```
*********************************************************
     659 Sat Feb  7 18:57:29 2015
new/usr/src/uts/armv6/bcm2835/Makefile.files
bcm2835: use the real uart instead of the mini-uart
The real uart is more capable.  We'll want to use it for the real console
eventually anyway, so let's bite the bullet now when no one will really
notice.  (For comparison, the Linux kernel uses the real uart and totally
lacks a driver for the miniuart.)
*********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright (c) 2013, Joyent, Inc.  All rights reserved.
# Copyright (c) 2015, Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
#

BCM2835_OBJS =                  \
        bcm2835_bsmdep.o        \
        bcm2835_uart.o          \
#endif /* ! codereview */
        boot_console.o          \
        locore.o
        locore.o                \
        miniuart.o

BCM2835_LOADER_OBJS = \
        bcm2835_ldep.o
```

```
**********************************************************
    4202 Sat Feb  7 18:57:29 2015
new/usr/src/uts/armv6/bcm2835/loader/bcm2835_ldep.c
bcm2835: use the real uart instead of the mini-uart
The real uart is more capable.  We'll want to use it for the real console
eventually anyway, so let's bite the bullet now when no one will really
notice.  (For comparison, the Linux kernel uses the real uart and totally
lacks a driver for the miniuart.)
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
 * Copyright (c) 2015 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */

#include <sys/elf.h>
#include <sys/atag.h>

/*
 * The primary serial console that we end up using is the normal UART, not
 * the mini-uart that shares interrupts and registers with the SPI masters
 * as well.
 * The primary serial console that we end up using is not in fact a normal UART,
 * but is instead actually a mini-uart that shares interrupts and registers with
 * the SPI masters as well. While the RPi also supports another more traditional
 * UART, that isn't what we are actually hooking up to generally with the
 * adafruit cable. We already wasted our time having to figure that out. -_-
 */

#define UART_BASE               0x20201000
#define UART_DR                 0x0
#define UART_FR                 0x18
#define UART_IBRD               0x24
#define UART_FBRD               0x28
#define UART_LCRH               0x2c
#define UART_CR                 0x30
#define UART_ICR                0x44

#define UART_FR_RXFE            0x10            /* RX fifo empty */
#define UART_FR_TXFF            0x20            /* TX fifo full */

#define UART_LCRH_FEN           0x00000010      /* fifo enable */
#define UART_LCRH_WLEN_8        0x00000060      /* 8 bits */

#define UART_CR_UARTEN          0x001           /* uart enable */
#define UART_CR_TXE             0x100           /* TX enable */
#define UART_CR_RXE             0x200           /* RX enable */
#define AUX_BASE        0x20215000
#define AUX_ENABLES     0x4
#define AUX_MU_IO_REG   0x40
#define AUX_MU_IER_REG  0x44
#define AUX_MU_IIR_REG  0x48
#define AUX_MU_LCR_REG  0x4C
#define AUX_MU_MCR_REG  0x50
#define AUX_MU_LSR_REG  0x54
#define AUX_MU_CNTL_REG 0x60
```



```
#define AUX_MU_BAUD     0x68

#define AUX_MU_RX_READY 0x01
#define AUX_MU_TX_READY 0x20

/*
 * All we care about are pins 14 and 15 for the UART.  Specifically, alt0
 * for GPIO14 is TXD0 and GPIO15 is RXD0. Those are controlled by FSEL1.
 * For the mini UART, all we care about are pins 14 and 15 for the UART.
 * Specifically, alt5 for GPIO14 is TXD1 and GPIO15 is RXD1. Those are
 * controlled by FSEL1.
 */
#define GPIO_BASE       0x20200000
#define GPIO_FSEL1      0x4
#define GPIO_PUD        0x94
#define GPIO_PUDCLK0    0x98

#define GPIO_SEL_ALT0   0x4
#define GPIO_SEL_ALT5   0x2
#define GPIO_UART_MASK  0xfffc0fff
#define GPIO_UART_TX_SHIFT      12
#define GPIO_UART_RX_SHIFT      15

#define GPIO_PUD_DISABLE        0x0
#define GPIO_PUDCLK_UART        0x0000c000

static __GNU_INLINE uint32_t arm_reg_read(uint32_t reg)
{
        volatile uint32_t *ptr = (volatile uint32_t *)reg;

        return *ptr;
}
______unchanged_portion_omitted_

/*
 * A simple nop
 */
static void
uart_nop(void)
bcm2835_miniuart_nop(void)
{
        __asm__ volatile("mov r0, r0\n" : : :);
}
______unchanged_portion_omitted_

void
fakeload_backend_init(void)
{
        uint32_t v;
        int i;

        /* disable UART */
        arm_reg_write(UART_BASE + UART_CR, 0);
        /* Enable the mini UAT */
        arm_reg_write(AUX_BASE + AUX_ENABLES, 0x1);

        /* Disable interrupts */
        arm_reg_write(AUX_BASE + AUX_MU_IER_REG, 0x0);

        /* Disable the RX and TX */
        arm_reg_write(AUX_BASE + AUX_MU_CNTL_REG, 0x0);

        /*
         * Enable 8-bit word length. External sources tell us the PRM is buggy
         * here and that even though bit 1 is reserved, we need to actually set
         * it to get 8-bit words.
