

```
1  --- src/arch/arm32/mach-f1c100s/driver/fb-f1c100s_orig.c 2018-12-30 22:22:53.370788418 +0800
2  +++ src/arch/arm32/mach-f1c100s/driver/fb-f1c100s.c 2018-12-31 19:45:36.824995967 +0800
3  @@ -30,6 +30,7 @@
4  #include <led.h>
5  #include <framebuffer/framebuffer.h>
6  #include <f1c100s_gpio.h>
7  #include <f1c100s_reg_gpio.h>
8  #include <f1c100s_reg_tcon.h>
9  #include <f1c100s_reg_defbe.h>
10 #include <f1c100s_reg_debe.h>
11 @@ -39,6 +40,7 @@
12 virtual_addr_t virtdefbe;
13 virtual_addr_t virtdebe;
14 virtual_addr_t virttcon;
15 + virtual_addr_t virtgpio;
16
17 char * clkdefbe;
18 char * clkdebe;
19 @@ -53,6 +55,7 @@
20 int bits_per_pixel;
21 int bytes_per_pixel;
22 int index;
23 + int ledc_use_gpio;
24 void * vram[2];
25
26
27 @@ -73,6 +76,6 @@
28 int brightness;
29
30 };
31
32 +static void sunxi_ledc_output(struct fb_f1c100s_pdata_t * pdat, uint32_t is_data, uint32_t val)
33 +{
34 + uint32_t ret;
35 +
36 + if (pdac->ledc_use_gpio) {
37 + ret = (val & 0x00ff) << 1;
38 + ret = (val & 0xf000) << 2;
39 + ret = is_data ? 0x80000 : 0;
40 + write32(pdat->virtgpio + PD_DATA, ret);
41 + ret = 0x0000;
42 + write32(pdat->virtgpio + PD_DATA, ret);
43 + }
44 + else {
45 + ret = read32(pdat->virttcon + TCON0_CPU_IF_REG);
46 + if (is_data) {
47 + ret = 0x2000000;
48 + }
49 + else {
50 + ret&= 0xf000000;
51 + }
52 + write32(pdat->virttcon + TCON0_CPU_IF_REG, ret);
53 + while(read32(pdat->virttcon + TCON0_CPU_IF_REG) & 0xc0000) {
54 + ndelay(1);
55 + }
56 + write32(pdat->virttcon + TCON0_CPU_WR_REG, ((val & 0xf00) << 8) | ((val & 0x0300) << 6) | ((val & 0x00e0) << 5) | ((val & 0x001f) << 3));
57 + //write32(pdat->virttcon + TCON0_CPU_WR_REG, val);
58 + }
59 +}
60
61 +static void r61520_lcd_cmd(struct fb_f1c100s_pdata_t * pdat, uint32_t val)
62 +{
63 + sunxi_ledc_output(pdat, 0, val);
64 +}
65
66 +static void r61520_lcd_dat(struct fb_f1c100s_pdata_t * pdat, uint32_t val)
67 +{
68 + sunxi_ledc_output(pdat, 1, val);
69 +}
70
71 +static inline void f1c100s_debe_set_mode(struct fb_f1c100s_pdata_t * pdat)
72 +{
73 + struct f1c100s_debe_reg_t * debe = (struct f1c100s_debe_reg_t *) (pdac->virtdebe);
74 +
75 + @ -147,7 +190,7 @@
76 write32((virtual_addr_t)&tcon->ctrl, val);
77
78 val = (pdac->timing.v_front_porch + pdac->timing.v_back_porch + pdac->timing.v_sync_len);
79 - write32((virtual_addr_t)&tcon->tcon0_ctrl, (1 << 31) | ((val & 0x1f) << 4));
80 + write32((virtual_addr_t)&tcon->tcon0_ctrl, (1 << 31) | ((val & 0x1f) << 4) | (1 << 24)); // 8080 interface
81 val = clk_get_rate(pdat->clk_tcon) / pdac->timing.pixel_clock_hz;
82 write32((virtual_addr_t)&tcon->tcon0_deik, (0xf << 28) | (val << 0));
