**uboot 显示logo之AM335X**

**一、uboot版本：uboot-2013.10**

**二、需要有LCD的驱动支持，TI已经提供：drivers/video/da8xx-fb.c**

**三、在config头文件（include/configs/am335x\_evm.h）中定义相关的宏：**

/\* djf 20151125 add for uboot logo \*/

#define CONFIG\_VIDEO

#define CONFIG\_VIDEO\_DA8XX // am335x的LCD驱动

#define CONFIG\_CFB\_CONSOLE

#define CONFIG\_CMD\_BMP

#define DA8XX\_LCD\_CNTL\_BASE 0x4830E000 // am335x的LCD寄存器基地址

#define CONFIG\_SYS\_CONSOLE\_IS\_IN\_ENV

#define VIDEO\_TSTC\_FCT serial\_tstc

#define VIDEO\_GETC\_FCT serial\_getc

#define VIDEO\_KBD\_INIT\_FCT 0

#define CONFIG\_VIDEO\_LOGO

#define CONFIG\_SPLASH\_SCREEN //定义后，才会调用函数显示nand或sd卡中的图片

/\*

#define CONFIG\_SPLASH\_SCREEN\_ALIGN

#define VIDEO\_FB\_LITTLE\_ENDIAN

\*/

/\* djf 20151125 add for uboot logo \*/

**四、代码调用流程：**

**1.LCD的参数设置：**

<1> arch/arm/lib/board.c->board\_init\_r()函数调用 stdio\_init()函数（/common/stdio.c）:

{

……

board\_init(); /\* Setup chipselects \*/ //djf 20151126 mark for uboot logo init LCD

……

}

<2>函数会调用具体板级文件，am335x为/board/ti/am335x/board. c:

{

gd->bd->bi\_boot\_params = CONFIG\_SYS\_SDRAM\_BASE + 0x100;

#if defined(CONFIG\_NOR) || defined(CONFIG\_NAND)

gpmc\_init();

#endif

/\* djf 20151125 add for uboot logo \*/

struct cm\_perpll \*const cmper = (struct cm\_perpll \*)CM\_PER;

struct cm\_dpll \*cmdpll = (struct cm\_dpll \*)CM\_DPLL;

struct dpll\_params dpll\_lcd = {24, 1, -1, -1, -1, -1, -1};

uint32\_t \*const clk\_domains[] = {

&cmper->lcdclkctrl,

0,

};

uint32\_t \*const clk\_modules\_explicit\_en[] = {

&cmper->lcdclkctrl,

&cmper->lcdcclkstctrl,

0,

};

do\_enable\_clocks(clk\_domains, clk\_modules\_explicit\_en, 1);

writel(0x0, &cmdpll->clklcdcpixelclk);

do\_setup\_dpll(&dpll\_lcd\_regs, &dpll\_lcd);

da8xx\_video\_init(&da8xx\_panel\_at043, &lcd\_cfg, lcd\_cfg.bpp);

/\* djf 20151125 add for uboot logo \*/

}

此函数中需添加参数设置初始化函数（da8xx\_video\_init(&da8xx\_panel\_at043, &lcd\_cfg, lcd\_cfg.bpp);），会调用具体驱动中的函数，此处为/drivers/video/da8xx-fb.c，函数原型如下：

void da8xx\_video\_init(const struct da8xx\_panel \*panel,

const struct lcd\_ctrl\_config \*lcd\_cfg, int bits\_pixel)

{

lcd\_panel = panel;

da8xx\_lcd\_cfg = lcd\_cfg;

bits\_x\_pixel = bits\_pixel;

}

由此函数看出需要在板级文件（/board/ti/am335x/board. c）定义好所需的参数结构体，与linux中的结构体基本相同，需根据具体LCD的参数设置，如下：

