

# blob

: 2001. 10. 19. ( ) 14:44:59 KST

: ([jhpark@doall.co.kr](mailto:jhpark@doall.co.kr))

: (greendrm@netsgo.com)

## 1.

### 1.1. Trap handler initialization

blob src/start.S .

\_start ,

\_start: b reset

reset ( ) .

reset:

/\* First, mask \*\*ALL\*\* interrupts \*/

ldr r0, IC\_BASE

mov r1, #0x00

str r1, [r0, #ICMR]

reset 가 , r0 IC\_BASE(IC\_BASE:  
.word 0x90050000 61 .) .(IC\_BASE  
.(SA1110 A-4( appendix)  
0x90050000 Interrupt Controller Register) .) 0x00  
r1 .

str . ( ARM7 .)

STR r0,[r1,r2] ; r1, r2

. \*(r1+r2)=r0 ..

str \*(r0+#ICMR) = r1 . , r1  
r0(=IC\_BASE) + #ICMR = 0x90050004 . r1  
(0x00) 0x90050004 .( SA1110 A-4 ICMR  
0x90050004 .)

### 1.2. CPU

```
ldr r0, PWR_BASE
ldr r1, cpuspeed
str r1, [r0, #PPCR]
```

, r1 cpuspeed .( cpuspeed 75-86  
 , (Assabet) , 가

```

str          *(r0+#PPCR) = r1          . , r1
r0(=PWR_BASE) + #PPCR = 0x90020014    . r1
          0x90020014          .(      SA1110  A-4      PPCR
0x90020014          .)

```

A-4 )

```
96     .globl memsetup
97     memsetup:
98     #if defined USE_SA1100
99     /* Setup the flash memory */
100     ldr r0, MEM_BASE
```

```
        r0    0xa0000000(44 MEM_BASE:    .long    0xa0000000 )
```

```
101
102     ldr r1, mcs0
103     str r1, [r0, #MCS0]
```

```
        r1    0xffff8fff8          (mcs0: .long 0xffff8fff8), *(r0+#MCS0=0xa0000010)
r1      .(#define MCS0    0x10 )
```

```
104
105     /* Set up the DRAM */
106
107     /* MDCAS0 */
108     ldr r1, mdcas0
109     str r1, [r0, #MDCAS0]
```

```
        r1    0xc71c703f          (      mdcas0:    .long    0xc71c703f),
*(r0+#MDCAS0=0xa0000004)  r1      .(#define MDCAS0  0x04)
```

```
110
111     /* MDCAS1 */
112     ldr r1, mdcas1
113     str r1, [r0, #MDCAS1]
```

```
        r1    0xffc71c71          (mdcas1:    .long    0xffc71c71),
*(r0+#MDCAS1=0xa0000008)  r1      .(#define MDCAS1  0x08)
```

```
114
115     /* MDCAS2 */
116     ldr r1, mdcas2
117     str r1, [r0, #MDCAS2]
```

```
        r1    0xffffffff          (mdcas2: .long 0xffffffff), *(r0+#MDCAS2=0xa000000c)
        r1      .(#define MDCAS2  0x0c)
```

```

118
119     /* MDCNFG */
120     ldr r1, mdcnfg
121     str r1, [r0, #MDCNFG]

```

```

r1      0x0334b22f      (mdcnfg:      .long      0x0334b22f),
*(r0+#MDCNFG=0xa0000000) r1      .(define MDCNFG 0x0)

, mdcas0, mdcas1, mdcas2, mdcnfg, msn0

```

```

SA1100 A-4      . MDCNFG
32bit read/write      DRAM      control bit 가      DRAM
bank      DRAM device      . MDCASX      가
      32bit read/write가 가 , CAS waveform
DRAM      CPU cycle      . MSCX      가
      32bit read/write가 가 , ROM flash Static
      . MECR      expansion      , PCMCIA
      BLOB
C      reset

```

```

138     ldr r1, MEM_START

MEM_START:      .long  0xc0000000 ) r1  0xc0000000

139
140 .rept  8
141     ldr r0, [r1]
142 .endr

145 #elif defined USE_SA1110
      cpu가 SA1110 ,

```

```

146     /* This part is actually for the Assabet only. If your board
147     * uses other settings, you'll have to ifdef them here.
148     */
149     /* Set up the SDRAM */
150     mov r1, #0xA0000000 /* MDCNFG base address */

r1      0xA0000000      . (      0xA0000000      DRAM configuration
register      . SA-1100      A-4      .)

```

```

151
152     ldr r2, =0xAAAAAA7F
153     str r2, [r1, #0x04]    /* MDCAS00 */
154     str r2, [r1, #0x20]    /* MDCAS20 */

```

```

r2  0xAAAAAA7F          , (r1(=0xA0000000)+0x04)=0xA0000004  r2
.   (r1(=0xA0000000)+0x20)=0xA0000020  r2          .

```

```

155
156     ldr r2, =0xAAAAAAAA
157     str r2, [r1, #0x08]    /* MDCAS01 */
158     str r2, [r1, #0x24]    /* MDCAS21 */

```

```

r2  0xAAAAAAAA          , (r1(=0xA0000000)+0x08)=0xA0000008  r2
.   r2  (r1(=0xA0000000)+0x24)=0xA0000024  r2          .

```

```

159
160     ldr r2, =0xAAAAAAAA
161     str r2, [r1, #0x0C]    /* MDCAS02 */
162     str r2, [r1, #0x28]    /* MDCAS22 */

```

```

r2  0xAAAAAAAA          , (r1(=0xA0000000)+0x0C)=0xA000000C  r2
.   .   (r1(=0xA0000000)+0x28)=0xA0000028  r2          .

```

```

163
164     ldr r2, =0x4dbc0327    /* MDREFR */
165     str r2, [r1, #0x1C]

```

```

r2  0x4dbc0327          , (r1(=0xA0000000)+0x1C)=0xA000001C  r2
.

```

```

166
167     ldr r2, =0x72547254    /* MDCNFG */
168     str r2, [r1, #0x00]

```

```

r2  0x72547254          , (r1(=0xA0000000)+0x00)=0xA0000000  r2
.

```

```

169
170     /* Issue read requests to disabled bank to start refresh */
171
172     ldr r1, =0xC0000000

```

```

r1  0xC0000000          .

```

```

173
174 .rept 8
175     ldr r0, [r1]
176 .endr

```

```

        register                . r1  MEM_START(=0cC0000000)
        , r0  r1                8          .          disable
bank7 refresh2)

```

```

177
178     mov r1, #0xA0000000      /* MDCNFG base address */
179
180     ldr r2, =0x72547255      /* Enable the banks */
181     str r2, [r1, #0x00]      /* MDCNFG */
182
183 /* Static memory chip selects on Assabet: */
184
185     ldr r2, =0x4b90          /* MCS0 */
186     orr r2,r2,r2,lsr #16
187     str r2, [r1, #0x10]
188
189     ldr r2, =0x22212419      /* MCS1 */
190     str r2, [r1, #0x14]
191
192     ldr r2, =0x42196669      /* MCS2 */
193     str r2, [r1, #0x2C]
194
195     ldr r2, =0xafccafcc      /* SMCNFG */
196     str r2, [r1, #0x30]
197

```

```

Assabet      SA1110          .          PCMCIA

```

```

198 /* Set up PCMCIA space */
199

```

## 2) DRAM

```
refresh
```

```

200    ldr r2, =0x994a994a
201    str r2, [r1, #0x18]
r2    0xx994a994a          , (r1(=0xA0000000)+0x18)=0xA00000018    r2
      . 0xA00000018      Expansion bus configuration register(
      ) .

```

```

202
203 /* All SDRAM memory settings should be ready to go... */
204 /* For best performance, should fill out remaining memory config regs: */
205
206
207 /* Testing ,Chester */
208 mov r3,#0x12000000
209 mov r2,#0x5000          /* D9_LED on and D8_LED off */
210 str r2,[r3]
211 mov r4, #0x20000
r3    0x12000000          , r2    0x5000          , *(r3) = r2          , r4
0x20000          .

```

```

212 gogogo2:
213     subs r4, r4, #1
214     bne gogogo2
215 #else
216 #error "Configuration error: CPU not defined!"
217 #endif
218
219     mov pc, lr

```

25 memsetup.s

pc( )  
 lr((link register)<sup>3</sup>) 가 (mov pc, lr).

#### 1.4. LED

start.S . (bl memsetup)

```

108 /* init LED */
109 bl ledinit

```

26 setup.S

3) r14가 link register

```

LED                                     . LED    GPIO
LED

```

```

51 .globl ledinit
52  /* initialise LED GPIO and turn LED on.
53     * clobbers r0 and r1
54     */
55 ledinit:
56     ldr r0, GPIO_BASE
57     ldr r1, LED
58     str r1, [r0, #GPDR]    /* LED GPIO is output */
59     str r1, [r0, #GPSR]    /* turn LED on */
60     mov pc, lr

```

27 ledasm.S

```

r0  GPIO_BASE(43 GPIO_BASE: .long 0x90040000)
r1  LED(LED: .long LED_GPIO(=0x00020000(Assabet )))
*(r0+#GPDR=0x90040004)  r1      .(define GPDR 0x00000004)
*(r0+#GPSR=0x90040008)  r1      .(define GPSR 0x00000008)

```

```

/* define the GPIO pin for the LED */
// LED      GPIO
#if defined ASSABET
# define LED_GPIO 0x00020000 /* GPIO 17 */
#elif (defined CLART) || (defined LART) || (defined NESA)
# define LED_GPIO 0x00800000 /* GPIO 23 */
#elif defined PLEB
# define LED_GPIO 0x00010000 /* GPIO 16 */
#else
#warning "FIXME: Include code to turn on one of the LEDs on your board"
# define LED_GPIO 0x00000000 /* safe mode: no GPIO, so no LED */
#endif

```

28 ../include/led.h



```

65 .globl led_on
66 /* turn LED on. clobbers r0 and r1 */
67 led_on:
68     ldr r0, GPIO_BASE
69     ldr r1, LED
70     str r1, [r0, #GPSR]
71     mov pc, lr

```

```

r0 GPIO_BASE(43 GPIO_BASE: .long 0x90040000)
r1 LED(LED: .long LED_GPIO(=0x00020000(Assabet )))
*(r0+#GPSR=0x90040008) r1 (#define GPSR 0x00000008).