83 write32((virtual_addr_t)&tcon->tcon0_timing_active, ((pdac->width - 1) << 16) | ((pdac->height - 1) << 0));
84 @@ -161,7 +204,7 @@
85 write32((virtual_addr_t)&tcon->tcon0_timing_sync, ((pdac->timing.h_sync_len - 1) << 16) | ((pdac->timing.v_sync_len - 1) << 0));
86
87 write32((virtual_addr_t)&tcon->tcon0_hw_intf, 0);
88 - write32((virtual_addr_t)&tcon->tcon0_epu_intf, 0);
89 + write32((virtual_addr_t)&tcon->tcon0_epu_intf, 0x4000000); // 8080 mode
90
91 if (pdac->bits_per_pixel == 18 || pdac->bits_per_pixel == 16)
92 {
93 @@ -268,9 +311,236 @@
94 memcpy(pdat->vram[pdat->index], render->pixels, render->pixlen);
95 dma_cache_sync(pdat->vram[pdat->index], render->pixlen, DMA_TO_DEVICE);
96 f1c100s_debe_set_address(pdat, pdat->vram[pdat->index]);
97
98 {
99 uint32_t x, y;
100 uint32_t val, *p = render->pixels;
101
102 r61520_lcd_cmd(pdat, 0x2c);
103 for (y=0; y<240; y++) {
104 for (x=0; x<320; x++) {
105 val = *p++;
106 r61520_lcd_dat(pdat, ((val & 0xf80000) >> 8) | ((val & 0x00f000) >> 5) | ((val & 0xf8) >> 3));
107 }
108 }
109 }
110
111
112 +static void sunxi_ledc_gpio_config(struct fb_f1c100s_pdata_t * pdat, uint32_t use_gpio)
113 +{
114 + uint32_t ret;
115 +
116 + pdac->ledc_use_gpio = use_gpio;
117 + if (pdac->ledc_use_gpio) {
118 + write32(pdat->virtgpio + PD_CFG0, 0x11111111); // 0x11111111
119 + write32(pdat->virtgpio + PD_CFG1, 0x11111111); // 0x11111111
120 + write32(pdat->virtgpio + PD_CFG2, 0x00111111); // 0x00111111, CS/RD/RS/WR
121 + write32(pdat->virtgpio + PD_DATA, 0xffffffff);
122 + }
123 + else {
124 + write32(pdat->virtgpio + PD_CFG0, 0x22222222); // 0x22222222
125 + write32(pdat->virtgpio + PD_CFG1, 0x22222222); // 0x22222222
126 + write32(pdat->virtgpio + PD_CFG2, 0x00222222); // 0x00222222, CS/RD/RS/WR
127 + }
128 + ret = read32(pdat->virtgpio + PE_CFG0);
129 + ret&= 0x00ffffff;
130 + ret|= 0x01000000;
131 + write32(pdat->virtgpio + PE_CFG0, ret);
132 +
133 + ret = read32(pdat->virtgpio + PE_CFG1);
134 + ret&= 0xffff0000;
135 + ret|= 0x00001000;
136 + write32(pdat->virtgpio + PE_CFG1, ret);
137 +}
138
139 +static void r61520_lcd_init(struct fb_f1c100s_pdata_t * pdat)
140 +{
141 + uint32_t ret;
142 +
143 + ret = read32(pdat->virtgpio + PE_DATA);
144 + ret = 0x0040;
145 + write32(pdat->virtgpio + PE_DATA, ret);
146 +
147 + ret = read32(pdat->virtgpio + PE_DATA);
148 + ret&= 0x0800;
149 + write32(pdat->virtgpio + PE_DATA, ret);
150 + mdelay(150);
151 + ret = 0x0800;
152 + write32(pdat->virtgpio + PE_DATA, ret);
153 + mdelay(50);
154 +
155 + r61520_lcd_cmd(pdat, 0xb0);
156 + r61520_lcd_dat(pdat, 0x00);
157 +
