Miyoo

移植XBoot

```
--- src/arch/arm32/mach-f1c100s/driver/fb-f1c100s_orig.c	2018-12-30 22:22:53.370788418 +0800
+++ src/arch/arm32/mach-f1c100s/driver/fb-f1c100s.c 2018-12-31 19:45:36.824995967 +0800
@@ -30,6 +30,7 @@
  #include <led/led.h>
  #include <framebuffer/framebuffer.h>
  #include <f1c100s-gpio.h>
+#include <f1c100s/reg-gpio.h>
  #include <f1c100s/reg-tcon.h>
  #include <f1c100s/reg-defe.h>
  #include <f1c100s/reg-debe.h>
@@ -39,6 +40,7 @@
  	virtual_addr_t virtdefe;
  	virtual_addr_t virtdebe;
  	virtual_addr_t virttcon;
+	virtual_addr_t virtgpio;

  	char * clkdefe;
  	char * clkdebe;
@@ -53,6 +55,7 @@
  	int bits_per_pixel;
  	int bytes_per_pixel;
  	int index;
+	int lcdc_use_gpio;
  	void * vram[2];

  	struct {
@@ -73,6 +76,46 @@
  	int brightness;
  };

+static void sunxi_lcdc_output(struct fb_f1c100s_pdata_t * pdat, uint32_t is_data, uint32_t val)
+{
+	uint32_t ret;
+
+	if (pdat->lcdc_use_gpio){
+		ret = (val & 0x00ff) << 1;
+		ret|= (val & 0xff00) << 2;
+		ret|= is_data ? 0x80000 : 0;
+		ret|= 0x100000;
+		write32(pdat->virtgpio + PD_DATA, ret);
+		ret|= 0x40000;
+		write32(pdat->virtgpio + PD_DATA, ret);
+	}
+	else {
+		ret = read32(pdat->virttcon + TCON0_CPU_IF_REG);
+		if (is_data) {
+			ret|= 0x2000000;
+		}
+		else {
+			ret&= 0xfdffffff;
+		}
+		write32(pdat->virttcon + TCON0_CPU_IF_REG, ret);
+		while(read32(pdat->virttcon + TCON0_CPU_IF_REG) & 0xc0000){
+			mdelay(1);
+		}
+		write32(pdat->virttcon + TCON0_CPU_WR_REG, ((val & 0xfc00) << 8) | ((val & 0x0300) << 6) | ((val & 0x00e0) << 5) | ((val & 0x001f) << 3));
+		//write32(pdat->virttcon + TCON0_CPU_WR_REG, val);
+	}
+}
+
+static void r61520_lcd_cmd(struct fb_f1c100s_pdata_t * pdat, uint32_t val)
+{
+	sunxi_lcdc_output(pdat, 0, val);
+}
+
+static void r61520_lcd_dat(struct fb_f1c100s_pdata_t * pdat, uint32_t val)
+{
+	sunxi_lcdc_output(pdat, 1, val);
+}
+
  static inline void f1c100s_debe_set_mode(struct fb_f1c100s_pdata_t * pdat)
  {
  	struct f1c100s_debe_reg_t * debe = (struct f1c100s_debe_reg_t *)(pdat->virtdebe);
@@ -147,7 +190,7 @@
  	write32((virtual_addr_t)&tcon->ctrl, val);

  	val = (pdat->timing.v_front_porch + pdat->timing.v_back_porch + pdat->timing.v_sync_len);
-	write32((virtual_addr_t)&tcon->tcon0_ctrl, (1 << 31) | ((val & 0x1f) << 4));
+	write32((virtual_addr_t)&tcon->tcon0_ctrl, (1 << 31) | ((val & 0x1f) << 4) | (1 << 24)); // 8080 interface
  	val = clk_get_rate(pdat->clktcon) / pdat->timing.pixel_clock_hz;
  	write32((virtual_addr_t)&tcon->tcon0_dclk, (0xf << 28) | (val << 0));
  	write32((virtual_addr_t)&tcon->tcon0_timing_active, ((pdat->width - 1) << 16) | ((pdat->height - 1) << 0));
@@ -161,7 +204,7 @@
  	write32((virtual_addr_t)&tcon->tcon0_timing_sync, ((pdat->timing.h_sync_len - 1) << 16) | ((pdat->timing.v_sync_len - 1) << 0));