```

```

76 .globl led_off
77 /* turn LED off. clobbers r0 and r1 */
78 led_off:
79     ldr r0, GPIO_BASE
80     ldr r1, LED
81     str r1, [r0, #GPCR]
82     mov pc, lr

```

```

r0 GPIO_BASE(43 GPIO_BASE: .long 0x90040000)
r1 LED(LED: .long LED_GPIO(=0x00020000(Assabet )))
*(r0+#GPCR=0x9004000c) r1 (#define GPCR 0x0000000c).

```

ASSABET LED\_GPIO가 0x00020000 , r1  
, GPDR output GPIO pin direction , GPSR GPCR  
bit ON/OFF . LED  
, start.S 가 .

## 1.5. wakeup

```

113     ldr r0, RST_BASE
114     ldr r1, [r0, #RCSR]
115     and r1, r1, #0x0f
116     teq r1, #0x08
117     bne normal_boot /* no, continue booting */

```

```

31 start.S
r0 RST_BASE(RST_BASE: .word 0x90030000 68 .)
. (RST_BASE

```

.(SA1110 A-3( appendix) 0x90030000 Reset controller software reset register) .)

```

    str                r1 = *(r0+#RCSR(#define RCSR 0x04))
.    , r1    r0(=RST_BASE) + #RCSR = 0x90030004
.    r1      0x90030004 .(    SA1110 A-4    RCSR
    0x90030004 .) r1    0x0f and
4    r1      .    r1    0x08    normal_boot
.

```

```

118
119    /* yes, a wake-up. clear RCSR by writing a 1 (see 9.6.2.1 from [1])
    */
120    mov r1, #0x08
121    str r1, [r0, #RCSR] ;

```

0x08 r1 . r1 r0(=RST\_BASE) + #RCSR = 0x90030004

```

122
123    /* get the value from the PSPR and jump to it */
    /* PSPR 가 . */
124    ldr r0, PWR_BASE
125    ldr r1, [r0, #PSPR]
126    mov pc, r1

```

r0 PWR\_BASE(PWR\_BASE: .word 0x90020000 64 .)  
. (PWR\_BASE

.(SA1110 A-3( appendix) 0x90020000 Power Manager control register) .)

```

    str                r1 = *(r0+#PSPR( #define PSPR 0x08))
.    , r1    r0(=PWR_BASE) + #PSPR = 0x90020008
.    (    SA1110 A-4    PSPR    0x90020008
.)

```

SA1110 Reset Controller Reset ,  
, register . software reset  
reset 가 가  
. RSRR RCSR .  
, RCSR r1 . 0x0f and 4

```

0x08 , , , (watchdog)
, RCSR RCSR 0x08
, PSPR( )
r1 , .
가 ,
normal_boot 가 . 가
normal_boot .

```

## 1.6. i-cache enable

```

129 normal_boot:
130     /* enable I-cache */
131     mrc p15, 0, r1, c1, c0, 0 @ read control reg
132     orr r1, r1, #0x1000 @ set Icache
133     mcr p15, 0, r1, c1, c0, 0 @ write it back

```

34 start.S

## 1.7.

```

136     /* check the first 1MB in increments of 4k */
137     mov r7, #0x1000
138     mov r6, r7, lsl #8 /* 4k << 2^8 = 1MB */
139     ldr r5, MEM_START

```

$r7 = 0x1000 (=4K)$  .  
 $r6 := r7 \ll 8$      $r7 = 8$      $r6 = 2^{20} = 1MB$  가    4).  
 $r5 = MEM\_START(72)$      $MEM\_START: .long 0xc0000000)$  .

```

141 mem_test_loop:
142     mov r0, r5
143     bl testram
144     teq r0, #1
145     beq badram

```

$r5 = r0$  .     $r0 = 0xc0000000$  .     $testram$  .     $r0$ 가 1  
 $badram$  .

---

4) 2    1    2    8  
      2 8     $2^8$     4K  $2^{12}$     8  
       $2^{12} * 2^8$      $2^{20}$  .

```

146
147     add r5, r5, r7
148     subs    r6, r6, r7
149     bne mem_test_loop

```

```

r5    r7          r5          . r6    r7          r6          . r6    0          ,
mem_test_loop    .(      )

.
, lcache
.
1Mbyte
.
r7    4K(=0x1000)    ,
8bit
r6          . r6    1M    가          . r5
MEM_START(=0xC0000000)    가
가          mem_test_loop    .    loop
4Kbytes    1Mbytes    . , r0
가    .(r5).    testram    (bl),
(r0)    1    , badram    , r5    r7(=4K)
, r6    r7    loop

```

```

34 .globl testram
35     @ r0 = address to test // r0    test
36     @ returns r0 = 0 - ram present, r0 = 1 - no ram // r0가 0    RAM
    , r0가 1
37     @ clobbers r1 - r4    // r1 ~ r4

```

38 testme.S

: ARM7 .

\* Block Data Transfer (LDM,STM)

```

LDR, STR    가
. ARM7    ...

LDR    가    LDM    가
,    가    . 가
, ARM7    Push, Pop
.    LDR    STR    ...    LDM    STM
...

```

(1) <LDM|STM>{cond}mode Rn{!},{reg\_list}{ ^ }

{cond} . Rn  
 ! WriteBack ... LDR STR  
 {reg\_list}  
 1000 r1,r2,r3 , 1000  
 ... r13 , Rn r13  
 , {reg\_list} {r1,r2,r3} . mode  
 가  
 가 8가 가

가 .

{ ^ } ... , 가 ... 가  
 가

{ ^ } if present set S bit to load the CPSR along with the PC, or  
 force transfer of user bank when in privileged mode...(???)

... 가 .

1 , LDM STM .

LDM : (Rn)

STM : LDM .

가 . 8086  
 , 8086

...

ARM7 LDM STM

Multiple Register

push ax  
 push bx  
 push cx  
 push dx

가 8086 , ARM7 Single

```
STR    r0,[sp],#4
STR    r1,[sp],#4
STR    r3,[sp],#4
STR    r4,[sp],#4
```

, Multiple ,

```
STMEA  sp!,{r0,r1,r2,r4}
```

가 ...

.

2 , {Reg\_List} ...

. 가

? ARM7

r0 r15

16 ..

LDM STM 16 . ,

, 가 .

, LDM 16 가

r0-r15 1:1 {reg\_list}

. !

, {r0,r2} , {r0-r5} ,

{r0-r3,r6-r7} .

, {r3,r2,r1} {r1,r2,r3} ,

가 .. .... 16 가

1:1 ... ..

가 .. .

... 3 . !!!

STM 가 STMEA . EA가

가 8가 가 , 4가 가 .

4가 .

### A) Post-Increment Addressing

가 ( ) , , / , / , / .

가 .

{r1,r2} R10 0x1000 가 R10 , Post-Increment ,

1. 0x1000 r1 .
2. Base 0x1004 가 .
3. 0x1004 r2가 .
4. Base 0x1008 가 .

! r10 0x1008 .

### B) Pre-Increment Addressing

가 . 가 .

1. Base 0x1004 가 .
2. 0x1004 r1 .
3. Base 0x1008 가 .
4. 0x1008 r2가 .

! r10 0x1008 .

### C) Post-Decrement Addressing

가 . , ( )

가 ,

... 가 .

1. 0x1000 r2가 .(!!! r1 r2)
2. Base 0x0FFC .
3. 0x0FFC r1 .

4. Base                    0x0FF8                    .

!                    r10                    0xFF8                    .

#### D) Pre-Decrement Addressing

1. Base                    0x0FFC                    .

2. 0x0FFC                    r2가                    .

3. Base                    0x0FF8                    .

4. 0x0FF8                    r1                    .

!                    r10                    0xFF8                    .

4                    .

8                    ...

,                    2가                    .

	Stack	Other
pre increment load	LDMED	LDMIB
post increment load	LDMFD	LDMIA
pre decrement load	LDMEA	LDMDB
post decrement load	LDMFA	LDMDA
pre increment store	STMFA	STMIB
post increment store	STMEA	STMIA
pre decrement store	STMFD	STMDB
post decrement store	STMED	STMDA

Stack                    가                    ,  
 . ,                    LDMED                    LDMIB                    ,  
 .

increment                    .. D                    decrement                    ..                    Other                    , I  
 B                    Before , A                    After

LDMDA                    post decrement                    ....