158 + r61520_lcd_cmd(pdat, 0xb1);
159 + r61520_lcd_dat(pdat, 0x00);
160 +
161 + r61520_lcd_cmd(pdat, 0xb3);
162 + r61520_lcd_dat(pdat, 0x02);
163 + r61520_lcd_dat(pdat, 0x00);
164 + r61520_lcd_dat(pdat, 0x00);
165 + r61520_lcd_dat(pdat, 0x00);
166 +
167 + r61520_lcd_cmd(pdat, 0xb4);
168 + r61520_lcd_dat(pdat, 0x00);
169 +
170 + r61520_lcd_cmd(pdat, 0xc0);
171 + r61520_lcd_dat(pdat, 0x07);
172 + r61520_lcd_dat(pdat, 0x4f);
173 + r61520_lcd_dat(pdat, 0x00);
174 + r61520_lcd_dat(pdat, 0x00);
175 + r61520_lcd_dat(pdat, 0x00);
```

```

176 + r61520_led_dat(pdat, 0x00);
177 + r61520_led_dat(pdat, 0x01);
178 + r61520_led_dat(pdat, 0x33);
179 +
180 + r61520_led_cmd(pdat, 0xc1);
181 + r61520_led_dat(pdat, 0x01);
182 + r61520_led_dat(pdat, 0x00);
183 + r61520_led_dat(pdat, 0x1a);
184 + r61520_led_dat(pdat, 0x08);
185 + r61520_led_dat(pdat, 0x08);
186 +
187 + r61520_led_cmd(pdat, 0xc3);
188 + r61520_led_dat(pdat, 0x01);
189 + r61520_led_dat(pdat, 0x00);
190 + r61520_led_dat(pdat, 0x1a);
191 + r61520_led_dat(pdat, 0x08);
192 + r61520_led_dat(pdat, 0x08);
193 +
194 + r61520_led_cmd(pdat, 0xc4);
195 + r61520_led_dat(pdat, 0x11);
196 + r61520_led_dat(pdat, 0x01);
197 + r61520_led_dat(pdat, 0x33);
198 + r61520_led_dat(pdat, 0x01);
199 +
200 + r61520_led_cmd(pdat, 0xc5);
201 + r61520_led_dat(pdat, 0x00);
202 + r61520_led_dat(pdat, 0x0a);
203 + r61520_led_dat(pdat, 0x08);
204 + r61520_led_dat(pdat, 0x8a);
205 + r61520_led_dat(pdat, 0x08);
206 + r61520_led_dat(pdat, 0x09);
207 + r61520_led_dat(pdat, 0x05);
208 + r61520_led_dat(pdat, 0x10);
209 + r61520_led_dat(pdat, 0x00);
210 + r61520_led_dat(pdat, 0x23);
211 + r61520_led_dat(pdat, 0x10);
212 + r61520_led_dat(pdat, 0x05);
213 + r61520_led_dat(pdat, 0x05);
214 + r61520_led_dat(pdat, 0x00);
215 + r61520_led_dat(pdat, 0x0a);
216 + r61520_led_dat(pdat, 0x08);
217 + r61520_led_dat(pdat, 0x05);
218 + r61520_led_dat(pdat, 0x00);
219 + r61520_led_dat(pdat, 0x10);
220 + r61520_led_dat(pdat, 0x00);
221 +
222 + r61520_led_cmd(pdat, 0xc9);
223 + r61520_led_dat(pdat, 0x00);
224 + r61520_led_dat(pdat, 0x0a);
225 + r61520_led_dat(pdat, 0x08);
226 + r61520_led_dat(pdat, 0x8a);
227 + r61520_led_dat(pdat, 0x08);
228 + r61520_led_dat(pdat, 0x09);
229 + r61520_led_dat(pdat, 0x05);
230 + r61520_led_dat(pdat, 0x10);
231 + r61520_led_dat(pdat, 0x00);
232 + r61520_led_dat(pdat, 0x23);
233 + r61520_led_dat(pdat, 0x10);
234 + r61520_led_dat(pdat, 0x05);
235 + r61520_led_dat(pdat, 0x09);
236 + r61520_led_dat(pdat, 0x8a);
237 + r61520_led_dat(pdat, 0x0a);
238 + r61520_led_dat(pdat, 0x08);
239 + r61520_led_dat(pdat, 0x0a);
240 + r61520_led_dat(pdat, 0x00);