/\* djf 20151125 add for uboot logo \*/

struct dpll\_regs dpll\_lcd\_regs = {

.cm\_clkmode\_dpll = CM\_WKUP + 0x98,

.cm\_idlest\_dpll = CM\_WKUP + 0x48,

.cm\_clksel\_dpll = CM\_WKUP + 0x54,

};

#define PLL\_GET\_M(v) ((v >> 8) & 0x7FF)

#define PLL\_GET\_N(v) (v & 0X7F)

int clk\_get(int clk)

{

uint32\_t val;

uint32\_t m, n;

val = readl(dpll\_lcd\_regs.cm\_clksel\_dpll);

m = PLL\_GET\_M(val);

n = PLL\_GET\_N(val);

printf("clk\_get val = 0x%x, m = 0x%x, n = 0x%x\n", val, m, n);

val = m \* V\_OSCK / n;

printf("clk = 0x%x V\_OSCK = %d\n", val, V\_OSCK);

return (m \* V\_OSCK) / n;

}

static const struct display\_panel disp\_panel = {

QVGA,

32,

32,

COLOR\_ACTIVE,

};

static struct da8xx\_panel da8xx\_panel\_at043= {

.name = "AT070TN94V\_1",

.width = 800,

.height = 480,

.hfp = 40,

.hbp = 40,

.hsw = 48,

.vfp = 13,

.vbp = 29,

.vsw = 3,

.pxl\_clk = 30000000,

.invert\_pxl\_clk = 0,

};

static struct lcd\_ctrl\_config lcd\_cfg = {

&disp\_panel,

.ac\_bias = 255,

.ac\_bias\_intrpt = 0,

.dma\_burst\_sz = 16,

.bpp = 32,

.fdd = 255,

.tft\_alt\_mode = 0,

.stn\_565\_mode = 0,

.mono\_8bit\_mode = 0,

.invert\_line\_clock = 1,

.invert\_frm\_clock = 1,

.sync\_edge = 0,

.sync\_ctrl = 1,

.raster\_order = 0,

};

/\* djf 20151125 add for uboot logo \*/

其中clk\_get()函数会在/drivers/video/da8xx-fb.c中lcd\_calc\_clk\_divider()用到，所以也必须给出。

至此，宏定义及参数设置就算完成了。接下来就是具体驱动的调用及logo的显示了。

**2.LCD驱动初始化及logo显示**

<1>arch/arm/lib/board.c->board\_init\_r()函数调用 stdio\_init()函数（/common/stdio.c）:

{

……

#if defined(CONFIG\_VIDEO) || defined(CONFIG\_CFB\_CONSOLE)

drv\_video\_init (); //djf 20151126 mark for uboot logo

#endif

……

}

<2> drv\_video\_init ()函数在/drivers/video/cfb\_console.c中：

{

……

/\* Init video chip - returns with framebuffer cleared \*/

skip\_dev\_init = (video\_init() == -1); // djf 20151126 add for uboot logo

……

}

<3> 调用同文件下video\_init()函数：

{

pGD = video\_hw\_init(); //djf 20151126 mark for uboot logo

if (pGD == NULL)

return -1;

video\_fb\_address = (void \*) VIDEO\_FB\_ADRS; //djf 20151126 mark for uboot logo : framebuffer address(#define VIDEO\_FB\_ADRS (pGD->frameAdrs))

}

<4> video\_hw\_init()函数会调用/drivers/video/下具体驱动内的函数，am335x的为da8xx-fb.c。此函数会根据第1步在板级文件中设置好的参数及clk\_get()函数对LCD进行初始化。

<5>再看<3>中video\_init()函数下面的部分：

{

#ifdef CONFIG\_VIDEO\_LOGO

/\* Plot the logo and get start point of console \*/

debug("Video: Drawing the logo ...\n");

video\_console\_address = video\_logo();//djf 20151126 mark for uboot logo

#else

video\_console\_address = video\_fb\_address;

