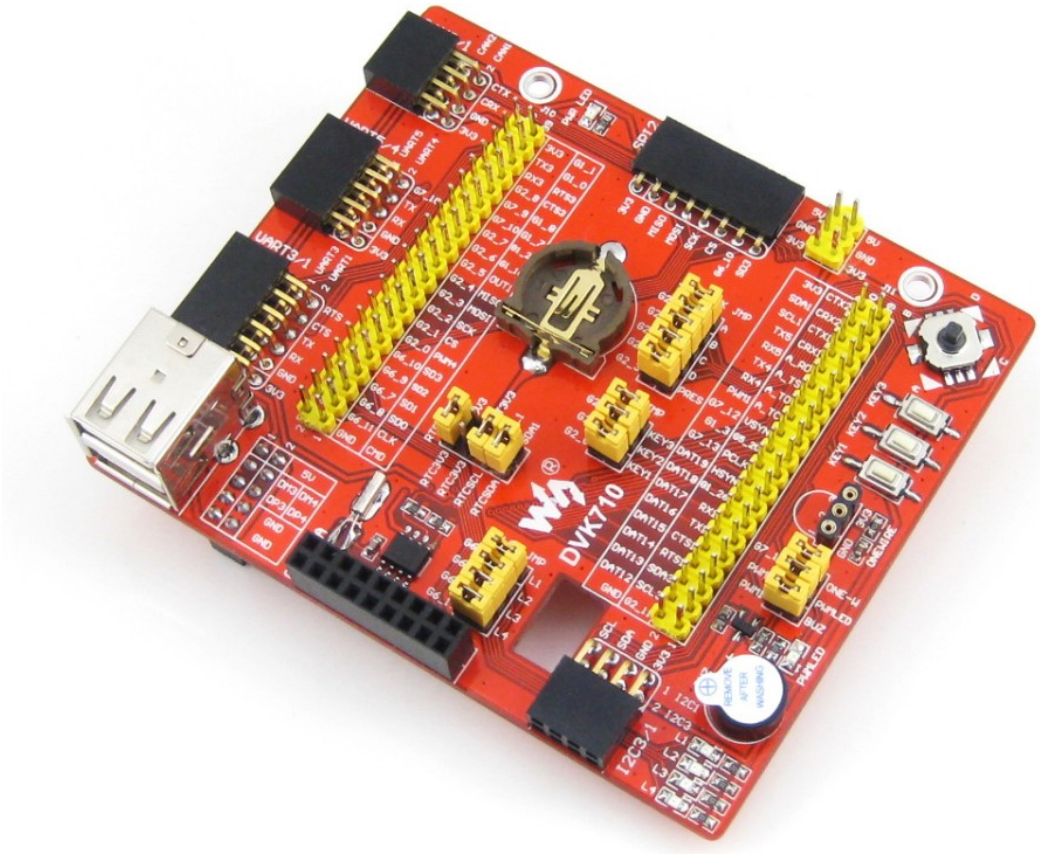
DVK710 扩展板

驱动移植手册

2014.06.03 V1.0



版权声明

版本更新记录

版本	日期	说明
V1.0	2014.06.03	初始发布

目录

特殊说明

命令前加“#”的表示 PC 机的 ubuntu 终端输入的，并且是 root 用户权限；命令前加“$”的表示开发板终端输入的。

1 按键驱动移植

1.1 管脚配置

1） 在 board-mx6q_marsboard.c 文件的

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {}结构体中，

第 332 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

```
MX6Q_PAD_NANDF_D1__GPIO_2_1,
MX6Q_PAD_NANDF_D2__GPIO_2_2,
MX6Q_PAD_NANDF_D3__GPIO_2_3,
MX6Q_PAD_NANDF_D4__GPIO_2_4,
MX6Q_PAD_NANDF_D5__GPIO_2_5,
MX6Q_PAD_NANDF_D6__GPIO_2_6,
MX6Q_PAD_NANDF_D7__GPIO_2_7,
```

1.2 添加设备资源

1）注释掉 board-mx6q_marsboard.c 的 `static struct gpio_keys_button`

marsboard_buttons[]中的内容

2）在 board-mx6q_marsboard.c 中添加

```
static struct gpio_keys_button marsboard_buttons[] = {
    GPIO_BUTTON(IMX_GPIO_NR(2,0), KEY_1, 1, "key-power", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,3), KEY_P, 1, "key-power", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,1), KEY_2, 1, "key-memu", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,4), KEY_D, 1, "key-home", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,2), KEY_3, 1, "key-back", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,7), KEY_A, 1, "volume-up", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,6), KEY_B, 1, "volume-down", 1),
    GPIO_BUTTON(IMX_GPIO_NR(2,5), KEY_C, 1, "volume-down", 1),
};
```