241 + r61520_led_dat(pdat, 0x23);
242 + r61520_led_dat(pdat, 0x00);
243 +
244 + r61520_led_cmd(pdat, 0xca);
245 + r61520_led_dat(pdat, 0x00);
246 + r61520_led_dat(pdat, 0x0a);
247 + r61520_led_dat(pdat, 0x08);
248 + r61520_led_dat(pdat, 0x8a);
249 + r61520_led_dat(pdat, 0x08);
250 + r61520_led_dat(pdat, 0x09);
251 + r61520_led_dat(pdat, 0x05);
252 + r61520_led_dat(pdat, 0x10);
253 + r61520_led_dat(pdat, 0x00);
254 + r61520_led_dat(pdat, 0x23);
255 + r61520_led_dat(pdat, 0x10);
256 + r61520_led_dat(pdat, 0x05);
257 + r61520_led_dat(pdat, 0x09);
258 + r61520_led_dat(pdat, 0x8a);
259 + r61520_led_dat(pdat, 0x0a);
260 + r61520_led_dat(pdat, 0x08);
261 + r61520_led_dat(pdat, 0x0a);
262 + r61520_led_dat(pdat, 0x00);
263 + r61520_led_dat(pdat, 0x23);
264 + r61520_led_dat(pdat, 0x00);
265 +
266 + r61520_led_cmd(pdat, 0xd0);
267 + r61520_led_dat(pdat, 0x07);
268 + r61520_led_dat(pdat, 0xc3);
269 + r61520_led_dat(pdat, 0xdc);
270 +
271 + r61520_led_cmd(pdat, 0xd1);
272 + r61520_led_dat(pdat, 0x54);
273 + r61520_led_dat(pdat, 0x04);
274 + r61520_led_dat(pdat, 0x02);
275 +
276 + r61520_led_cmd(pdat, 0xd2);
277 + r61520_led_dat(pdat, 0x63);
278 + r61520_led_dat(pdat, 0x24);
279 +
280 + r61520_led_cmd(pdat, 0xd4);
281 + r61520_led_dat(pdat, 0x63);
282 + r61520_led_dat(pdat, 0x24);
283 +
284 + r61520_led_cmd(pdat, 0xd8);
285 + r61520_led_dat(pdat, 0x07);
286 + r61520_led_dat(pdat, 0x07);
287 +
288 + r61520_led_cmd(pdat, 0xe0);
289 + r61520_led_dat(pdat, 0x00);
290 + r61520_led_dat(pdat, 0x00);
291 +
292 + r61520_led_cmd(pdat, 0x13);
293 +
294 + r61520_led_cmd(pdat, 0x20);
295 +
296 + r61520_led_cmd(pdat, 0x35);
297 + r61520_led_dat(pdat, 0x00);
298 +
299 + r61520_led_cmd(pdat, 0x44);
300 + r61520_led_dat(pdat, 0x00);
301 + r61520_led_dat(pdat, 0x30);
302 +
303 + r61520_led_cmd(pdat, 0x36);
304 + r61520_led_dat(pdat, 0xe0);
305 +
306 + r61520_led_cmd(pdat, 0x3a);
307 + r61520_led_dat(pdat, 0x55);
308 +
309 + r61520_led_cmd(pdat, 0x2a);
310 + r61520_led_dat(pdat, 0x00);
311 + r61520_led_dat(pdat, 0x00);
312 + r61520_led_dat(pdat, 0x01);
313 + r61520_led_dat(pdat, 0x3f);
314 +
315 + r61520_led_cmd(pdat, 0x2b);
316 + r61520_led_dat(pdat, 0x00);
317 + r61520_led_dat(pdat, 0x00);
318 + r61520_led_dat(pdat, 0x00);
319 + r61520_led_dat(pdat, 0xe7);
320 +
321 + r61520_led_cmd(pdat, 0x11);
322 + r61520_led_cmd(pdat, 0x29);
323 + r61520_led_cmd(pdat, 0x2c);
324 +
325 +
326 + static struct device_t * fb_f1c100s_probe(struct driver_t * drv, struct dtnode_t * n)
327 + {
328 +     struct fb_f1c100s_pdata_t * pdat;
329 +
330 +     pdat->virtdefe = phys_to_virt(F1C100S_DEFE_BASE);