#endif

}

<6>会调用到同文件下的video\_logo()函数：

{

#ifdef CONFIG\_SPLASH\_SCREEN

s = getenv("splashimage"); //djf 20151126 mark for uboot logo

if (s != NULL) {

splash\_screen\_prepare();//什么都没做

addr = simple\_strtoul(s, NULL, 16);

if (video\_display\_bitmap(addr,

video\_logo\_xpos,

video\_logo\_ypos) == 0) {

video\_logo\_height = 0;

return ((void \*) (video\_fb\_address));

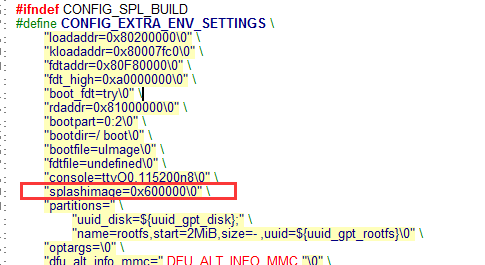
} //djf 20151126 mark for uboot logo

}

#endif /\* CONFIG\_SPLASH\_SCREEN \*/

}

此函数中会获取uboot环境变量splashimage的值（需要在板级config文件/include/configs/am335x\_evm.h中设置，见下图：），



获取后进行转换，这个值就是bmp图片的存储地址，但是原生的uboot是只支持Nor Flash和内核的读取的，如果图片放在nand flash、SD卡或者SPI、I2C 接口flash中，需要自己实现将图片文件读取到内存中，然后再显示。调用同文件下video\_display\_bitmap()函数实现具体的logo显示，函数原型如下：(由于想要显示nand或者sd卡中的bmp图片做为logo，所以添加了图片地址判断及图片预读取到内存0x82000000相关代码，其中涉及到的地址与am335x地址分配及板级文件对nand分区等有关)

/\*

\* Display the BMP file located at address bmp\_image.

\*/

int video\_display\_bitmap(ulong bmp\_image, int x, int y)

{

// printf( "video\_display\_bitmap() is executing \n" ); //djf 20151126 add for test

ushort xcount, ycount;

uchar \*fb;

// bmp\_image\_t \*bmp = (bmp\_image\_t \*) bmp\_image; //djf 20151127 del

bmp\_image\_t \*bmp; //djf 20151127 add

uchar \*bmap;

ushort padded\_line;

unsigned long width, height, bpp;

unsigned colors;

unsigned long compression;

bmp\_color\_table\_entry\_t cte;

#ifdef CONFIG\_VIDEO\_BMP\_GZIP

unsigned char \*dst = NULL;

ulong len;

#endif

WATCHDOG\_RESET();

/\* djf 20151127 add for uboot logo:copy data to ram from nand \*/

char \*argv[3];

char buf[100];

if(bmp\_image < 0x40000000)

{

sprintf(buf, "mw.b 0x82000000 0xff 0x200000"); // 清除2M内存

setenv("tempenv", buf);

argv[0] = "run";

argv[1] = "tempenv";

argv[2] = NULL;

if (do\_run(NULL, 0, 2, argv))

{

printf("Error: mw.b err \n");

return 1;

}

setenv("tempenv", NULL);

// 读取图片信息到内存

sprintf(buf, "nand read 0x82000000 0x600000 0x200000");

setenv("tempenv", buf);

argv[0] = "run";

argv[1] = "tempenv";

argv[2] = NULL;

if(do\_run(NULL, 0, 2, argv))

{

printf("Error: nand read err \n");

return 1;

}

setenv("tempenv", NULL);

bmp = (bmp\_image\_t \*)0x82000000;

}

// printf("copy logo data from nand done\n"); //djf 20151127 add for test

/\* djf 20151127 add for uboot logo:copy data to ram from nand \*/

if (!((bmp->header.signature[0] == 'B') &&

(bmp->header.signature[1] == 'M'))) {

#ifdef CONFIG\_VIDEO\_BMP\_GZIP

/\*

\* Could be a gzipped bmp image, try to decrompress...

