1.开启 spi 设备

(1)在 Linux 源码目录下的 arch/arm/boot/dts/sama5d3mb.dtsi 文件添加以下内容:

```
spi0: spi@f0004000 {
                    m25p80@0 {
                        compatible = "atmel,at25df321a";
                        spi-max-frequency = <50000000>;
                        reg = <0>;
                    };
//************************添加 spi0 设备*******************************
                    spidev@1 {
                        compatible = "spidev";
                        spi-max-frequency = <50000000>;
                        reg = <1>;
                    };
//*************************************************************
                };
//************************添加 spi1 设备*******************************
                spi1: spi@f8008000 {
                    spidev@0 {
                        compatible = "spidev";
                        spi-max-frequency = <50000000>;
                        reg = <0>;
                    };
                };
//*************************************************************
```

(2)编译选项

```
$ make ARCH=arm menuconfig
Device Drivers  --->
            SPI support  --->
                <*>   User mode SPI device driver support
```

(3)编译

编译内核和设备树

(4)测试设备

测试程序:

```
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
```

```
#define ARRAY_SIZE(a) (sizeof(a) / sizeof((a)[0]))

static void pabort(const char *s)
{
		perror(s);
		abort();
}

static const char *device = "/dev/spidev1.1";
static uint8_t mode;
static uint8_t bits = 8;
static uint32_t speed = 500000;
static uint16_t delay;

static void transfer(int fd)
{
		int ret;
		uint8_t tx[] = {
			0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
			0x40, 0x00, 0x00, 0x00, 0x00, 0x95,
			0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
			0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
			0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
			0xDE, 0xAD, 0xBE, 0xEF, 0xBA, 0xAD,
			0xF0, 0x0D,
		};
		uint8_t rx[ARRAY_SIZE(tx)] = {0, };
		struct spi_ioc_transfer tr = {
			.tx_buf = (unsigned long)tx,
			.rx_buf = (unsigned long)rx,
			.len = ARRAY_SIZE(tx),
			.delay_usecs = delay,
			//.speed_hz = speed,
			//.bits_per_word = bits,
		};

		ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
		if (ret < 1)
			pabort("can't send spi message");

		for (ret = 0; ret < ARRAY_SIZE(tx); ret++) {
			if (!(ret % 6))
			    puts("");
			printf("%.2X ", rx[ret]);
		}
		puts("");
}

static void print_usage(const char *prog)
{
		printf("Usage: %s [-DsbdlHOLC3]\n", prog);
		puts("  -D --device   device to use (default /dev/spidev1.1)\n"
	     "  -s --speed    max speed (Hz)\n"
	     "  -d --delay    delay (usec)\n"
	     "  -b --bpw      bits per word \n"
	     "  -l --loop     loopback\n"
	     "  -H --cpha     clock phase\n"
	     "  -O --cpol     clock polarity\n"
	     "  -L --lsb      least significant bit first\n"
	     "  -C --cs-high  chip select active high\n"
	     "  -3 --3wire    SI/SO signals shared\n");
	exit(1);
}

static void parse_opts(int argc, char *argv[])
{
	while (1) {
		static const struct option lopts[] = {
			{ "device",  1, 0, 'D' },
			{ "speed",   1, 0, 's' },
			{ "delay",   1, 0, 'd' },
			{ "bpw",     1, 0, 'b' },
			{ "loop",    0, 0, 'l' },
			{ "cpha",    0, 0, 'H' },
			{ "cpol",    0, 0, 'O' },
			{ "lsb",     0, 0, 'L' },
			{ "cs-high", 0, 0, 'C' },
			{ "3wire",   0, 0, '3' },
			{ "no-cs",   0, 0, 'N' },
			{ "ready",   0, 0, 'R' },
			{ NULL, 0, 0, 0 },
		};
		int c;

		c = getopt_long(argc, argv, "D:s:d:b:lHOLC3NR",
lopts, NULL);

		if (c == -1)
			break;

		switch (c) {
		case 'D':
			device = optarg;
			break;
		case 's':
			speed = atoi(optarg);
			break;
		case 'd':
			delay = atoi(optarg);
			break;
		case 'b':
			bits = atoi(optarg);
			break;
		case 'l':
				mode |= SPI_LOOP;
				break;
			case 'H':
				mode |= SPI_CPHA;
				break;
			case 'O':
				mode |= SPI_CPOL;
				break;
			case 'L':
				mode |= SPI_LSB_FIRST;
				break;
			case 'C':
				mode |= SPI_CS_HIGH;
				break;
			case '3':
				mode |= SPI_3WIRE;
				break;
			case 'N':
				mode |= SPI_NO_CS;
				break;
			case 'R':
				mode |= SPI_READY;
				break;
			default:
				print_usage(argv[0]);
				break;
			}
		}
}

int main(int argc, char *argv[])
{
		int ret = 0;
		int fd;

		parse_opts(argc, argv);

		fd = open(device, O_RDWR);
		if (fd < 0)
			pabort("can't open device");

		/*
		 * spi mode
		 */
		ret = ioctl(fd, SPI_IOC_WR_MODE, &mode);
		if (ret == -1)
			pabort("can't set spi mode");

		ret = ioctl(fd, SPI_IOC_RD_MODE, &mode);
		if (ret == -1)
			pabort("can't get spi mode");
```

```
        /*
         * bits per word
         */
        ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
        if (ret == -1)
                pabort("can't set bits per word");

        ret = ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits);
        if (ret == -1)
                pabort("can't get bits per word");

        /*
         * max speed hz
         */
        ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
        if (ret == -1)
                pabort("can't set max speed hz");

        ret = ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);
        if (ret == -1)
                pabort("can't get max speed hz");

        printf("spi mode: %d\n", mode);
        printf("bits per word: %d\n", bits);
        printf("max speed: %d Hz (%d KHz)\n", speed, speed/1000);

        transfer(fd);

        close(fd);

        return ret;
}
```