331 +     pdat->virtdebe = phys_to_virt(F1C100S_DEBE_BASE);
332 +     pdat->virttcon = phys_to_virt(F1C100S_TCON_BASE);
333 +     pdat->virttcon = phys_to_virt(F1C100S_TCON_BASE);
334 +     pdat->clkdefe = strdup(ckldefe);
335 +     pdat->clkdebe = strdup(ckldebe);
336 +     pdat->clktcon = strdup(ckltcon);
337 +     pdat->rstrdefe = dt_read_int(n, "reset-defe", -1);
338 +     pdat->rstrdebe = dt_read_int(n, "reset-debe", -1);
339 +     pdat->rstrtcon = dt_read_int(n, "reset-tcon", -1);
340 +     pdat->width = dt_read_int(n, "width", 800);
341 +     pdat->height = dt_read_int(n, "height", 400);
342 +     pdat->width = dt_read_int(n, "width", 320);
343 +     pdat->height = dt_read_int(n, "height", 240);
344 +     pdat->pwidht = dt_read_int(n, "physical-width", 216);
345 +     pdat->bits_per_pixel = dt_read_int(n, "bits-per-pixel", 18);
346 +     pdat->bytes_per_pixel = dt_read_int(n, "bytes-per-pixel", 4);
347 +     pdat->bits_per_pixel = dt_read_int(n, "bits-per-pixel", 16);
348 +     pdat->bytes_per_pixel = dt_read_int(n, "bytes-per-pixel", 2);
349 +     pdat->index = 0;
350 +     pdat->vram[0] = dma_alloc_noncoherent(pdat->width * pdat->height * pdat->bytes_per_pixel);
351 +     pdat->vram[1] = dma_alloc_noncoherent(pdat->width * pdat->height * pdat->bytes_per_pixel);
352 +
353 +     pdat->timing.pixel_clock_hz = dt_read_long(n, "clock-frequency", 33000000);
354 +     pdat->timing.pixel_clock_hz = dt_read_long(n, "clock-frequency", 8000000);
355 +     pdat->timing.h_front_porch = dt_read_int(n, "hfront-porch", 40);
356 +     pdat->timing.h_back_porch = dt_read_int(n, "hback-porch", 87);

```

```

358 pdat->timing.h_sync_len = dt_read_int(u, "hsync-len", 1);
359 @@ -371,6 +642,8 @@
360 }
361 dev->driver = drv;
362
363 + sunxi_ledc_gpio_config(pdmat, 1);
364 + r61520_led_init(pdmat);
365 return dev;
366 }
367
368 --- src/arch/arm32/mach-f1c100s/include/f1c100s/reg-gpio.h
369 +++ src/arch/arm32/mach-f1c100s/include/f1c100s/reg-gpio.h
370 @@ -1,55 +0,0 @@
371 */
372 + * Copyright (C) 2018 Steward Fu (steward.fu@gmail.com)
373 + *
374 + * This program is free software; you can redistribute it and/or modify
375 + * it under the terms of the GNU General Public License as published by
376 + * the Free Software Foundation; either version 2 of the License, or
377 + * (at your option) any later version.
378 + *
379 + * This program is distributed in the hope that it will be useful,
380 + * but WITHOUT ANY WARRANTY; without even the implied warranty of
381 + * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
382 + * GNU General Public License for more details.