Stack                    , E                    Empty                    F                    Full                    .  
 sp가 가                    ,                    .



... Post , ,  
 / 가 가  
 . Empty가 ...

Load Empty sp가 가  
 sp 가 .. Load Pre가 Empty  
 . Store Empty  
 sp . post 가 Empty ...!!

Full .

D Decending, A Ascending .  
 . 8086 Push sp ?  
 decending Stack . push STM STM  
 Decrement D ... Pop LDM  
 decending stack Pop sp 가 ... LDM  
 increment 가 D .

A ... LDM STM  
 . LDM STM ...

---

```
38 testram:
39     ldmia    r0, {r1, r2}    @ store current value in r1 and r2 //
```

```
40 testmem.S
r0(      가      .)
r1      , r0      가      가(r0+0x4)      ,      가      r2
.
```

```
139     ldr r5, MEM_START
140
141     mem_test_loop:
142     mov r0, r5
143     bl testram
```

```
40     mov r3, #0x55    @ write 0x55 to first word // r3    0x55    .
41     mov     r4, #0xaa    @ 0xaa to second // r4    0xaa    .
42     stmia   r0, {r3, r4}
```

```
stmia   r0(      가      .)
r3      , r0      (r0+0x4)가      가      ,      가      r4      .
```

```

43      ldmia    r0, {r3, r4}    @ read it back      //
44      teq      r3, #0x55    @ do the values match // r3가 0x55
45      teqeq    r4, #0xaa    // r4가 0xaa
46      bne      bad      @ oops, no
47      mov      r3, #0xaa    @ write 0xaa to first word // r3 0xaa
48      mov      r4, #0x55    @ 0x55 to second      // r4 0x55
49      stmia    r0, {r3, r4}
50      ldmia    r0, {r3, r4}    @ read it back      //
51      teq      r3, #0xaa    @ do the values match //
52      teqeq    r4, #0x55

53 bad:  stmia    r0, {r1, r2}    @ in any case, restore old data //

54      moveq    r0, #0      @ ok - all values matched //
r0 0
55      movne    r0, #1      @ no ram at this location //
r0 1
56      mov      pc, lr      //

```

```

r0      test    가,      test    r0      가
0      , RAM
r1      r4      subroutine    register
r0가 가      32bit    r1    r2      (ldmia). r3    0x55    r4
0xaa      bad      r1, r2      r0가 가
(stmia).    r0가 가      r3    r4
.(teq,teqeq)
RAM      , bad
r3      0xaa      , r4    0x55      , r0      가      r3    r4
.(stmia).    r3    r4      가      (ldmia)
. r0      r1    r2      , r0      RAM
, 0    1      .(moveq, movne)
(start.S)      r0      가

```

```

208 badram:
209      b      blinky
210

```

```

44 start.S
blinky

```

```

214 blinky:
215     /* This is test code to blink the LED
216         very useful if nothing else works */
217         // LED가 test
218     bl led_on
219     bl wait_loop
220     bl led_off
221     bl wait_loop
222     b blinky

```

led\_on  
wait\_loop  
led\_off  
wait\_loop

```

223 wait_loop:
224     /* busy wait loop*/
225     mov r2, #0x1000000

```

r2 0x1000000

	RAM		, badram		, badram
blinky	. blinky	LED		. led_on	
led_off	ledasm.S	가	,	led	,
wait_loop	LED가				

```

65 .globl led_on
66     /* turn LED on. clobbers r0 and r1 */
67         // LED
68 led_on:
69     ldr r0, GPIO_BASE
70     ldr r1, LED
71     str r1, [r0, #GPSR]
72     mov pc, lr

```

47	ledasm.S				
GPIO_BASE(=0x90040000)	ledasm.S	43	GPIO_BASE:	.long	
0x90040000					
LED	LED_GPIO	ledasm.S	42	LED:	.long LED_GPIO
	LED_GPIO	led.h			

```

30 /* define the GPIO pin for the LED */
31 #if defined ASSABET
32 # define LED_GPIO 0x00020000 /* GPIO 17 */
33 #elif (defined CLART) || (defined LART) || (defined NESA)
34 # define LED_GPIO 0x00800000 /* GPIO 23 */
35 #elif defined PLEB
36 # define LED_GPIO 0x00010000 /* GPIO 16 */
37 #else
38 #warning "FIXME: Include code to turn on one of the LEDs on your board"
39 # define LED_GPIO 0x00000000 /* safe mode: no GPIO, so no LED */
40 #endif

```

```

48 include/led.h

```

```

GPSR    ledasm.S    45    #define GPSR    0x00000008
.
*(r0+#GPSR=0x90040008)    r1    .(#define GPSR 0x00000008) [    .
SA1110    A-4    GPSR    .]

```

```

76 .globl led_off
77 /* turn LED off. clobbers r0 and r1 */
    // LED    .
78 led_off:
79    ldr r0, GPIO_BASE
80    ldr r1, LED
81    str r1, [r0, #GPCR]
82    mov pc, lr

```

```

49 ledasm.S
    , GPCR    ledasm.S    46    #define GPCR    0x00000008
.
*(r0+#GPCR=0x9004000c)    r1    .(#define GPCR 0x0000000c) [    .
SA1110    A-4    GPCR    .]

```

```

152  /* the first megabyte is OK, so let's clear it */
      // 1MBytes가
153  mov r0, #((1024 * 1024) / (8 * 4)) /* 1MB in steps of 32 bytes */
154  ldr r1, MEM_START
155  mov r2, #0
156  mov r3, #0
157  mov r4, #0
158  mov r5, #0
159  mov r6, #0
160  mov r7, #0
161  mov r8, #0
162  mov r9, #0

```

50 start.S

r0 1Mbyte 32  
r1 MEM\_STAR(=0xc0000000) 가  
r2-r9 0  
1Mbyte , 1M 0

```

164 clear_loop:
165  stmia r1!, {r2 - r9}
166  subs r0, r0, #(8 * 4) // r0 32 r0
167  bne clear_loop // (r0 != 0) clear_loop

```

51 start.S

stmia(Store and Increment after) r1 가 r2 r9  
. r0 32 가 . r0가 0 loop

```

170  /* get a clue where we are running, so we know what to copy */
      //
171  and r0, pc, #0xff000000 /* we don't care about the low bits */
      //pc( ) 0xff000000 and r0
174  /* relocate the second stage loader */
      //
175  add r2, r0, #(128 * 1024) /* blob is 128kb */ // blob
128Kbytes.

```

5) r1! ! auto-index r1 . stmia가  
r1 r1 32(4\*8) . , stmia r1, {r2, r8}  
r1 r1 .

```

176    add r0, r0, #0x400    /* skip first 1024 bytes */    //    1024
byte
177    ldr r1, MEM_START    // r1    MEM_START    .
178    add r1, r1, #0x400    /* skip over here as well */
    // r1    0x400    r1    .
r0    (128*1024)    r2    .
r0    0x400(100000000000(    )=2 ^ 10=1024)    r0    .
1024    .
r1    MEM_START(=0xc0000000)    .
r1    0x400(100000000000(    )=2 ^ 10=1024)    r1    .
C0000000    1024    .(0xC0000400)

```

```

179
180    /* r0 = source address    // r0    .
181    * r1 = target address    // r1    .
182    * r2 = source end address    // r2    .
183    */

```

```

r0    0d    r0    .    128K    r2    ,
0x400    1024    .
reset-ld-script    0xC0000400
blob가    1024Bytes    가    .

```

```

184 copy_loop:
185    ldmia    r0!, {r3 - r10}
186    stmia    r1!, {r3 - r10}
187    cmp r0, r2
188    ble copy_loop
189
190
191    /* turn off the LED. if it stays off it is an indication that
192    * we didn't make it into the C code
193    */
    // LED    .    OFF    , C
.
194    bl led_off
195
196
197    /* set up the stack pointer */

```

```

//
198  ldr r0, MEM_START // r0 MEM_START
199  add r1, r0, #(1024 * 1024) // r1 = r0 + (1024*1024 = 1M)
200  sub sp, r1, #0x04 // sp = r1 - 0x04
201
202
203  /* blob is copied to ram, so jump to it */
        // blob 가 , .
204  add r0, r0, #0x400 // r0 = r0 + 0x400
205  mov pc, r0 // sp = r0

```

54 start.S

```

184 copy_loop:
185  ldmbia r0!, {r3 - r10}
186  stmbia r1!, {r3 - r10}
187  cmp r0, r2
188  ble copy_loop
189
190
191  /* turn off the LED. if it stays off it is an indication that
192   * we didn't make it into the C code
193   */
        // LED . OFF , C
        .
194  bl led_off
195
196
197  /* set up the stack pointer */
        //
198  ldr r0, MEM_START // r0 MEM_START
199  add r1, r0, #(1024 * 1024) // r1 = r0 + (1024*1024 = 1M)
200  sub sp, r1, #0x04 // sp = r1 - 0x04
201
202
203  /* blob is copied to ram, so jump to it */
        // blob 가 , .
204  add r0, r0, #0x400 // r0 = r0 + 0x400
205  mov pc, r0 // sp = r0

```

r0 copy source 가 가 , r1 target  
, r2 copy (128Kbyte)가 가

```

        .      r0가 가      r1 가      copy      .      ldmia
stmia      가      , r0가 r2      , copy가      .
        , copy_loop      copy      .
copy가      , LED off      (led_off),r0
        (MEM_START), r1      1MBytes      , sp
r1      4      가      .      stack pointer      C
routine      가      .      1024Bytes      MEM_START
cpy      , r0      0x400      pc
        .(mov pc, r0).      trampoline.S
second stage boot loader      .