在 static void __init mx6_marsboard_board_init(void){}添加：

```
marsboard_add_device_buttons();
```

3）执行 make menuconfig

```
 Device Drivers    --->
    <*>    Userspace I/O drivers    --->
        <*>    Userspace I/O platform driver
            <*> Userspace I/O platform driver with generic IRQ
        handling
```

4）内核源码根目录下执行

#make uImage 编译内核，下载新内核到开发板中

2 pwm 驱动移植

2.1 管脚配置

在 board-mx6q_marsboard.c 文件

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {}结构体，

第 340 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

```
MX6Q_PAD_GPIO_9__PWM1_PWMO,
MX6Q_PAD_SD1_DAT2__PWM2_PWMO,
MX6Q_PAD_SD4_DAT1__PWM3_PWMO,
MX6Q_PAD_SD4_DAT2__PWM4_PWMO,
```

2.2 添加平台资源

1）在 board-mx6q_marsboard.c 文件

大概在第 1172 行，添加平台资源

```
static struct platform_pwm_backlight_data mx6_marsboard_buz_data =
{
        .pwm_id = 0,
        .max_brightness = 255,
        .dft_brightness = 255,
        .pwm_period_ns = 50000,
};
```

```
static struct platform_pwm_backlight_data mx6_marsboard_pwm_usr_data = {
        .pwm_id = 1,
        .max_brightness = 255,
        .dft_brightness = 255,
        .pwm_period_ns = 50000,
};
```

在 static void __init mx6_marsboard_board_init(void){}中

大概在 1321 行，添加

```
imx6q_add_mxc_pwm_backlight(0,&mx6_marsboard_buz_data);
imx6q_add_mxc_pwm_backlight(1,&mx6_marsboard_pwm_usr_data);
```

2）内核源码根目录下执行

#make uImage 编译内核，下载新内核到开发板中。

3 led 驱动移植

3.1 配置管脚

在 board-mx6q_marsboard.c 文件

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {

}结构体，

第 345 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

MX6Q_PAD_NANDF_CS0__GPIO_6_11,

MX6Q_PAD_NANDF_ALE__GPIO_6_8,

MX6Q_PAD_NANDF_CLE__GPIO_6_7,

MX6Q_PAD_NANDF_WP_B__GPIO_6_9,

3.2 配置内核

添加 Led 驱动源码 led.c，将/相关源码/ws_driver 复制到内核源码的/driver/char 下，修改 driver/char 下 Kconfig 和 Makefile

在 Kconfig 中，第 8 行，添加 source "drivers/char/ws_driver/Kconfig"

在 Makefile 中，第 71 行，添加 obj-y += ws_driver/

将 ws_driver 文件夹添加到内核中。

#make menuconfig

Device Drivers --->

Character devices --->

ws_add_drivers --->

[*] LED support

2）内核源码根目录下执行

#make uImage 编译内核，下载新内核到开发板中

4 RTC 驱动移植

4.1 修改板级文件

在 board-mx6q_marsboard.c 文件找到 static struct i2c_board_info mxc_i2c0_board_info[] __initdata = {}结构：

第 687 行，添加：

```
{
        I2C_BOARD_INFO("pcf8563", 0x51),
},
```

4.2 添加驱动文件

根目录下#make menuconfig，进入

```
Device Drivers    --->
    [*] Real Time Clock    --->
        [*]     /sys/class/rtc/rtcN (sysfs)
        [*]     /proc/driver/rtc (procfs for rtc0)
        [*]     /dev/rtcN (character devices)
        [*]     Set system time from RTC on startup and resum
        <*>     Philips PCF8563/Epson RTC8564
        <*>     Freescale SNVS Real Time Clock
```

在下面括号中填入 rtc0：

```
        (rtc0)    RTC used to set the system time
```

选中一下选项：

2）内核源码根目录下执行

#make uImage 编译内核，下载新内核到开发板中。

注意：应该要先完成第 7 节 I2C1 驱动移植

5 USB CAMERA 驱动移植

1）添加驱动文件：

根目录下#make menuconfig，进入

Device Drivers --->

<*> Multimedia support --->

<*> Video For Linu

[*] Video capture adapters --->

[*] V4L USB devices --->

<*> USB Video Class (UVC)

