram_init,          /* configure available RAM banks */
        display_dram_config,
        NULL,
    };*/

    //printf("**********Start *************\n");
    /* Pointer is writable since we allocated a register for it */
    gd = &gd_data;
    memset (gd, 0, sizeof (gd_t));//初始化为 0
    gd->bd = &bd_data;
    memset(gd->bd, 0, sizeof (bd_t));//初始化为 0

//注意，下面的循环是依次调用各个硬件初始化函数，顺序见上面的的数组，我们在下一篇中依次
解释每种硬//件的初始化，第六个就是串口的初始化，这时候我们就可以通过串口输出信息了
    for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc_ptr) {
        if ((*init_fnc_ptr)() != 0) {
            hang ();      //如果不成功的话，就“输出信息，然后就进入死循环”
        }
    }

    /* configure available FLASH banks */
    size = flash_init ();  //flash 的初始化
    display_flash_config (size);//打印 flash 的配置信息

    /* initialize environment */
    env_relocate ();//环境的初始化，代码在 common\env_common.c 中

    /* IP Address */
    bd_data.bi_ip_addr = getenv_IPaddr ("ipaddr");//读取 IP 地址，保存到 bd_t 数据结构中

    /* MAC Address */ //MAC 地址的初始化
    {
        int i;
        ulong reg;
        char *s, *e;
        uchar tmp[64];

        i = getenv_r ("ethaddr", tmp, sizeof (tmp));
        s = (i > 0) ? tmp : NULL;

        for (reg = 0; reg < 6; ++reg) {
            bd_data.bi_enetaddr[reg] = s ? simple_strtoul (s, &e, 16) : 0;
            if (s)
                s = (*e) ? e + 1 : e;
        }
    }


    /* enable exceptions */  //允许中断
    enable_interrupts ();

#ifdef CONFIG_DRIVER_CS8900  //配置网卡
    if (!getenv ("ethaddr")) {
        cs8900_get_enetaddr (gd->bd->bi_enetaddr);
    }
#endif

//直接进入 main_loop 该函数在 common\main.c 中，至此，硬件初始化完成
    /* main_loop() can return to retry autoboot, if so just run it again. */
    for (;;) {
        main_loop ()
    }

    /* NOTREACHED - no way out of command loop except booting */
}
```

呵呵，好像很简单哦，其实我们并没有深入下去，只是在水面上来高屋建瓴，目的就是让大家有一个整体的映像，然后再下一篇，再根据 start_armboot()中的初始化步骤来一个一个解释每种硬件的初始化。依次是 cpu_init、board_init、interrupt_init、env_init、init_baudrate、serial_init、dram_init、flash_init 等。