\*/

len = CONFIG\_SYS\_VIDEO\_LOGO\_MAX\_SIZE;

dst = malloc(CONFIG\_SYS\_VIDEO\_LOGO\_MAX\_SIZE);

if (dst == NULL) {

printf("Error: malloc in gunzip failed!\n");

return 1;

}

if (gunzip(dst, CONFIG\_SYS\_VIDEO\_LOGO\_MAX\_SIZE,

(uchar \*) bmp\_image,

&len) != 0) {

printf("Error: no valid bmp or bmp.gz image at %lx\n",

bmp\_image);

free(dst);

return 1;

}

if (len == CONFIG\_SYS\_VIDEO\_LOGO\_MAX\_SIZE) {

printf("Image could be truncated "

"(increase CONFIG\_SYS\_VIDEO\_LOGO\_MAX\_SIZE)!\n");

}

/\*

\* Set addr to decompressed image

\*/

bmp = (bmp\_image\_t \*) dst;

if (!((bmp->header.signature[0] == 'B') &&

(bmp->header.signature[1] == 'M'))) {

printf("Error: no valid bmp.gz image at %lx\n",

bmp\_image);

free(dst);

return 1;

}

#else

/\* djf 20151127 add for uboot logo:copy data to ram from mmc \*/

sprintf(buf, "mmc rescan");

setenv("tempenv", buf);

argv[0] = "run";

argv[1] = "tempenv";

argv[2] = NULL;

if (do\_run(NULL, 0, 2, argv))

{

printf("There is no valid bmp image at %lx\n and no mmc", bmp\_image);

return 1;

}

setenv("tempenv", NULL);

sprintf(buf, "mw.b 0x82000000 0xff 0x200000");

setenv("tempenv", buf);

argv[0] = "run";

argv[1] = "tempenv";

argv[2] = NULL;

if (do\_run(NULL, 0, 2, argv))

{

printf("Error: mw.b err \n");

return 1;

}

setenv("tempenv", NULL);

sprintf(buf, "fatload mmc 0 0x82000000 logo.bmp");

setenv("tempenv", buf);

argv[0] = "run";

argv[1] = "tempenv";

argv[2] = NULL;

if(do\_run(NULL, 0, 2, argv))

{

printf("Error: fatload logo.bmp err \n");

return 1;

}

setenv("tempenv", NULL);

bmp = (bmp\_image\_t \*)0x82000000;

if (!((bmp->header.signature[0] == 'B') &&

(bmp->header.signature[1] == 'M')))

{

printf("There is no valid bmp image at %lx\n or mmc", bmp\_image);

return 1;

}

/\* djf 20151127 add for uboot logo:copy data to ram from mmc \*/

/\* //djf 20151127 del

printf("Error: no valid bmp image at %lx\n or mmc", bmp\_image);

return 1;

\*/

#endif /\* CONFIG\_VIDEO\_BMP\_GZIP \*/

}

// printf( "valid bmp image at %lx\n", bmp\_image); //djf 20151126 add for test

width = le32\_to\_cpu(bmp->header.width);

height = le32\_to\_cpu(bmp->header.height);

bpp = le16\_to\_cpu(bmp->header.bit\_count);

colors = le32\_to\_cpu(bmp->header.colors\_used);

compression = le32\_to\_cpu(bmp->header.compression);

debug("Display-bmp: %ld x %ld with %d colors\n",

width, height, colors);

if (compression != BMP\_BI\_RGB

#ifdef CONFIG\_VIDEO\_BMP\_RLE8

&& compression != BMP\_BI\_RLE8

#endif

) {

printf("Error: compression type %ld not supported\n",

compression);

#ifdef CONFIG\_VIDEO\_BMP\_GZIP

if (dst)

free(dst);

#endif

return 1;