```

## 1.8. trampoline.S

```

30 .globl _trampoline
31 _trampoline:
32     bl  main // main      . ( C code )
33     /* if main ever returns we just call it again */
        //      .(      )
34     b   _trampoline

```

```

56 trampoline.S
        main.c      main()      .
main      trampoline
.

```

## 2. C

```

main.c      .
main.c      serial      , timer      ,
        , blob      .      RAMDISK      Flash
RAM      ,      .

```

```

----- main.c -----
76 int main(void)
77 {
78     u32 blockSize = 0x00800000; //
79     int numRead = 0; //
80     char cmdline[128]; //
.
81     int i;
82     int retval = 0;
83
84     /* Turn the LED on again, so we can see that we safely made it

```



```

85         * into C code.
86     */
           // LED가 , C
.
87     led_on();
88
89     /* We really want to be able to communicate, so initialise the
90     * serial port at 9k6 (which works good for terminals)
91     */
           //
9600         . ( .)
           // baud9k6 serial.h
           // BaudRate
           BRD
BaudRate = (3.6864*10 ^ 6/16*(BRD+1))
           BRD serial port UART control register
가 , serial BaudRate
----- main.c -----
----- ./include/serial.h -----
41 typedef enum { /* Some useful SA-1100 baud rates */
42     baud1k2 = 191,
43     baud9k6 = 23,
44     baud19k2 = 11,
45     baud38k4 = 5,
46     baud57k6 = 3,
47     baud115k2 = 1
48 } eBauds;
----- ./include/serial.h -----
           BRD 가
           115200 baud115k2
SerialInit
SerialInit(baud115k2); //
----- ./serial.c -----
48 void SerialInit(eBauds baudrate)
49 {
50     /* Theory of operations:
51     * - Flush the output buffer
52     * - switch receiver and transmitter off

```

```

53      * - clear all sticky bits in control register 3
54      * - set the port to sensible defaults (no break, no interrupts,
55      *   no parity, 8 databits, 1 stopbit, transmitter and receiver
56      *   enabled
57      * - set the baudrate to the requested value
58      * - turn the receiver and transmitter back on
59      */
        //
        // -          가
        // -          (          )
        // -          3
        // -          .(no break, no interrupts, no
parity, 8 databits, 1 stopbit, transmitter and receiver enabled.)
        // -
        // -          (          )

60
61 #if defined USE_SERIAL1          // 1
62     while(Ser1UTSR1 & UTSR1_TBY) {
63     }
64
65     Ser1UTCR3 = 0x00;
66     Ser1UTSR0 = 0xff;
67     Ser1UTCR0 = ( UTCR0_1StpBit | UTCR0_8BitData );
68     Ser1UTCR1 = 0;
69     Ser1UTCR2 = (u32)baudrate;
70     Ser1UTCR3 = ( UTCR3_RXE | UTCR3_TXE );
71 #elif defined USE_SERIAL3 // 3
72     while(Ser3UTSR1 & UTSR1_TBY) {
73     }
74
75     Ser3UTCR3 = 0x00;
76     Ser3UTSR0 = 0xff;
77     Ser3UTCR0 = ( UTCR0_1StpBit | UTCR0_8BitData );
78     Ser3UTCR1 = 0;
79     Ser3UTCR2 = (u32)baudrate;
80     Ser3UTCR3 = ( UTCR3_RXE | UTCR3_TXE );
81 #else
82 #error "Configuration error: No serial port used at all!" //
        .
83 #endif
84 }

```

```

----- ./serial.c -----
1, 3, 1 3
가, 가, 가
( ), 가 UTCR3(UART Control Register 3)
0x00 Rx Tx OFF, UTCR0(UART Control Register 0) 0xFF
( ). UTCR0 1 8
, UTCR1 0 UTCR2
, UTCR3 Rx
UART3 Tx Rx Tx가
UTCR1 BRD 4bit, UTCR2 8bit 가
.
=( ,
#include/asm-arm/arch-sa1100/SA-1100.h 가
가 .)

----- main.c -----
92 SerialInit(baud9k6); //
93 TimerInit(); //
----- main.c -----
TimerInit() time.c
timer interrupt
----- time.c -----
50 void TimerInit(void)
51 {
52 /* clear counter */
// OSCR(OS timer Counter Register(
))
// OSCR 0
53 OSCR = 0;
54
55 /* we don't want to be interrupted */
// 가
// OSER(OS timer Interrupt Enable Register(OS
))
// OIER 0
56 OIER = 0;
57
58 /* wait until OSCR > 0 */
// OSCR 0
59 while(OSCR == 0) // OSCR 0

```

```

60      ;
61
62      /* clear match register 0 */
        // 0
        // OSMR0 (OS timer Match Register 0(0
    ))
        // OSMR0      0
63      OSMR0 = 0;
64
65      /* clear match bit for OSMR0 */
        // OSMR0

        // OSSR (OS timer State Register(
    ))
        // OSSR      OSSR_M0
-----SA-1100.h-----
#define OSSR_M(Nb)      /* Match detected [0..3]      */ \
    (0x00000001 << (Nb))
#define OSSR_M0      OSSR_M (0) /* Match detected 0 */
OSSR_M0 = 0x00000001
-----SA-1100.h-----
66      OSSR = OSSR_M0;
67
68      numOverflows = 0;
69 }

----- time.c -----
SA1100      OSCR(Operating System Counter Register)      up-counter
register 4      OSMR(Operation System Match Register)가      . OSCR
reset      , OSMR      가 read write
      OSCR      OSMR      , interrupt enable bit
      OSSR(Operating System Status Register)      bit
bit interrupt controller bit route
      OSMR3      watchdog match register
      OSCR      OSMR3      , SA1100 reset
Reset      known state
WMER(Watchdog Match Enable Register)가      ,      FIQ(Fast Interrupt)
IRQ(Interrupt) CPU
      (status register)
      OSCR      0      OIER      0
counter interrupt      .      OSCR      countering
      , OSMR0      0      , OSSR      OSMR0      bit

```

```

clear          , OSSR      bit  1  write      clear
numOverflows  Overflow가   가          0

```

```

----- main.c -----
94
95  /* Print the required GPL string */
    // GPL
96  SerialOutputString("\nConsider yourself LARTed!\n\n");
97  SerialOutputString(PACKAGE " version " VERSION "\n"
98                      "Copyright (C) 1999 2000 2001 "
99                      "Jan-Derk Bakker and Erik Mouw\n"
100                     "Copyright (C) 2000 "
101                     "Johan Pouwelse\n");
102  SerialOutputString(PACKAGE " comes with ABSOLUTELY NO
WARRANTY; "
103                     "read the GNU GPL for details.\n");
104  SerialOutputString("This is free software, and you are welcome "
105                     "to redistribute it\n");
106  SerialOutputString("under certain conditions; "
107                     "read the GNU GPL for details.\n");

```

```

----- main.c -----
serial
SerialOutputString()      string.c

```

```

----- serial.c -----

116 /*
117  * Write a null terminated string to the serial port.
118  */
    //
119 void SerialOutputString(const char *s) {
120
121     while(*s != 0)
122         SerialOutputByte(*s++);
123
124 } /* SerialOutputString */

197 /*
198  * Read a single byte from the serial port. Returns 1 on success, 0
199  * otherwise. When the function is succesfull, the character read is

```

```

200  * written into its argument c.
201  */
    //          .          1          0
    .          ,          c          .

202 int SerialInputByte(char *c)
203 {
204 #if defined USE_SERIAL1          //          1          .
205     if(Ser1UTSR1 & UTSR1_RNE) {
206         int err = Ser1UTSR1 & (UTSR1_PRE | UTSR1_FRE |
UTSR1_ROR);
207         *c = (char)Ser1UTDR;
208 #elif defined USE_SERIAL3 //          3          .
209     if(Ser3UTSR1 & UTSR1_RNE) {
210         int err = Ser3UTSR1 & (UTSR1_PRE | UTSR1_FRE |
UTSR1_ROR);
211         *c = (char)Ser3UTDR;
212 #else
213 #error "Configuration error: No serial port at all"
214 #endif
215
216     /* If you're lucky, you should be able to use this as
217        * debug information ;- ) -- Erik
218        */
        //

219     if(err & UTSR1_PRE)          // err & UTSR1_PRE
220         SerialOutputByte('@'); // '@'
221     else if(err & UTSR1_FRE) // err & UTSR1_FRE
222         SerialOutputByte('#'); // '#'
223     else if(err & UTSR1_ROR) // err & UTSR1_ROR
224         SerialOutputByte('$'); // '$'
225
226     /* We currently only care about framing and parity errors */
        //

227     if((err & (UTSR1_PRE | UTSR1_FRE)) != 0) {
228         return SerialInputByte(c);
229     } else {
230         led_toggle();
231         return(1);
232     }