```

```
113          */
114         arm_reg_write(AUX_BASE + AUX_MU_LCR_REG, 0x3);

116         /* Set RTS high */
117         arm_reg_write(AUX_BASE + AUX_MU_MCR_REG, 0x0);

119         /* Disable interrupts */
120         arm_reg_write(AUX_BASE + AUX_MU_IER_REG, 0x0);

122         /* Set baud rate */
123         arm_reg_write(AUX_BASE + AUX_MU_IIR_REG, 0xc6);
124         arm_reg_write(AUX_BASE + AUX_MU_BAUD, 0x10e);

106         /* TODO: Factor out the gpio bits */
107         v = arm_reg_read(GPIO_BASE + GPIO_FSEL1);
108         v &= GPIO_UART_MASK;
109         v |= GPIO_SEL_ALT0 << GPIO_UART_RX_SHIFT;
110         v |= GPIO_SEL_ALT0 << GPIO_UART_TX_SHIFT;
129         v |= GPIO_SEL_ALT5 << GPIO_UART_RX_SHIFT;
130         v |= GPIO_SEL_ALT5 << GPIO_UART_TX_SHIFT;
111         arm_reg_write(GPIO_BASE + GPIO_FSEL1, v);

113         arm_reg_write(GPIO_BASE + GPIO_PUD, GPIO_PUD_DISABLE);
114         for (i = 0; i < 150; i++)
115                 uart_nop();
135                 bcm2835_miniuart_nop();
116         arm_reg_write(GPIO_BASE + GPIO_PUDCLK0, GPIO_PUDCLK_UART);
117         for (i = 0; i < 150; i++)
118                 uart_nop();
138                 bcm2835_miniuart_nop();
139         // XXX: GPIO_PUD_DISABLE again?
119         arm_reg_write(GPIO_BASE + GPIO_PUDCLK0, 0);

121         /* clear all interrupts */
122         arm_reg_write(UART_BASE + UART_ICR, 0x7ff);

124         /* set the baud rate */
125         arm_reg_write(UART_BASE + UART_IBRD, 1);
126         arm_reg_write(UART_BASE + UART_FBRD, 40);

128         /* select 8-bit, enable FIFOs */
129         arm_reg_write(UART_BASE + UART_LCRH, UART_LCRH_WLEN_8 | UART_LCRH_FEN);

131         /* enable UART */
132         arm_reg_write(UART_BASE + UART_CR, UART_CR_UARTEN | UART_CR_TXE |
133             UART_CR_RXE);
142         /* Finally, go back and enable RX and TX */
143         arm_reg_write(AUX_BASE + AUX_MU_CNTL_REG, 0x3);
134 }

136 void
137 fakeload_backend_putc(int c)
138 {
139         if (c == '\n')
140                 fakeload_backend_putc('\r');

142         while (arm_reg_read(UART_BASE + UART_FR) & UART_FR_TXFF)
143                 ;
144         arm_reg_write(UART_BASE + UART_DR, c & 0x7f);
145         if (c == '\n')
146                 fakeload_backend_putc('\r');
152         for (;;) {
153                 if (arm_reg_read(AUX_BASE + AUX_MU_LSR_REG) & AUX_MU_TX_READY)
154                         break;
155         }
156         arm_reg_write(AUX_BASE + AUX_MU_IO_REG, c & 0x7f);
```



```
147 }

149 /*
150  * Add a map for the uart.