将 SPI0_MOSI 与 SPI0_MISO 短接，运行命令，输出应该输出如下内容：

```
# ./spidev_test -D /dev/spidev32766.1 -s 50000000
spi mode: 0
bits per word: 8
max speed: 5000000 Hz (5000 KHz)

FF FF FF FF FF FF
40 00 00 00 00 95
FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF FF FF FF FF
DE AD BE EF BA AD
F0 0D
```

2.修改开机画面

（1）获取一张 BMP 的图片，修改之，让其色深为 8 位，即 256 色，如果用 24 位，则显示出问题。

（2）将制作好的 BMP 图片，放置到<u-boot>/tools/logos 下面。

（3）修改<u-boot>/tools/Makefile 中的 LOGO_BMP，使其指向你的 bmp 图片。如：LOGO_BMP= logos/myir.bmp ，替换为我自己的 meng.bmp，或者将此放 LOGO_BMP 赋值的最后，否则会被覆盖掉。如下第六行：

```
ifeq ($(LOGO_BMP),)
LOGO_BMP= logos/denx.bmp
endif
ifeq ($(VENDOR),atmel)
#LOGO_BMP= logos/atmel.bmp
#LOGO_BMP= logos/log.bmp
endif
ifeq ($(VENDOR),esd)
#LOGO_BMP= logos/esd.bmp
endif
ifeq ($(VENDOR),freescale)
#LOGO_BMP= logos/freescale.bmp
endif
ifeq ($(VENDOR),ronetix)
#LOGO_BMP= logos/ronetix.bmp
endif
ifeq ($(VENDOR),syteco)
#LOGO_BMP= logos/syteco.bmp
endif
ifeq ($(VENDOR),intercontrol)
#LOGO_BMP= logos/intercontrol.bmp
endif
```

注：此处的 VENDOR 信息是从<boards.cfg>文件中来。

（4）修改开机系统信息文字

如要不显示：

MYIR TECH Corp
support@myirtech.com

文字，修改 u-boot 相关源码，如 A5D3 的：board/atmel/sama5d3xek/sama5d3xek.c 第 161 和 162 行

```
void lcd_show_board_info(void)
{
		ulong dram_size, nand_size;
		int i;
		char temp[32];

		//lcd_printf ("%s\n", U_BOOT_VERSION);
		lcd_printf ("Welcome!\nTrace Gas Analyzer Team\n");//想要显示的文字信息
		//lcd_printf ("support@myirtech.com\n");
		//lcd_printf ("%s CPU at %s MHz\n",
		//	get_cpu_name(),
		//	strmhz(temp, get_cpu_clk_rate()));
```

```
			dram_size = 0;
			for (i = 0; i < CONFIG_NR_DRAM_BANKS; i++)
				dram_size += gd->bd->bi_dram[i].size;
			nand_size = 0;
	#ifdef CONFIG_CMD_NAND
			for (i = 0; i < CONFIG_SYS_MAX_NAND_DEVICE; i++)
				nand_size += nand_info[i].size;
	#endif
			//lcd_printf ("  %ld MB SDRAM, %ld MB NAND\n",
			//	dram_size >> 20,
			//	nand_size >> 20 );
	}
```

重新编译 u-boot, 生成 u-boot.bin。

注意：（貌似不用加也可以使用）
在 SAMA5D3X 中如果使用电容屏，还需要同时在 board/atmel/sama5d3xek/sama5d3xek.c 文件中添加如下蓝色内容：

```
static void sama5d3xek_lcd_hw_init(void)
{
	gd->fb_base = CONFIG_SAMA5D3_LCD_BASE;
	/* Add by JBO for int_pin and reset_pin */
	at91_set_gpio_input(AT91_PIN_PB25, 1);
	at91_set_gpio_output(AT91_PIN_PE23, 1);

	at91_lcd_hw_init();
}
```