  	write32((virtual_addr_t)&tcon->tcon0_hv_intf, 0);
-	write32((virtual_addr_t)&tcon->tcon0_cpu_intf, 0);
+	write32((virtual_addr_t)&tcon->tcon0_cpu_intf, 0xe4000000); // 8080 mode

  	if(pdat->bits_per_pixel == 18 || pdat->bits_per_pixel == 16)
  	{
@@ -268,9 +311,236 @@
  			memcpy(pdat->vram[pdat->index], render->pixels, render->pixlen);
  			dma_cache_sync(pdat->vram[pdat->index], render->pixlen, DMA_TO_DEVICE);
  			f1c100s_debe_set_address(pdat, pdat->vram[pdat->index]);
+
+		{
+			uint32_t x, y;
+			uint32_t val, *p = render->pixels;
+
+			r61520_lcd_cmd(pdat, 0x2c);
+			for(y=0; y<240; y++){
+				for(x=0; x<320; x++){
+					val = *p++;
+					r61520_lcd_dat(pdat, ((val & 0xf80000) >> 8) | ((val & 0x00fc00) >> 5) | ((val & 0xf8) >> 3));
+				}
+			}
+		}
+		}
  	}

+static void sunxi_lcdc_gpio_config(struct fb_f1c100s_pdata_t * pdat, uint32_t use_gpio)
+{
+	uint32_t ret;
+
+	pdat->lcdc_use_gpio = use_gpio;
+	if(pdat->lcdc_use_gpio){
+		write32(pdat->virtgpio + PD_CFG0, 0x11111117); // 0x11111117
+		write32(pdat->virtgpio + PD_CFG1, 0x11111171); // 0x11111171
+		write32(pdat->virtgpio + PD_CFG2, 0x00111111); // 0x00111111, CS/RD/RS/WR
+		write32(pdat->virtgpio + PD_DATA, 0xffffffff);
+	}
+	else {
+		write32(pdat->virtgpio + PD_CFG0, 0x22222227); // 0x22222221
+		write32(pdat->virtgpio + PD_CFG1, 0x22222273); // 0x22222212
+		write32(pdat->virtgpio + PD_CFG2, 0x00222222); // 0x00222222, CS/RD/RS/WR
+	}
+	ret = read32(pdat->virtgpio + PE_CFG0);
+	ret&= 0xf0ffffff;
+	ret|= 0x01000000;
+	write32(pdat->virtgpio + PE_CFG0, ret);
+
+	ret = read32(pdat->virtgpio + PE_CFG1);
+	ret&= 0xffff0fff;
+	ret|= 0x00001000;
+	write32(pdat->virtgpio + PE_CFG1, ret);
+}
+
+static void r61520_lcd_init(struct fb_f1c100s_pdata_t * pdat)
+{
+	uint32_t ret;
+
+	ret = read32(pdat->virtgpio + PE_DATA);
+	ret|= 0x0040;
+	write32(pdat->virtgpio + PE_DATA, ret);
+
+	ret = read32(pdat->virtgpio + PE_DATA);
+	ret&= ~0x0800;
+	write32(pdat->virtgpio + PE_DATA, ret);
+	mdelay(150);
+	ret|= 0x0800;
+	write32(pdat->virtgpio + PE_DATA, ret);
+	mdelay(50);
+
+	r61520_lcd_cmd(pdat, 0xb0);
+	r61520_lcd_dat(pdat, 0x00);
+
+	r61520_lcd_cmd(pdat, 0xb1);
+	r61520_lcd_dat(pdat, 0x00);
+
+	r61520_lcd_cmd(pdat, 0xb3);
+	r61520_lcd_dat(pdat, 0x02);
+	r61520_lcd_dat(pdat, 0x00);
+	r61520_lcd_dat(pdat, 0x00);
+	r61520_lcd_dat(pdat, 0x00);
+
+	r61520_lcd_cmd(pdat, 0xb4);
+	r61520_lcd_dat(pdat, 0x00);
+
+	r61520_lcd_cmd(pdat, 0xc0);
+	r61520_lcd_dat(pdat, 0x07);
+	r61520_lcd_dat(pdat, 0x4f);
+	r61520_lcd_dat(pdat, 0x00);
+	r61520_lcd_dat(pdat, 0x00);
+	r61520_lcd_dat(pdat, 0x00);
```

```
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x01);
+    r61520_lcd_dat(pdat, 0x33);
+
+    r61520_lcd_cmd(pdat, 0xc1);
+    r61520_lcd_dat(pdat, 0x01);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x1a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x08);