```

```

233     } else {
234         /* no bit ready */
235         return(0);
236     }
237 } /* SerialInputByte */
----- serial.c -----
SerialOutputByte() 1Byte
SerialOutputByte()
SerialOutputByte() Ser1UTDR Ser3UTDR
(character)
UTSR , Tx FIFO가 가
Tx FIFO가 half-full , UTDR
가 " \ n" " \ r" 가 가
----- main.c -----
108
109
110 /* get the amount of memory */
111 // 가
112 get_memory_map();
----- main.c -----
get_memory_map
— , 가
blob_status
main.h blob_status_t ,
----- memory.c -----
49 void get_memory_map(void)
50 {
51     u32 addr;
52     int i;
53
54     /* init */ //
55     for(i = 0; i < NUM_MEM_AREAS; i++)
56         memory_map[i].used = 0;
57
58     /* first write a 0 to all memory locations */
59     // 0
60     for(addr = MEMORY_START; addr < MEMORY_END; addr +=
TEST_BLOCK_SIZE)

```

```

60         * (u32 *)addr = 0;
61
62         /* scan memory in blocks */    //
63         i = 0;
64         for(addr = MEMORY_START; addr < MEMORY_END; addr +=
TEST_BLOCK_SIZE)    {
65             if(testram(addr) == 0) {
66                 /* yes, memory */
67                 if(* (u32 *)addr != 0) { /* alias? */
68 #ifdef BLOB_DEBUG
69                     SerialOutputString("Detected alias at 0x");
70                     SerialOutputHex(addr);
71                     SerialOutputString(", aliased from 0x");
72                     SerialOutputHex(* (u32 *)addr);
73                     SerialOutputByte(' \n');
74 #endif
75                     if(memory_map[i].used)
76                         i++;
77
78                     continue;
79                 }
80
81                 /* not an alias, write the current address */
82                 // Alias가 ,
83                 * (u32 *)addr = addr;
84 #ifdef BLOB_DEBUG
85                 SerialOutputString("Detected memory at 0x");
86                 SerialOutputHex(addr);
87                 SerialOutputByte(' \n');
88 #endif
89                 /* does this start a new block? */    //
90
91                 if(memory_map[i].used == 0) {
92                     memory_map[i].start = addr;
93                     memory_map[i].len = TEST_BLOCK_SIZE;
94                     memory_map[i].used = 1;
95                 } else {
96                     memory_map[i].len += TEST_BLOCK_SIZE;
97                 }
98             } else {
99                 /* no memory here */

```



```

98         if(memory_map[i].used == 1)
99             i++;
100     }
101 }
102
103 SerialOutputString("Memory map: \n");
104 for(i = 0; i < NUM_MEM_AREAS; i++) {
105     if(memory_map[i].used) {
106         SerialOutputString(" 0x");
107         SerialOutputHex(memory_map[i].len);
108         SerialOutputString(" @ 0x");
109         SerialOutputHex(memory_map[i].start);
110         SerialOutputString(" (");
111         SerialOutputDec(memory_map[i].len / (1024 * 1024));
112         SerialOutputString(" MB) \n");
113     }
114 }
115 }
----- memory.c -----

```

(map) memory\_map[] used  
 0 clear . NUM\_MEM\_AREAS  
 가 가 32 가 , memory.h 32  
 alias .  
 memory\_map[] memory\_area\_t .  
 loop 가  
 , testram() 가 . 가  
 C ,  
 가 0  
 , 1 . testmem2.S  
 .

```

----- memory.h -----

34 typedef struct {
35     u32 start;
36     u32 len;
37     int used;
38 } memory_area_t;

----- memory.h -----

```

```

----- main.h -----

37 /* memory start and end */
38 #define MEMORY_START      (0xc0000000)
39 #define MEMORY_END        (0xe0000000)

----- main.h -----

1MBytes      (=TEST_BLOCK_SIZE)      가      block
loop      0      .      (MEMORY_START)
      (MEMORY_END)      0XC0000000      0XE0000000      .
      SA1110      DRAM BANK 0,1,3,4가      .(
10-4      ) (      가      .-_-;;)
testram()      가 0      , RAM
, memory_map[]      used      가 1
i      가      ,      loop      ,      가 alias
      .      memory_map[]      block
used      1      가      i      가
      ,      (addr)      addr      가 가
, alias
      ,      block      (memory_map[i].used == 0),
memory_map[]      element      ,      addr      ,
1MBytes(=TEST_BLCOK_SIZE),      used      1      .
(memory_map[i].used == 1),      가      (TEST_BLOCK_SIZE
).      ,      block      block      .

----- main.c -----

113 /* initialise status */
      //
114 blob_status.kernelSize = 0;      //      0
.
115 blob_status.kernelType = fromFlash;      // fromFlash
.
116 blob_status.ramdiskSize = 0;      //      0
.
117 blob_status.ramdiskType = fromFlash;// fromFlash
.
118 blob_status.blockSize = blockSize;      //      blockSize
blockSize(=0x00800000)
119 blob_status.downloadSpeed = baud115k2;//

```

115200BPS

```
120
121
122  /* Load kernel and ramdisk from flash to RAM */
           // , , flash RAM
123  Reload("blob"); //
124  Reload("kernel"); //
125  Reload("ramdisk"); //
126
----- main.c -----
Reload() . Reload
commandline

----- main.c -----
470 void Reload(char *commandline)
471 {
472     u32 *src = 0;
473     u32 *dst = 0;
474     int numWords;
475
476     if(MyStrNCmp(commandline, "blob", 4) == 0) { //
        'blob'
477         src = (u32 *)BLOB_RAM_BASE;
478         dst = (u32 *)BLOB_START;
479         numWords = BLOB_LEN / 4;
480         blob_status.blobSize = 0;
481         blob_status.blobType = fromFlash;
482         SerialOutputString("Loading blob from flash ");
483     } else if(MyStrNCmp(commandline, "kernel", 6) == 0) { //
        'kernel'
484         src = (u32 *)KERNEL_RAM_BASE;
485         dst = (u32 *)KERNEL_START;
486         numWords = KERNEL_LEN / 4;
487         blob_status.kernelSize = 0;
488         blob_status.kernelType = fromFlash;
489         SerialOutputString("Loading kernel from flash ");
490     } else if(MyStrNCmp(commandline, "ramdisk", 7) == 0) { //
        'ramdisk'
491         src = (u32 *)RAMDISK_RAM_BASE;
492         dst = (u32 *)INITRD_START;
493         numWords = INITRD_LEN / 4;
```

```

494         blob_status.ramdiskSize = 0;
495         blob_status.ramdiskType = fromFlash;
496         SerialOutputString("Loading ramdisk from flash ");
497     } else {
498         SerialOutputString("*** Don't know how to reload \ "");
499         SerialOutputString(commandline);
500         SerialOutputString(" \ " \ n");
501         return;
502     }
503
504     MyMemCpy(src, dst, numWords);
505     SerialOutputString(" done \ n");
506 }
----- main.c -----

        MyStrNCmp()                                , blob, kernel,
ramdisk(      )
    . MyStrNCmp()
        가      . MyStrNCmp()        util.c
        MyMemCpy()

    if      MyStrNCmp()      . 가      "blob"
        ,      src      dst      BLOB_RAM_BASE(=0xC1000000)
BLOB_START(=0x00000000)      가      .(:      Assabet
        (flash.h      ))

        , numWords      BLOB_LEN(=0x10000 or 64K)      4
blob_status.blobSize      0      blobType      fromFlash(=0 or download from flash)
        ,      loading      serial
"loading blob from flash"      .      MyMemCpy()      dst가 가

        commandline      "kernel"      src
KERNEL_RAM_BASE(=0xC0008000(      compile
        0xC0008000      ))      ,      dst
KERNEL_START(=0x10000 or 64K)      . numWords
KERNEL_LEN(=0xC0000 or 3*256K)      4      4byte
        , blob_status.kernelSize      0      blob_status.kernelType
fromFlash      flash      loading
loading      SerialOutputString()      serial      output
        .(      MyMemCpy()가      .)

```

```

----- util.c -----
//
94 int MyStrNCmp(const char *s1, const char *s2, int maxlen)
95 {
96     int i;
97
98     for(i = 0; i < maxlen; i++) {
99         if(s1[i] != s2[i])
100             return ((int) s1[i]) - ((int) s2[i]);
101         if(s1[i] == 0)
102             return 0;
103     }
104
105     return 0;
106 } /* MyStrNCmp */

//
45 void MyMemCpy(u32 *dest, const u32 *src, int numWords)
46 {
47 #ifdef BLOB_DEBUG
48     SerialOutputString(" \n### Now copying 0x");
49     SerialOutputHex(numWords);
50     SerialOutputString(" words from 0x");
51     SerialOutputHex((int)src);
52     SerialOutputString(" to 0x");
53     SerialOutputHex((int)dest);
54     SerialOutputByte(' \n');
55 #endif
56
57     while(numWords--) {
58         if((numWords & 0xffff) == 0x0)
59             SerialOutputByte('.');
60
61         *dest++ = *src++;
62     }
63
64 #ifdef BLOB_DEBUG
65     SerialOutputString(" done \n");
66 #endif
67 } /* MyMemCpy */