151  */
152 void
153 fakeload_backend_addmaps(atag_header_t *chain)
154 {
155         atag_illumos_mapping_t aim;

157         aim.aim_header.ah_size = ATAG_ILLUMOS_MAPPING_SIZE;
158         aim.aim_header.ah_tag = ATAG_ILLUMOS_MAPPING;
159         aim.aim_paddr = GPIO_BASE;
160         aim.aim_vaddr = GPIO_BASE;
161         aim.aim_vlen = 0x1000;
162         aim.aim_plen = 0x1000;
163         aim.aim_mapflags = PF_R | PF_W | PF_NORELOC | PF_DEVICE;
164         atag_append(chain, &aim.aim_header);

166         aim.aim_header.ah_size = ATAG_ILLUMOS_MAPPING_SIZE;
167         aim.aim_header.ah_tag = ATAG_ILLUMOS_MAPPING;
168         aim.aim_paddr = UART_BASE;
169         aim.aim_vaddr = UART_BASE;
178         aim.aim_paddr = AUX_BASE;
179         aim.aim_vaddr = AUX_BASE;
170         aim.aim_vlen = 0x1000;
171         aim.aim_plen = 0x1000;
172         aim.aim_mapflags = PF_R | PF_W | PF_NORELOC | PF_DEVICE;
173         atag_append(chain, &aim.aim_header);
174 }
______unchanged_portion_omitted_
```

```
**********************************************************
    3422 Sat Feb  7 18:57:29 2015
new/usr/src/uts/armv6/bcm2835/os/bcm2835_uart.c
bcm2835: use the real uart instead of the mini-uart
The real uart is more capable.  We'll want to use it for the real console
eventually anyway, so let's bite the bullet now when no one will really
notice.  (For comparison, the Linux kernel uses the real uart and totally
lacks a driver for the miniuart.)
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2013 (c) Joyent, Inc.  All rights reserved.
 * Copyright 2015 (c) Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */

/*
 * A simple uart driver for the RPi.
 */
#include <sys/types.h>

#include "bcm2835_uart.h"

extern uint32_t arm_reg_read(uint32_t);
extern void arm_reg_write(uint32_t, uint32_t);

/*
 * The primary serial console that we end up using is the normal UART, not
 * the mini-uart that shares interrupts and registers with the SPI masters
 * as well.
 */

#define UART_BASE               0x20201000
#define UART_DR                 0x0
#define UART_FR                 0x18
#define UART_IBRD               0x24
#define UART_FBRD               0x28
#define UART_LCRH               0x2c
#define UART_CR                 0x30
#define UART_ICR                0x44

#define UART_FR_RXFE            0x10            /* RX fifo empty */
#define UART_FR_TXFF            0x20            /* TX fifo full */

#define UART_LCRH_FEN           0x00000010      /* fifo enable */
#define UART_LCRH_WLEN_8        0x00000060      /* 8 bits */

#define UART_CR_UARTEN          0x001           /* uart enable */
#define UART_CR_TXE             0x100           /* TX enable */
#define UART_CR_RXE             0x200           /* RX enable */


/*
 * All we care about are pins 14 and 15 for the UART.  Specifically, alt0
 * for GPIO14 is TXD0 and GPIO15 is RXD0. Those are controlled by FSEL1.
 */
#define GPIO_BASE       0x20200000
```



```
#define GPIO_FSEL1      0x4
#define GPIO_PUD        0x94
#define GPIO_PUDCLK0    0x98

#define GPIO_SEL_ALT0   0x4
#define GPIO_UART_MASK  0xfffc0fff
#define GPIO_UART_TX_SHIFT      12
#define GPIO_UART_RX_SHIFT      15

#define GPIO_PUD_DISABLE        0x0
#define GPIO_PUDCLK_UART        0x0000c000

/*
 * A simple nop
 */
static void
bcm2835_uart_nop(void)
{
        __asm__ volatile("mov r0, r0\n" : : :);
}

void
bcm2835_uart_init(void)
{
        uint32_t v;
        int i;

        /* disable UART */
        arm_reg_write(UART_BASE + UART_CR, 0);

        /* TODO: Factor out the gpio bits */
        v = arm_reg_read(GPIO_BASE + GPIO_FSEL1);
        v &= GPIO_UART_MASK;
        v |= GPIO_SEL_ALT0 << GPIO_UART_RX_SHIFT;
        v |= GPIO_SEL_ALT0 << GPIO_UART_TX_SHIFT;
        arm_reg_write(GPIO_BASE + GPIO_FSEL1, v);

        arm_reg_write(GPIO_BASE + GPIO_PUD, GPIO_PUD_DISABLE);
        for (i = 0; i < 150; i++)
                bcm2835_uart_nop();
        arm_reg_write(GPIO_BASE + GPIO_PUDCLK0, GPIO_PUDCLK_UART);