+
+    r61520_lcd_cmd(pdat, 0xc3);
+    r61520_lcd_dat(pdat, 0x01);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x1a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x08);
+
+    r61520_lcd_cmd(pdat, 0xc4);
+    r61520_lcd_dat(pdat, 0x11);
+    r61520_lcd_dat(pdat, 0x01);
+    r61520_lcd_dat(pdat, 0x43);
+    r61520_lcd_dat(pdat, 0x01);
+
+    r61520_lcd_cmd(pdat, 0xc8);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x8a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x09);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x23);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x60);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x00);
+
+    r61520_lcd_cmd(pdat, 0xc9);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x8a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x09);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x23);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x09);
+    r61520_lcd_dat(pdat, 0x88);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x23);
+    r61520_lcd_dat(pdat, 0x00);
+
+    r61520_lcd_cmd(pdat, 0xca);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x8a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x09);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x23);
+    r61520_lcd_dat(pdat, 0x10);
+    r61520_lcd_dat(pdat, 0x05);
+    r61520_lcd_dat(pdat, 0x09);
+    r61520_lcd_dat(pdat, 0x88);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x08);
+    r61520_lcd_dat(pdat, 0x0a);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x23);
+    r61520_lcd_dat(pdat, 0x00);
+
+    r61520_lcd_cmd(pdat, 0xd0);
+    r61520_lcd_dat(pdat, 0x07);
+    r61520_lcd_dat(pdat, 0xc6);
+    r61520_lcd_dat(pdat, 0xdc);
+
+    r61520_lcd_cmd(pdat, 0xd1);
+    r61520_lcd_dat(pdat, 0x54);
+    r61520_lcd_dat(pdat, 0x0d);
+    r61520_lcd_dat(pdat, 0x02);
+
+    r61520_lcd_cmd(pdat, 0xd2);
+    r61520_lcd_dat(pdat, 0x63);
+    r61520_lcd_dat(pdat, 0x24);
+
+    r61520_lcd_cmd(pdat, 0xd4);
+    r61520_lcd_dat(pdat, 0x63);
+    r61520_lcd_dat(pdat, 0x24);
+
+    r61520_lcd_cmd(pdat, 0xd8);
+    r61520_lcd_dat(pdat, 0x07);
+    r61520_lcd_dat(pdat, 0x07);
+
+    r61520_lcd_cmd(pdat, 0xe0);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x00);
+
+    r61520_lcd_cmd(pdat, 0x13);
+
+    r61520_lcd_cmd(pdat, 0x20);
+
+    r61520_lcd_cmd(pdat, 0x35);
+    r61520_lcd_dat(pdat, 0x00);
+
+    r61520_lcd_cmd(pdat, 0x44);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x30);
+
+    r61520_lcd_cmd(pdat, 0x36);
+    r61520_lcd_dat(pdat, 0xe0);
+
+    r61520_lcd_cmd(pdat, 0x3a);
+    r61520_lcd_dat(pdat, 0x55);
+
+    r61520_lcd_cmd(pdat, 0x2a);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x01);
+    r61520_lcd_dat(pdat, 0x3f);