}

padded\_line = (((width \* bpp + 7) / 8) + 3) & ~0x3;

//此宏定义必须打开

#ifdef CONFIG\_SPLASH\_SCREEN\_ALIGN

if (x == BMP\_ALIGN\_CENTER)

x = max(0, (VIDEO\_VISIBLE\_COLS - width) / 2);

else if (x < 0)

x = max(0, VIDEO\_VISIBLE\_COLS - width + x + 1);

if (y == BMP\_ALIGN\_CENTER)

y = max(0, (VIDEO\_VISIBLE\_ROWS - height) / 2);

else if (y < 0)

y = max(0, VIDEO\_VISIBLE\_ROWS - height + y + 1);

#endif /\* CONFIG\_SPLASH\_SCREEN\_ALIGN \*/

/\*

\* Just ignore elements which are completely beyond screen

\* dimensions.

\*/

if ((x >= VIDEO\_VISIBLE\_COLS) || (y >= VIDEO\_VISIBLE\_ROWS))

return 0;

if ((x + width) > VIDEO\_VISIBLE\_COLS)

width = VIDEO\_VISIBLE\_COLS - x;

if ((y + height) > VIDEO\_VISIBLE\_ROWS)

height = VIDEO\_VISIBLE\_ROWS - y;

bmap = (uchar \*) bmp + le32\_to\_cpu(bmp->header.data\_offset);

fb = (uchar \*) (video\_fb\_address +

((y + height - 1) \* VIDEO\_COLS \* VIDEO\_PIXEL\_SIZE) +

x \* VIDEO\_PIXEL\_SIZE);

#ifdef CONFIG\_VIDEO\_BMP\_RLE8

if (compression == BMP\_BI\_RLE8) {

return display\_rle8\_bitmap(bmp, x, y, width, height);

}

#endif

/\* We handle only 4, 8, or 24 bpp bitmaps \*/

switch (le16\_to\_cpu(bmp->header.bit\_count)) {

case 4:

padded\_line -= width / 2;

ycount = height;

switch (VIDEO\_DATA\_FORMAT) {

case GDF\_32BIT\_X888RGB:

while (ycount--) {

WATCHDOG\_RESET();

/\*

\* Don't assume that 'width' is an

\* even number

\*/

for (xcount = 0; xcount < width; xcount++) {

uchar idx;

if (xcount & 1) {

idx = \*bmap & 0xF;

bmap++;

} else

idx = \*bmap >> 4;

cte = bmp->color\_table[idx];

FILL\_32BIT\_X888RGB(cte.red, cte.green,

cte.blue);

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

default:

puts("4bpp bitmap unsupported with current "

"video mode\n");

break;

}

break;

case 8:

// printf( "the bmp is 8bpp\n" ); //djf 20151126 add for test

padded\_line -= width;

if (VIDEO\_DATA\_FORMAT == GDF\_\_8BIT\_INDEX) {

/\* Copy colormap \*/

for (xcount = 0; xcount < colors; ++xcount) {

cte = bmp->color\_table[xcount];

video\_set\_lut(xcount, cte.red, cte.green,

cte.blue);

}

}

ycount = height;

switch (VIDEO\_DATA\_FORMAT) {

case GDF\_\_8BIT\_INDEX:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

\*fb++ = \*bmap++;

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_\_8BIT\_332RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

cte = bmp->color\_table[\*bmap++];

FILL\_8BIT\_332RGB(cte.red, cte.green,

cte.blue);

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_15BIT\_555RGB:

while (ycount--) {

#if defined(VIDEO\_FB\_16BPP\_PIXEL\_SWAP)

int xpos = x;

#endif

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

cte = bmp->color\_table[\*bmap++];

#if defined(VIDEO\_FB\_16BPP\_PIXEL\_SWAP)

fill\_555rgb\_pswap(fb, xpos++, cte.red,

cte.green,

cte.blue);

fb += 2;