```

```

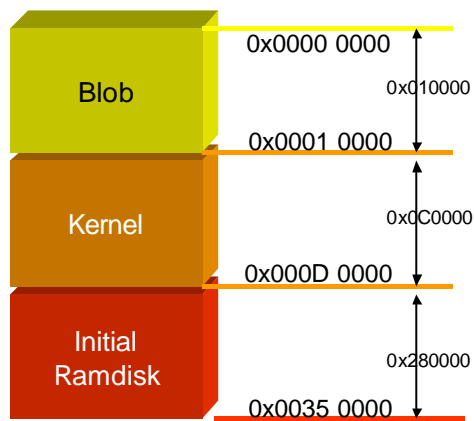
----- util.c -----
MyStrNCmp()
    0
    string index
MyMemCpy() (dest) (src)
    32 loop (copy)
    가 0xFFFF AND 0 "."
    가

----- main.c -----
127 #ifdef BLOB_DEBUG // BLOB_DEBUG가 (
    )
128 /* print some information */
    //
129 SerialOutputString("Running from ");
130 if(RunningFromInternal())
131     SerialOutputString("internal");
132 else
133     SerialOutputString("external");
134
135 SerialOutputString(" flash, blockSize = 0x");
136 SerialOutputHex(blockSize);
137 SerialOutputByte(' \n');
138 #endif

----- main.c -----
    가 memory_map[] element
    loop
SerialOutputHex() (SeralOutputString() SerialOutputByte()

    RAM disk src
RAMDISK_RAM_BASE(=0xC0800000) , dst INITRD_START(Initial RAM
Disk Start = KERNEL_START + KERNEL_LEN) . numWords
INITRD_LAN(=0x280000 or 5*512K, 2.5MBytes ) 4 ,
blob_status.ramdiskSize 0 blob_status.ramdiskType fromFlash ,
flash Reload()
    RAM
    Reload() debugging
    RAM loading flash

```



Reload()

, flash

DRAM

loading

```

----- serial.c -----
129 /*
130  * Write the argument of the function in hexadecimal to the serial
131  * port. If you want "0x" in front of it, you'll have to add it
132  * yourself.
133  */
    // , 16

"0x"가

134 void SerialOutputHex(const u32 h)
135 {
136     char c;
137     int i;
138
139     for(i = NIBBLES_PER_WORD - 1; i >= 0; i--) {
140         c = (char)((h >> (i * 4)) & 0x0f);
141
142         if(c > 9)
143             c += ('A' - 10);
144         else
145             c += '0';
146
147         SerialOutputByte(c);
148     }
149 }

----- serial.c -----
SerialOutputHex()    32bit    16

```

```

SerialOutputByte()
(16

)

main()
----- main.c -----
139  /* wait 10 seconds before starting autoboot */
      // 10
140  SerialOutputString("Autoboot in progress, press any key to stop ");
141  for(i = 0; i < 10; i++) {
142      SerialOutputByte('.');
143
144      retval = SerialInputBlock(commandline, 1, 1);
145
146      if(retval > 0)
147          break;
148  }
149
150  /* no key was pressed, so proceed booting the kernel */
      // 가 ,
151  if(retval == 0) {
152      commandline[0] = '\0';
153      boot_linux(commandline);
154  }
155
      //
156  SerialOutputString("\nAutoboot aborted\n");
157  SerialOutputString("Type \"help\" to get a list of commands\n");
----- main.c -----
SerialOutputString()      Autoboot      for
      , SerialInputBlock()
      , retval 0      , commandline  "\0"
boot_linux()
      SerialInputBlock()      SerialInputByte()

----- serial.c -----

197 /*
198  * Read a single byte from the serial port. Returns 1 on success, 0
199  * otherwise. When the function is succesfull, the character read is

```



```

200  * written into its argument c.
201  */
        //                                .          1          0

        //                                ,          c          .

202 int SerialInputByte(char *c)
203 {
204 #if defined USE_SERIAL1
205     if(Ser1UTSR1 & UTSR1_RNE) {
206         int err = Ser1UTSR1 & (UTSR1_PRE | UTSR1_FRE |
UTSR1_ROR);
207         *c = (char)Ser1UTDR;
208 #elif defined USE_SERIAL3
209     if(Ser3UTSR1 & UTSR1_RNE) {
210         int err = Ser3UTSR1 & (UTSR1_PRE | UTSR1_FRE |
UTSR1_ROR);
211         *c = (char)Ser3UTDR;
212 #else
213 #error "Configuration error: No serial port at all"
214 #endif
215
216     /* If you're lucky, you should be able to use this as
217        * debug information ;- ) -- Erik
218        */
        //                                ,

219     if(err & UTSR1_PRE)
220         SerialOutputByte('@');
221     else if(err & UTSR1_FRE)
222         SerialOutputByte('#');
223     else if(err & UTSR1_ROR)
224         SerialOutputByte('$');
225
226     /* We currently only care about framing and parity errors */
        //      frame      parity      .

227     if((err & (UTSR1_PRE | UTSR1_FRE)) != 0) {
228         return SerialInputByte(c);
229     } else {
230         led_toggle();
231         return(1);
232     }

```

```

233     } else {
234         /* no bit ready */ //   가   가   .
235         return(0);
236     }
237 } /* SerialInputByte */

```

----- serial.c -----

```

SerialInputByte()   serial port   input   .   1
                  ,   0   .   character
c   .   serial port가   가   Ser1UTSSR1   ,
Ser3URSR1   , UTSR1_RNE   Receive FIFO Not Empty
.   receiver FIFO가   ,
input   Ser1UTDR   Ser3UTDR   .   c
.   가   , Ser1UTSR1   Ser3UTSR1   err
.
가   . PRE   Parity Error   ,
FRE   Framing Error   , ROR   Receive Overrun   . Parity error
parity가   가   , frame error   stop bit   1
, 0   , receive overrun   receiver   FIFO가   full
.   SerialOutputByte()
display   .   가 PRE   FRE   ,   SerialInputByte()
,   led_toggle()   led_off()   led_on()
LED   (led_state)   , 1   . Receive FIFO가
,   0   .

```

----- serial.c -----

```

296 /*
297  * SerialInputBlock(): almost the same as SerialInputString(), but
298  * this one just reads a block of characters without looking at
299  * special characters.
300  */
    //
301 int SerialInputBlock(char *buf, int bufsize, const int timeout)
302 {
303     u32 startTime, currentTime;
304     char c;
305     int i;
306     int numRead;

```

```

307     int maxRead = bufsize;
308
309     startTime = TimerGetTime();
310
311     for(numRead = 0, i = 0; numRead < maxRead;) {
312         /* try to get a byte from the serial port */
313         while(!SerialInputByte(&c)) {
314             currentTime = TimerGetTime();
315
316             /* check timeout value */
317             if((currentTime - startTime) >
318                (timeout * TICKS_PER_SECOND)) {
319                 /* timeout! */
320                 return(numRead);
321             }
322         }
323
324         buf[i++] = c;
325         numRead ++;
326     }
327
328     return(numRead);
329 }
----- serial.c -----
SerialInputBlock()          (character)가 ,
        block              .      startTime
(TimeGetTimer())            , for loop
가 return                  .(numRead). Timeout
        SerialInputByte() while loop
        , for              ,          timeout
        TimeGetTime()      time.c
----- time.c -----
74 /* returns the time in 1/TICKS_PER_SECOND seconds */
75 u32 TimerGetTime(void)
76 {
77     /* turn LED always on after one second so the user knows that
78        * the board is on
79        */
80     if((OSCR % TICKS_PER_SECOND) < (TICKS_PER_SECOND >>7))
81         led_on();

```

```

82
83     return((u32) OSCR);
84 }
----- time.c -----

OSCR
    led_on()          LED          . TICKS_PER_SECOND    time.h

-----
45 #define TICKS_PER_SECOND 3686400
-----
TICKS_PER_SECOND    OS

----- main.c -----
159     /* the command loop. endless, of course */
        //
160     for(;;) {
            //                                tbloader
tbloader>가 . blob    NULL
161         DisplayPrompt(NULL);
162
163         /* wait 10 minutes for a command */
164         numRead = GetCommand(commandline, 128, 600);
165
            // blob
            // boot, clock, download, flash, help,
            // reblob, reboot, reload, reset, speed, status. 11

            //
            // *** Unknown command:      가

166         if(numRead > 0) {
167             if(MyStrNCmp(commandline, "boot", 4) == 0) {
168                 boot_linux(commandline + 4);
169             } else if(MyStrNCmp(commandline, "clock", 5) == 0) {
170                 SetClock(commandline + 5);
171             } else if(MyStrNCmp(commandline, "download ", 9) == 0) {
172                 Download(commandline + 9);
173             } else if(MyStrNCmp(commandline, "flash ", 6) == 0) {
174                 Flash(commandline + 6);
175             } else if(MyStrNCmp(commandline, "help", 4) == 0) {

```

```

176             PrintHelp();
177         } else if(MyStrNCmp(commandline, "reblob", 6) == 0) {
178             Reblob();
179         } else if(MyStrNCmp(commandline, "reboot", 6) == 0) {
180             Reboot();
181         } else if(MyStrNCmp(commandline, "reload ", 7) == 0) {
182             Reload(commandline + 7);
183         } else if(MyStrNCmp(commandline, "reset", 5) == 0) {
184             ResetTerminal();
185         } else if(MyStrNCmp(commandline, "speed ", 6) == 0) {
186             SetDownloadSpeed(commandline + 6);
187         }
188     else if(MyStrNCmp(commandline, "status", 6) == 0) {
189         PrintStatus();
190     } else {
191         SerialOutputString("*** Unknown command: ");
192         SerialOutputString(commandline);
193         SerialOutputByte('\n');
194     }
195 }
196 }
197
198 return 0;
199 } /* main */
----- main.c -----
가

----- linux.c -----
49 void boot_linux(char *commandline)
50 {
51     register u32 i;
52     void (*theKernel)(int zero, int arch) = (void (*)(int,
int))KERNEL_RAM_BASE;
53
54     setup_start_tag();
55     setup_memory_tags();
56     setup_commandline_tag(commandline);
57     setup_initrd_tag();
58     setup_ramdisk_tag();
59     setup_end_tag();
60