383 + *
384 + * You should have received a copy of the GNU General Public License
385 + * along with this program; if not, write to the Free Software
386 + * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
387 + */
388 #ifndef __REG_GPIO_H__
389 #define __REG_GPIO_H__
390 +
391 + #define F1C100S_GPIO_BASE 0x01c20800
392 +
393 + #define PD_CFG0 ((3*0x24)+0x00) //Port n Configure Register 0 (n=0~5)
394 + #define PD_CFG1 ((3*0x24)+0x04) //Port n Configure Register 1 (n=0~5)
395 + #define PD_CFG2 ((3*0x24)+0x08) //Port n Configure Register 2 (n=0~5)
396 + #define PD_CFG3 ((3*0x24)+0x0c) //Port n Configure Register 3 (n=0~5)
397 + #define PD_DATA ((3*0x24)+0x10) //Port n Data Register (n=0~5)
398 + #define PD_DRV0 ((3*0x24)+0x14) //Port n Multi-Driving Register 0 (n=0~5)
399 + #define PD_DRV1 ((3*0x24)+0x18) //Port n Multi-Driving Register 1 (n=0~5)
400 + #define PD_PUL0 ((3*0x24)+0x1c) //Port n Pull Register 0 (n=0~5)
401 + #define PD_PUL1 ((3*0x24)+0x20) //Port n Pull Register 1 (n=0~5)
402 +
403 + #define PD_INT_CFG0 (0x200+(n*0x20)+0x0) //PIO Interrupt Configure Register 0 (n=0~2)
404 + #define PD_INT_CFG1 (0x200+(n*0x20)+0x4) //PIO Interrupt Configure Register 1 (n=0~2)
405 + #define PD_INT_CFG2 (0x200+(n*0x20)+0x8) //PIO Interrupt Configure Register 2 (n=0~2)
406 + #define PD_INT_CFG3 (0x200+(n*0x20)+0xc) //PIO Interrupt Configure Register 3 (n=0~2)
407 + #define PD_INT_CTRL (0x200+(n*0x20)+0x10) //PIO Interrupt Control Register (n=0~2)
408 + #define PD_INT_STA (0x200+(n*0x20)+0x14) //PIO Interrupt Status Register (n=0~2)
409 + #define PD_INT_DEB (0x200+(n*0x20)+0x18) //PIO Interrupt Debounce Register (n=0~2)
410 +
411 + #define PE_CFG0 ((3*0x24)+0x00) //Port n Configure Register 0 (n=0~5)
412 + #define PE_CFG1 ((3*0x24)+0x04) //Port n Configure Register 1 (n=0~5)
413 + #define PE_CFG2 ((3*0x24)+0x08) //Port n Configure Register 2 (n=0~5)
414 + #define PE_CFG3 ((3*0x24)+0x0c) //Port n Configure Register 3 (n=0~5)
415 + #define PE_DATA ((3*0x24)+0x10) //Port n Data Register (n=0~5)
416 + #define PE_DRV0 ((3*0x24)+0x14) //Port n Multi-Driving Register 0 (n=0~5)
417 + #define PE_DRV1 ((3*0x24)+0x18) //Port n Multi-Driving Register 1 (n=0~5)
418 + #define PE_PUL0 ((3*0x24)+0x1c) //Port n Pull Register 0 (n=0~5)
419 + #define PE_PUL1 ((3*0x24)+0x20) //Port n Pull Register 1 (n=0~5)
420 +
421 + #define SDR_PAD_DRV 0x2c0 //SDRAM Pad Multi-Driving Register
422 + #define SDR_PAD_PUL 0x2c4 //SDRAM Pad Pull Register
423 +
424 #endif
425
426 --- src/arch/arm32/mach-f1c100s/include/f1c100s/reg-tcon.h
427 +++ src/arch/arm32/mach-f1c100s/include/f1c100s/reg-tcon.h
428 + #define F1C100S_TCON_BASE (0x01e0000)
429 + #define TCON0_CPU_IF_REG 0x060 //TCON0 CPU Interface Control Register
430 + #define TCON0_CPU_WF_REG 0x064 //TCON0 CPU Mode Write Register

```

完成



[返回上一頁 \(../handheld.htm\)](#)