#else

FILL\_15BIT\_555RGB(cte.red, cte.green,

cte.blue);

#endif

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_16BIT\_565RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

cte = bmp->color\_table[\*bmap++];

FILL\_16BIT\_565RGB(cte.red, cte.green,

cte.blue);

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_32BIT\_X888RGB:

// printf( "The LCD bit type is 32bit\n" ); //djf 20151126 add for test

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

cte = bmp->color\_table[\*bmap++];

FILL\_32BIT\_X888RGB(cte.red, cte.green,

cte.blue);

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_24BIT\_888RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

cte = bmp->color\_table[\*bmap++];

FILL\_24BIT\_888RGB(cte.red, cte.green,

cte.blue);

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

}

break;

case 24:

padded\_line -= 3 \* width;

ycount = height;

switch (VIDEO\_DATA\_FORMAT) {

case GDF\_\_8BIT\_332RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

FILL\_8BIT\_332RGB(bmap[2], bmap[1],

bmap[0]);

bmap += 3;

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_15BIT\_555RGB:

while (ycount--) {

#if defined(VIDEO\_FB\_16BPP\_PIXEL\_SWAP)

int xpos = x;

#endif

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

#if defined(VIDEO\_FB\_16BPP\_PIXEL\_SWAP)

fill\_555rgb\_pswap(fb, xpos++, bmap[2],

bmap[1], bmap[0]);

fb += 2;

#else

FILL\_15BIT\_555RGB(bmap[2], bmap[1],

bmap[0]);

#endif

bmap += 3;

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_16BIT\_565RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

FILL\_16BIT\_565RGB(bmap[2], bmap[1],

bmap[0]);

bmap += 3;

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_32BIT\_X888RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

FILL\_32BIT\_X888RGB(bmap[2], bmap[1],

bmap[0]);

bmap += 3;

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

case GDF\_24BIT\_888RGB:

while (ycount--) {

WATCHDOG\_RESET();

xcount = width;

while (xcount--) {

FILL\_24BIT\_888RGB(bmap[2], bmap[1],

bmap[0]);

bmap += 3;

}

bmap += padded\_line;

fb -= (VIDEO\_VISIBLE\_COLS + width) \*

VIDEO\_PIXEL\_SIZE;

}

break;

default:

printf("Error: 24 bits/pixel bitmap incompatible "

"with current video mode\n");

break;

}

break;

default:

printf("Error: %d bit/pixel bitmaps not supported by U-Boot\n",

le16\_to\_cpu(bmp->header.bit\_count));

break;

}

#ifdef CONFIG\_VIDEO\_BMP\_GZIP

if (dst) {

free(dst);

}

#endif

if (cfb\_do\_flush\_cache)

flush\_cache(VIDEO\_FB\_ADRS, VIDEO\_SIZE);

return (0);

}

**五、问题**

**1.uboot 中logo 和kernel 中 logo因色深度影响，不完全一致**

**2.uboot显示logo会有轻微波纹**

**3.修改nand的分区等时，需要修改最后一步中自己添加的拷贝bmp图片信息到内存相关的代码，涉及到的地址与分区及为图片分配空间大小有关，现在的分区是MLO，MLO三个备用分区，每个分区2M，其中第三个备用分区（0x600000-0x7fffff）用来保存logo图片，uboot分区。接下来是2M的 Uboot，2M的Env，8M的kernel，剩下的rootfs。**