```

```

61  /* we assume that the kernel is in place */
62  SerialOutputString("\nStarting kernel ... \n\n");
63
64  /* turn off I-cache */
65  asm ("mrc p15, 0, %0, c1, c0, 0": "=r" (i));
66  i &= ~0x1000;
67  asm ("mcr p15, 0, %0, c1, c0, 0": : "r" (i));
68
69  /* flush I-cache */
70  asm ("mcr p15, 0, %0, c7, c5, 0": : "r" (i));
71
72  theKernel(0, ARCH_NUMBER);
73
74  SerialOutputString("Hey, the kernel returned! This should not
happen. \n");
75 }

```

```

----- linux.c -----

boot_linux()                                I-cache(
(SA-1100      6-1      )) OFF      , flush      ,
Linux                                uncompression
code가      가      .
      0      ,                                . Kernel
return      ,      line      SerialOutputString()
      architecture      linux.h
.      source      ~/arch/arm/tools/mach_types
.      architecture      source      ~/arch/arm/boot/compressed/head.S
.

```

```

----- kernel head.S -----

92 start:
93      .type      start,#function
94      .rept      8
95      mov r0, r0
96      .endr
97
98      b      1f
99      .word      0x016f2818      @ Magic numbers to help the loader
//
100     .word      start      @ absolute load/run zImage address //

```

```

zImage
    101      .word  _edata      @ zImage end address      /  /
zImage
    102 1:      mov r7, r1      @ save architecture ID
    //      ID
    103      mov r8, #0      @ save r0
    // r0

----- kernel  head.S -----
head.S      r0  0 , r1  ARCH_NUMBER가  가
,      r7  가 , architecture ID
setup_()      . setup_()
.      x86  LILO

tag

/include/asm-arm/setup.h      가

----- kernel  setup.h -----
202 struct tag {
203     struct tag_header hdr;
204     union {
205         struct tag_core    core;
206         struct tag_mem32    mem;
207         struct tag_videotext videotext;
208         struct tag_ramdisk  ramdisk;
209         struct tag_initrd   initrd;
210         struct tag_serialnr serialnr;
211         struct tag_revision revision;
212         struct tag_videofb videofb;
213         struct tag_cmdline  cmdline;
214
215         /*
216          * Acorn specific
217          */
218         struct tag_acorn    acorn;
219
220         /*
221          * DC21285 specific
222          */
223         struct tag_memclk   memclk;

```

```

224     } u;
225 };
----- kernel    setup.h    -----
                                tag                                ,
                                                                .
                                                                main.h

BOOT_PARAMS(=0XC0000100)    가    .
----- main.h    ,-----
#define BOOT_PARAMS (0xc0000100)    // 77    .
----- main.h    ,-----

----- linux.c    ,-----
78 static void setup_start_tag(void)
79 {
80     params = (struct tag *)BOOT_PARAMS;
81
82     params->hdr.tag = ATAG_CORE;
83     params->hdr.size = tag_size(tag_core);
84
85     params->u.core.flags = 0;
86     params->u.core.pagesize = 0;
87     params->u.core.rootdev = 0;
88
89     params = tag_next(params);
90 }
----- linux.c    ,-----

                                BOOT_PARAMS    가    tag    .
ATAG_CORE    kernel/include/asm-arm/setup.h    0x54410001
                                가    .(    가    .)
tag_core    (tag_size()), union    core
                                flags    0, pagesize    0, rootdev    0    .    tag
                                tag_next()    .(    : tag_size()    tag_next()
                                                                .
                                                                2.4.X

~/include/asm-arm/setup.h    .
                                BLOB    가    .

----- kernel    setup.h    -----
240 #define tag_next(t) ((struct tag *)((u32 *) (t) + (t)->hdr.size))
241 #define tag_size(type) ((sizeof(struct tag_header) + sizeof(struct type))
>> 2)
----- kernel    setup.h    -----

```



tag_next()	tag		, tag_size()
tag header	type	4	4byte

----- linux.c -----

```

93 static void setup_memory_tags(void)
94 {
95     int i;
96
97     for(i = 0; i < NUM_MEM_AREAS; i++) {
98         if(memory_map[i].used) {
99             params->hdr.tag = ATAG_MEM;
100             params->hdr.size = tag_size(tag_mem32);
101
102             params->u.mem.start = memory_map[i].start;
103             params->u.mem.size = memory_map[i].len;
104
105             params = tag_next(params);
106         }
107     }
108 }

```

----- linux.c -----

setup_memory_tags()	가
Tag header ATAG_MEM tag	,
(memory_map[]) entry	.

----- linux.c -----

```

111 static void setup_commandline_tag(char *commandline)
112 {
113     char *p;
114
115     /* eat leading white space */ // . ( )
116     for(p = commandline; *p == ' '; p++)
117         ;
118
119     /* skip non-existent command lines so the kernel will still
120        * use its default command line.
121        */
122     if(*p == '\0')

```

```

123         return;
124
125     params->hdr.tag = ATAG_CMDLINE;
126     params->hdr.size = (sizeof(struct tag_header) + strlen(p) + 1 + 4)
>> 2;
127
128     strcpy(params->u.cmdline.cmdline, p);
129
130     params = tag_next(params);
131 }

```

```

----- linux.c -----
tag
blank
" \ 0"
tag
blank
header
ATAG_CMDLINE
blank

```

```

----- linux.c -----
134 static void setup_initrd_tag(void)
135 {
136     /* an ATAG_INITRD node tells the kernel where the compressed
137        * ramdisk can be found. ATAG_RDIMG is a better name,
actually.
138        */
        // ATAG_INITRD
        // ATAG_INITRD
139     params->hdr.tag = ATAG_INITRD;
140     params->hdr.size = tag_size(tag_initrd);
141
142     params->u.initrd.start = RAMDISK_RAM_BASE;
143     params->u.initrd.size = INITRD_LEN;
144
145     params = tag_next(params);
146 }

```

```

----- linux.c -----
setup_initrd_tag() initial tag . ATAG_INITRD tag
head , RAMDISK_RAM_BASE(=0xC0800000) initial

```

```

,          INITRD_LEN(=0X280000)          .

----- linux.c -----
149 static void setup_ramdisk_tag(void)
150 {
151     /* an ATAG_RAMDISK node tells the kernel how large the
152        * decompressed ramdisk will become.
153        */
154        // ATAG_INITRD          가
155
156
157     params->hdr.tag = ATAG_RAMDISK;
158     params->hdr.size = tag_size(tag_ramdisk);
159     params->u.ramdisk.start = 0;
160     params->u.ramdisk.size = RAMDISK_SIZE;
161     params->u.ramdisk.flags = 1;    /* automatically load ramdisk */
162     //
163     params = tag_next(params);
164 }

----- linux.c -----
setup_ramdisk_tag()          tag          .          0
,          RAMDISK_SIZE(=8*1024*1024 or 8MBytes)          ,
          flags 1          .

----- linux.c -----
165 static void setup_end_tag(void)
166 {
167     params->hdr.tag = ATAG_NONE;
168     params->hdr.size = 0;
169 }

----- linux.c -----
tag          setup_end_tag()          . ATAG_NONE tag
header          , tag          가 0          tag

.

tag          가
kernel/arch/arm/kernel/setup.c parse_tag_()
tag tag가          , core,          ,
initrd,          가          .

```

```

----- kernel/arch/arm/kernel/setup.c -----
311 /*
312  * Tag parsing.
313  *
314  * This is the new way of passing data to the kernel at boot time.
Rather
315  * than passing a fixed inflexible structure to the kernel, we pass a list
316  * of variable-sized tags to the kernel. The first tag must be a
ATAG_CORE
317  * tag for the list to be recognised (to distinguish the tagged list from
318  * a param_struct). The list is terminated with a zero-length tag (this
tag
319  * is not parsed in any way).
320 */
    //
    //
    //
    //
    // ATAG_CORE
aram_struct
    // zero-length
321 static int __init parse_tag_core(const struct tag *tag)
322 {
323     if (tag->hdr.size > 2) {
324         if ((tag->u.core.flags & 1) == 0)
325             root_mountflags &= ~MS_RDONLY;
326         ROOT_DEV = to_kdev_t(tag->u.core.rootdev);
327     }
328     return 0;
329 }
330
331 __tagtable(ATAG_CORE, parse_tag_core);
332
333 static int __init parse_tag_mem32(const struct tag *tag)
334 {
335     if (meminfo.nr_banks >= NR_BANKS) {
336         printk(KERN_WARNING
337             "Ignoring memory bank 0x%08x size %dKB \n",
338             tag->u.mem.start, tag->u.mem.size / 1024);
339         return -EINVAL;