+
+    r61520_lcd_cmd(pdat, 0x2b);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0x00);
+    r61520_lcd_dat(pdat, 0xef);
+
+    r61520_lcd_cmd(pdat, 0x11);
+    r61520_lcd_cmd(pdat, 0x29);
+    r61520_lcd_cmd(pdat, 0x2c);
+}
+
  static struct device_t * fb_f1c100s_probe(struct driver_t * drv, struct dtnode_t * n)
  {
      struct fb_f1c100s_pdata_t * pdat;
@@ -298,23 +568,24 @@
      pdat->virtdefe = phys_to_virt(F1C100S_DEFE_BASE);
      pdat->virtdebe = phys_to_virt(F1C100S_DEBE_BASE);
      pdat->virttcon = phys_to_virt(F1C100S_TCON_BASE);
+    pdat->virtgpio = phys_to_virt(F1C100S_GPIO_BASE);
      pdat->clkdefe = strdup(clkdefe);
      pdat->clkdebe = strdup(clkdebe);
      pdat->clktcon = strdup(clktcon);
      pdat->rstdefe = dt_read_int(n, "reset-defe", -1);
      pdat->rstdebe = dt_read_int(n, "reset-debe", -1);
      pdat->rsttcon = dt_read_int(n, "reset-tcon", -1);
-    pdat->width = dt_read_int(n, "width", 800);
-    pdat->height = dt_read_int(n, "height", 400);
+    pdat->width = dt_read_int(n, "width", 320);
+    pdat->height = dt_read_int(n, "height", 240);
      pdat->pwidth = dt_read_int(n, "physical-width", 216);
      pdat->pheight = dt_read_int(n, "physical-height", 135);
-    pdat->bits_per_pixel = dt_read_int(n, "bits-per-pixel", 18);
-    pdat->bytes_per_pixel = dt_read_int(n, "bytes-per-pixel", 4);
+    pdat->bits_per_pixel = dt_read_int(n, "bits-per-pixel", 16);
+    pdat->bytes_per_pixel = dt_read_int(n, "bytes-per-pixel", 2);
      pdat->index = 0;
      pdat->vram[0] = dma_alloc_noncoherent(pdat->width * pdat->height * pdat->bytes_per_pixel);
      pdat->vram[1] = dma_alloc_noncoherent(pdat->width * pdat->height * pdat->bytes_per_pixel);

-    pdat->timing.pixel_clock_hz = dt_read_long(n, "clock-frequency", 33000000);
+    pdat->timing.pixel_clock_hz = dt_read_long(n, "clock-frequency", 80000000);
      pdat->timing.h_front_porch = dt_read_int(n, "hfront-porch", 40);
      pdat->timing.h_back_porch = dt_read_int(n, "hback-porch", 87);
```

```
        pdat->timing.h_sync_len = dt_read_int(n, "hsync-len", 1);
@@ -371,6 +642,8 @@
        }
        dev->driver = drv;

+   sunxi_lcdc_gpio_config(pdat, 1);
+   r61520_lcd_init(pdat);
        return dev;
  }

--- src/arch/arm32/mach-f1c100s/include/f1c100s/reg-gpio_orig.h
+++ src/arch/arm32/mach-f1c100s/include/f1c100s/reg-gpio.h
@@ -1,55 +0,0 @@
+/*
+ * Copyright (C) 2018 Steward Fu <steward.fu@gmail.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License as published by
+ * the Free Software Foundation; either version 2 of the License, or
+ * (at your option)any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
+ */
+#ifndef __REG_GPIO_H__
+#define __REG_GPIO_H__
+
+#define F1C100S_GPIO_BASE          0x01c20800
+