        for (i = 0; i < 150; i++)
                bcm2835_uart_nop();
        arm_reg_write(GPIO_BASE + GPIO_PUDCLK0, 0);

        /* clear all interrupts */
        arm_reg_write(UART_BASE + UART_ICR, 0x7ff);

        /* set the baud rate */
        arm_reg_write(UART_BASE + UART_IBRD, 1);
        arm_reg_write(UART_BASE + UART_FBRD, 40);

        /* select 8-bit, enable FIFOs */
        arm_reg_write(UART_BASE + UART_LCRH, UART_LCRH_WLEN_8 | UART_LCRH_FEN);

        /* enable UART */
        arm_reg_write(UART_BASE + UART_CR, UART_CR_UARTEN | UART_CR_TXE |
            UART_CR_RXE);
}

void
bcm2835_uart_putc(uint8_t c)
{
        while (arm_reg_read(UART_BASE + UART_FR) & UART_FR_TXFF)
                ;
        arm_reg_write(UART_BASE + UART_DR, c & 0x7f);
```

```
124         if (c == '\n')
125                 bcm2835_uart_putc('\r');
126 }

128 uint8_t
129 bcm2835_uart_getc(void)
130 {
131         while (arm_reg_read(UART_BASE + UART_FR) & UART_FR_RXFE)
132                 ;
133         return (arm_reg_read(UART_BASE + UART_DR) & 0x7f);
134 }

136 int
137 bcm2835_uart_isc(void)
138 {
139         return ((arm_reg_read(UART_BASE + UART_FR) & UART_FR_RXFE) == 0);
140 }
141 #endif /* ! codereview */
```

```
**********************************************************
     884 Sat Feb  7 18:57:29 2015
new/usr/src/uts/armv6/bcm2835/os/bcm2835_uart.h
bcm2835: use the real uart instead of the mini-uart
The real uart is more capable.  We'll want to use it for the real console
eventually anyway, so let's bite the bullet now when no one will really
notice.  (For comparison, the Linux kernel uses the real uart and totally
lacks a driver for the miniuart.)
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2013 (c) Joyent, Inc.  All rights reserved.
 * Copyright 2015 (c) Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */

#ifndef _BCM2835_UART_H
#define _BCM2835_UART_H

/*
 * Interface to the BCM2835's uart.
 */

#ifdef __cplusplus
extern "C" {
#endif

#include <sys/types.h>

void bcm2835_uart_init(void);
void bcm2835_uart_putc(uint8_t);
uint8_t bcm2835_uart_getc(void);
int bcm2835_uart_isc(void);

#ifdef __cplusplus
}
#endif

#endif /* _BCM2835_UART_H */
#endif /* ! codereview */
```

```
**********************************************************
```
1609 Sat Feb 7 18:57:29 2015
new/usr/src/uts/armv6/bcm2835/os/boot_console.c
bcm2835: use the real uart instead of the mini-uart
The real uart is more capable. We'll want to use it for the real console
eventually anyway, so let's bite the bullet now when no one will really
notice. (For comparison, the Linux kernel uses the real uart and totally
lacks a driver for the miniuart.)
```
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
 */

/*
 * bcm2835 boot console implementation
 */

#include "bcm2835_uart.h"
#include "miniuart.h"

/*
 * There are a few different potential boot consoles that we could have on the
 * bcm2835. There is both a mini uart and a full functioning uart. Generally,
 * people will use one of them, but we want to support both. As such we have a
 * people will use the mini uart, but we want to support both. As such we have a
 * single global ops vector that we set once during bcons_init and never again.
 */
#define BMC2835_CONSNAME_MAX    24
typedef struct bcm2835_consops {
        char bco_name[BMC2835_CONSNAME_MAX];
        void (*bco_putc)(uint8_t);
        uint8_t (*bco_getc)(void);
        int (*bco_isc)(void);
} bcm2835_consops_t;

static bcm2835_consops_t consops;

/*
 * For now, we only support the real uart.
 * For now, we only support the mini uart.
 */
void
bcons_init(char *bstr)
{
        bcm2835_uart_init();
        consops.bco_putc = bcm2835_uart_putc;
        consops.bco_getc = bcm2835_uart_getc;
        consops.bco_isc = bcm2835_uart_isc;
        bcm2835_miniuart_init();
        consops.bco_putc = bcm2835_miniuart_putc;
        consops.bco_getc = bcm2835_miniuart_getc;
        consops.bco_isc = bcm2835_miniuart_isc;
}
```
_____***unchanged_portion_omitted_***