```

```

340     }
341     meminfo.bank[meminfo.nr_banks].start = tag->u.mem.start;
342     meminfo.bank[meminfo.nr_banks].size  = tag->u.mem.size;
343     meminfo.bank[meminfo.nr_banks].node  =
PHYS_TO_NID(tag->u.mem.start);
344     meminfo.nr_banks += 1;
345
346     return 0;
347 }
348
349 __tagtable(ATAG_MEM, parse_tag_mem32);
350
351     #if                defined(CONFIG_VGA_CONSOLE)                ||
defined(CONFIG_DUMMY_CONSOLE)
352     struct screen_info screen_info = {
353     orig_video_lines: 30,
354     orig_video_cols:  80,
355     orig_video_mode:  0,
356     orig_video_ega_bx: 0,
357     orig_video_isVGA: 1,
358     orig_video_points: 8
359 };
360
361 static int __init parse_tag_videotext(const struct tag *tag)
362 {
363     screen_info.orig_x      = tag->u.videotext.x;
364     screen_info.orig_y      = tag->u.videotext.y;
365     screen_info.orig_video_page = tag->u.videotext.video_page;
366     screen_info.orig_video_mode = tag->u.videotext.video_mode;
367     screen_info.orig_video_cols = tag->u.videotext.video_cols;
368     screen_info.orig_video_ega_bx = tag->u.videotext.video_ega_bx;
369     screen_info.orig_video_lines = tag->u.videotext.video_lines;
370     screen_info.orig_video_isVGA = tag->u.videotext.video_isvga;
371     screen_info.orig_video_points = tag->u.videotext.video_points;
372     return 0;
373 }
374
375 __tagtable(ATAG_VIDEOTEXT, parse_tag_videotext);
376 #endif
377
378 static int __init parse_tag_ramdisk(const struct tag *tag)

```

```

379 {
380     setup_ramdisk((tag->u.ramdisk.flags & 1) == 0,
381                 (tag->u.ramdisk.flags & 2) == 0,
382                 tag->u.ramdisk.start, tag->u.ramdisk.size);
383     return 0;
384 }
385
386 __tagtable(ATAG_RAMDISK, parse_tag_ramdisk);
387
388 static int __init parse_tag_initrd(const struct tag *tag)
389 {
390     setup_initrd(tag->u.initrd.start, tag->u.initrd.size);
391     return 0;
392 }
393
394 __tagtable(ATAG_INITRD, parse_tag_initrd);
395
396 static int __init parse_tag_serialnr(const struct tag *tag)
397 {
398     system_serial_low = tag->u.serialnr.low;
399     system_serial_high = tag->u.serialnr.high;
400     return 0;
401 }
402
403 __tagtable(ATAG_SERIAL, parse_tag_serialnr);
404
405 static int __init parse_tag_revision(const struct tag *tag)
406 {
407     system_rev = tag->u.revision.rev;
408     return 0;
409 }
410
411 __tagtable(ATAG_REVISION, parse_tag_revision);
412
413 static int __init parse_tag_cmdline(const struct tag *tag)
414 {
415     strncpy(default_command_line, tag->u.cmdline.cmdline,
COMMAND_LINE_SIZE);
416     default_command_line[COMMAND_LINE_SIZE - 1] = '\0';
417     return 0;
418 }

```

```

419
420 __tagtable(ATAG_CMDLINE, parse_tag_cmdline);
421
----- kernel/arch/arm/kernel/setup.c . -----
    main.c
----- main.c .-----
159     /* the command loop. endless, of course */
    //
160     for(;;) {
    //                                     tloader      tloader>가
blob  NULL
161         DisplayPrompt(NULL);
162
163         /* wait 10 minutes for a command */
164         numRead = GetCommand(commandline, 128, 600);
165
                                // blob
                                // boot, clock, download, flash, help,
                                // reblob, reboot, reload, reset, speed, status. 11

                                //
                                // *** Unknown command:      가

166         if(numRead > 0) {
167             if(MyStrNCmp(commandline, "boot", 4) == 0) {
168                 boot_linux(commandline + 4);
169             } else if(MyStrNCmp(commandline, "clock", 5) == 0) {
170                 SetClock(commandline + 5);
171             } else if(MyStrNCmp(commandline, "download ", 9) == 0) {
172                 Download(commandline + 9);
173             } else if(MyStrNCmp(commandline, "flash ", 6) == 0) {
174                 Flash(commandline + 6);
175             } else if(MyStrNCmp(commandline, "help", 4) == 0) {
176                 PrintHelp();
177             } else if(MyStrNCmp(commandline, "reblob", 6) == 0) {
178                 Reblob();
179             } else if(MyStrNCmp(commandline, "reboot", 6) == 0) {
180                 Reboot();
181             } else if(MyStrNCmp(commandline, "reload ", 7) == 0) {
182                 Reload(commandline + 7);
183             } else if(MyStrNCmp(commandline, "reset", 5) == 0) {

```

```

184             ResetTerminal();
185         } else if(MyStrNCmp(commandline, "speed ", 6) == 0) {
186             SetDownloadSpeed(commandline + 6);
187         }
188         else if(MyStrNCmp(commandline, "status", 6) == 0) {
189             PrintStatus();
190         } else {
191             SerialOutputString("*** Unknown command: ");
192             SerialOutputString(commandline);
193             SerialOutputByte('\n');
194         }
195     }
196 }
197
198 return 0;
199 } /* main */
----- main.c -----
autoboot , main() for
. DisplayPrompt() 가
.
. command.c
GetCommand() commandline ,
가 (MySTRNCmp()), loop
boot, clock, download, flash, help, reblob, reboot, reload, reset,
speed, status. 11 가 , Unknown command
. DisplayPrompt() GetCommand() ,
.

----- command.c -----
46 /* display a prompt, or the standard prompt if prompt == NULL */
47 void DisplayPrompt(char *prompt)
48 {
49     if(prompt == NULL) {
50         SerialOutputString(PACKAGE "> ");
51     } else {
52         SerialOutputString(prompt);
53     }
54 }

----- command.c -----
DisplayPrompt() ">"
prompt NULL ">" ,
prompt ,

```



```

----- command.c -----
60 int GetCommand(char *command, int len, int timeout)
61 {
62     u32 startTime, currentTime;
63     char c;
64     int i;
65     int numRead;
66     int maxRead = len - 1;
67
68     TimerClearOverflow();
69
70     startTime = TimerGetTime();
71
72     for(numRead = 0, i = 0; numRead < maxRead;) {
73         /* try to get a byte from the serial port */
74         //
75         while(!SerialInputByte(&c)) {
76             currentTime = TimerGetTime();
77             /* check timeout value */ // timeout
78             if((currentTime - startTime) >
79                (timeout * TICKS_PER_SECOND)) {
80                 /* timeout */
81                 command[i++] = '\0';
82                 return(numRead);
83             }
84         }
85
86         if((c == '\r') || (c == '\n')) {
87             command[i++] = '\0';
88
89             /* print newline */
90             SerialOutputByte('\n');
91             return(numRead);
92         } else if(c == '\b') { /* FIXME: is this backspace? */ //
93
94             if(i > 0) {
95                 i--;
96                 numRead--;
97                 /* cursor one position back. */ //

```

```

97         SerialOutputString(" \ b \ b");
98     }
99     } else {
100         command[i++] = c;
101         numRead++;
102
103         /* print character */
104         SerialOutputByte(c);
105     }
106 }
107
108 return(numRead);
109 }
----- command.c -----

GetCommand()                                input
TimerClearOverflow()                        , timer가 overflow가
time.c                                     .

----- time.c -----

89 int TimerDetectOverflow(void)
90 {
91     return(OSSR & OSSR_M0);
92 }
93
94
95
96 void TimerClearOverflow(void)
97 {
98     if(TimerDetectOverflow())
99         numOverflows++;
100
101     OSSR = OSSR_M0;
102 }
----- time.c -----

TimerClearOverflow()    TimerDetectOverflow()    overflow가
                        , numOverflows            가
OSSR(OS Status Register)  OSMR(OS Match Register) 4
                        OSCR(OS Counter Register)

```

```

OSMR0      가      (OSSR_MO = 0x00000001), overflow가
. ,      가      timer      mach register      OSMR0
      (TimerInit()). for
      (TimerGetTime())startTime      ,
      timeout      .
for      byte      , timeout
byte      ,      " \ r"      " \ n"      ,
.      SerialInputByte()      .
, timeout      ,      timeout      ,      byte
.      가 " \ b"(      )
,      " \ b"      ,      .
      echo      .

```

3.

- ARM7 ( )
- BLOB .

4.

가 .  
- ([greendrm@netsgo.com](mailto:greendrm@netsgo.com)) -

#### 4.1. load, store, move

```

load      가      , store
.
      ,      가      move
.

```

```

ldr r0, [r1]
r1      가 가
      r0      가      . C      r0 = *(r1)가      .

```

```

ldr r0, [r1, #4]
r1+4      r0      가      . C      r0 = *(r1 + 4)가
(pre-incrementing index).

```

```

ldr r0, [r1], #4
r1      r0      가      , r1      4      가      . C      r0 =
*r1; r1 = r1 + 4가      (post-incrementing index).

```

```

str r0, [r1]
r0 = r1 . C          *(r1) = r0가 .

str r0, [r1, #4]
r0 = r1 + 4 . C          *(r1 + 4) = r0가
(pre-incrementing index).

str r0, [r1], #4
r0 = r1 , r1 = 4 가 . C          *(r1) = r0;
r1 = r1 + 4가 (post-incrementing index).

mov r0, r1
r1 = r0 . C          r0 = r1 .

```