**4.记得添加LCD引脚配置：/board/ti/am335x/mux.c->enable\_board\_pin\_mux():**

**void enable\_board\_pin\_mux(struct am335x\_baseboard\_id \*header)**

**{**

**/\* Do board-specific muxes. \*/**

**if (board\_is\_bone(header)) {**

**/\* Beaglebone pinmux \*/**

**configure\_module\_pin\_mux(i2c1\_pin\_mux);**

**configure\_module\_pin\_mux(mii1\_pin\_mux);**

**configure\_module\_pin\_mux(mmc0\_pin\_mux);**

**#ifndef CONFIG\_NOR**

**configure\_module\_pin\_mux(mmc1\_pin\_mux);**

**#endif**

**#if defined(CONFIG\_NOR) && !defined(CONFIG\_NOR\_BOOT)**

**configure\_module\_pin\_mux(bone\_norcape\_pin\_mux);**

**#endif**

**} else if (board\_is\_gp\_evm(header)) {**

**/\* General Purpose EVM \*/**

**//unsigned short profile = detect\_daughter\_board\_profile();**

**configure\_module\_pin\_mux(rgmii1\_pin\_mux);**

**configure\_module\_pin\_mux(mmc0\_pin\_mux);**

**/\* In profile #2 i2c1 and spi0 conflict. \*/**

**/\*dragoniye \*/**

**configure\_module\_pin\_mux(nand\_pin\_mux);**

**//configure\_module\_pin\_mux(spi0\_pin\_mux);//djf 20150105 del pin not use**

**//configure\_module\_pin\_mux(i2c1\_pin\_mux);//djf 20150105 del pin not use**

**configure\_module\_pin\_mux( lcd\_pin\_mux);**

**//configure\_module\_pin\_mux( gpio\_pin\_mux);**

**……**

**}**

**}**

**其中lcd\_pin\_mux[]如下：**

**/\* djf 20150105 add start \*/**

**static struct module\_pin\_mux lcd\_pin\_mux[] = {**

**{OFFSET(gpmc\_ad8), (MODE(1))},/\* LCD\_DATA23 \*/**

**{OFFSET(gpmc\_ad9), (MODE(1))},**

**{OFFSET(gpmc\_ad10), (MODE(1))},**

**{OFFSET(gpmc\_ad11), (MODE(1))},**

**{OFFSET(gpmc\_ad12), (MODE(1))},**

**{OFFSET(gpmc\_ad13), (MODE(1))},**

**{OFFSET(gpmc\_ad14), (MODE(1))},**

**{OFFSET(gpmc\_ad15), (MODE(1))},**

**{OFFSET(lcd\_data0), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data1), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data2), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data3), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data4), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data5), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data6), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data7), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data8), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data9), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data10), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data11), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data12), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data13), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data14), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_data15), (MODE(0) | PULLUDDIS)},**

**{OFFSET(lcd\_vsync), (MODE(0))},**

**{OFFSET(lcd\_hsync), (MODE(0))},**

**{OFFSET(lcd\_pclk), (MODE(0))},**

**{OFFSET(lcd\_ac\_bias\_en), (MODE(0))},**

**{-1},**

**};**

**/\* djf 20150105 add end \*/**

**5.LCD背光打开是在/arch/arm/lib/board.c->board\_init\_r()->**

**board\_late\_init()(/board/ti/am335x/board.c)->** **tps65217\_open\_backlight()中完成的，但是默认的是没有打开，需修改如下：**

**void tps65217\_open\_backlight(void)**

**{**

**printf("TPS65217 open backlight entry \n");**

**if (i2c\_probe(TPS65217\_CHIP\_PM))**

**return;**

**printf("TPS65217 probe OK \n");**

**if (tps65217\_reg\_write(TPS65217\_PROT\_LEVEL\_NONE,**

**TPS65217\_WLEDCTRL2,**

**40,**

**TPS65217\_MASK\_ALL\_BITS))**

**printf("TPS65217 set wled duty error \n");**

**if (tps65217\_reg\_write(TPS65217\_PROT\_LEVEL\_NONE,**

**TPS65217\_WLEDCTRL1,**

**8,**

**TPS65217\_WLEDCTRL1\_ON\_OFF\_BITMASK))// djf 2015 1126 change value 1 to 8**

**printf("TPS65217 set wled On error \n");**

**printf("TPS65217 open backlight OK \n");**

**}**