+#define PD_CFG0          ((3*0x24)+0x00)             //Port n Configure Register 0 (n=0~5)
+#define PD_CFG1          ((3*0x24)+0x04)             //Port n Configure Register 1 (n=0~5)
+#define PD_CFG2          ((3*0x24)+0x08)             //Port n Configure Register 2 (n=0~5)
+#define PD_CFG3          ((3*0x24)+0x0C)             //Port n Configure Register 3 (n=0~5)
+#define PD_DATA          ((3*0x24)+0x10)             //Port n Data Register (n=0~5)
+#define PD_DRV0          ((3*0x24)+0x14)             //Port n Multi-Driving Register 0 (n=0~5)
+#define PD_DRV1          ((3*0x24)+0x18)             //Port n Multi-Driving Register 1 (n=0~5)
+#define PD_PUL0          ((3*0x24)+0x1C)             //Port n Pull Register 0 (n=0~5)
+#define PD_PUL1          ((3*0x24)+0x20)             //Port n Pull Register 1 (n=0~5)
+
+#define PD_INT_CFG0 (0x200+(n*0x20)+0x0)   //PIO Interrupt Configure Register 0 (n=0~2)
+#define PD_INT_CFG1 (0x200+(n*0x20)+0x4)   //PIO Interrupt Configure Register 1 (n=0~2)
+#define PD_INT_CFG2 (0x200+(n*0x20)+0x8)   //PIO Interrupt Configure Register 2 (n=0~2)
+#define PD_INT_CFG3 (0x200+(n*0x20)+0xC)   //PIO Interrupt Configure Register 3 (n=0~2)
+#define PD_INT_CTRL (0x200+(n*0x20)+0x10) //PIO Interrupt Control Register (n=0~2)
+#define PD_INT_STA    (0x200+(n*0x20)+0x14) //PIO Interrupt Status Register (n=0~2)
+#define PD_INT_DEB    (0x200+(n*0x20)+0x18) //PIO Interrupt Debounce Register (n=0~2)
+
+#define PE_CFG0          ((3*0x24)+0x00)             //Port n Configure Register 0 (n=0~5)
+#define PE_CFG1          ((3*0x24)+0x04)             //Port n Configure Register 1 (n=0~5)
+#define PE_CFG2          ((3*0x24)+0x08)             //Port n Configure Register 2 (n=0~5)
+#define PE_CFG3          ((3*0x24)+0x0C)             //Port n Configure Register 3 (n=0~5)
+#define PE_DATA          ((3*0x24)+0x10)             //Port n Data Register (n=0~5)
+#define PE_DRV0          ((3*0x24)+0x14)             //Port n Multi-Driving Register 0 (n=0~5)
+#define PE_DRV1          ((3*0x24)+0x18)             //Port n Multi-Driving Register 1 (n=0~5)
+#define PE_PUL0          ((3*0x24)+0x1C)             //Port n Pull Register 0 (n=0~5)
+#define PE_PUL1          ((3*0x24)+0x20)             //Port n Pull Register 1 (n=0~5)
+
+#define SDR_PAD_DRV 0x2C0                            //SDRAM Pad Multi-Driving Register
+#define SDR_PAD_PUL 0x2C4                            //SDRAM Pad Pull Register
+
+#endif

--- src/arch/arm32/mach-f1c100s/include/f1c100s/reg-tcon_orig.h
+++ src/arch/arm32/mach-f1c100s/include/f1c100s/reg-tcon.h
@@ -0,0 +0,0 @@
+#define F1C100S_TCON_BASE               (0x01c0c000)
+#define TCON0_CPU_IF_REG                0x060 //TCON0 CPU Interface Control Register
+#define TCON0_CPU_WR_REG                0x064 //TCON0 CPU Mode Write Register
```

完成

返回上一頁 (../../handheld.htm)