[*] UVC input events device support

选中驱动文件。

2）内核源码根目录下执行

#make uImage 编译内核，下载新内核到开发板中。

6 UART 驱动移植

6.1 修改板级文件

1）在 board-mx6q_marsboard.c 文件找到

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {}结构体，

第 358 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

```
MX6Q_PAD_EIM_D19__UART1_CTS,
MX6Q_PAD_EIM_D20__UART1_RTS,
MX6Q_PAD_CSI0_DAT10__UART1_TXD,
MX6Q_PAD_CSI0_DAT11__UART1_RXD,
MX6Q_PAD_EIM_EB3__UART3_RTS,
MX6Q_PAD_EIM_D23__UART3_CTS,
MX6Q_PAD_EIM_D24__UART3_TXD,
MX6Q_PAD_EIM_D25__UART3_RXD,
MX6Q_PAD_KEY_COL0__UART4_TXD,
MX6Q_PAD_KEY_ROW0__UART4_RXD,
MX6Q_PAD_KEY_COL1__UART5_TXD,
MX6Q_PAD_KEY_ROW1__UART5_RXD,
```

static inline void mx6q_marsboard_init_uart(void){}函数：

第 502 行，添加：

```
imx6q_add_imx_uart(2, NULL);
```

imx6q_add_imx_uart(3, NULL) ;

2）内核源码根目录下执行

#make uImage 编译内核，下载新内核到开发板中。

在开发板的/dev/ 可以看到 ttymxc2,ttymxc3 这两个设备

6.2 添加 ttymxc4

1）在 arch/arm/mach-mx6/board-mx6q_marsboard.c

static iomux_v3_cfg_t mx6q_marsboard_pads[] ={}结构体中

第 365 行，添加

MX6Q_PAD_KEY_COL1__UART5_TXD,

MX6Q_PAD_KEY_ROW1__UART5_RXD,

3）在 static inline void mx6q_marsboard_init_uart(void)中

第 504 行，添加

imx6q_add_imx_uart(4, NULL);

4）在 arch/arm/mach-mx6/clock.c

第 5080 行添加

_REGISTER_CLOCK("imx-uart.4", NULL, uart_clk[0]),

5）在 arch/arm/plat-mxc/devices/platform-imx-uart.c

第 137 行添加

imx6q_imx_uart_data_entry(4, 5),

6）在 arch/arm/plat-mxc/include/mach/mx6.h

第 255 行添加

#define MX6Q_UART5_BASE_ADDR UART5_BASE_ADDR

第 436 行添加

#define MX6Q_INT_UART5 MXC_INT_UART5_ANDED

注意：UART4 的移植涉及到较多的文件，修改文件后记得保存

7 I2C1 驱动移植

1）在 board-mx6q_marsboard.c 文件找到

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {}结构体，

第 350 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

MX6Q_PAD_CSI0_DAT8__I2C1_SDA,

MX6Q_PAD_CSI0_DAT9__I2C1_SCL,

修改 I2C1 的管脚与开发板的 I2C1 接口对应

8 SPI2 驱动移植

1） 在 board-mx6q_marsboard.c 文件找到

static struct spi_board_info imx6_marsboard_spi_devices[] __initdata = {}结构体

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

注释掉以下代码:

```
        {
                .modalias = "ads7846",
                .bus_num = 1,
                .chip_select = 0,
                .max_speed_hz = 1500000,
                .irq = gpio_to_irq(MX6Q_MARSBOARD_RES_TCH_INT),
                .platform_data = &ads7846_config,
        },
```

添加:

```
        {
                .modalias = "spidev",
                .bus_num = 1,
                .chip_select = 0,
                .max_speed_hz = 1500000,
        }
```

2) 配置内核

#make menuconfig

```
Device Drivers      --->
    [*] SPI support      --->
        <*> User mode SPI device driver support
```

注意：按照上面的方法移植，触摸屏将不能使用。不修改源码即可使用触摸屏功能

9 USBWIFI 驱动移植

9.1 添加驱动源码

1) 拷贝源码/rtl8188eu 到内核 drivers/net/wireless ，并修改相应的 Makefile 和 Kconfig 文件：

#vim drivers/net/wireless/Makefile

添加

```
obj-$(CONFIG_RTL8188EU)      += rtl8188eu/
```

#vi drivers/net/wireless/Kconfig

添加

```
source "drivers/net/wireless/rtl8188eu/Kconfig"
```

9.2 配置内核

#make menuconfig

```
Device Drivers      --->
        [*] Network device support      --->
            [*] Wireless LAN      --->
                <*> Realtek 8188EU USB WiFi
```

10 DS18B20 的添加

10.1 添加管脚配置

1 ）3.2 配置内核章节拷贝的 ws_driver 文件夹中，包含动源码 ds18b20.c。

在 board-mx6q_marsboard.c 文件

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {}结构体，

第 368 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

MX6Q_PAD_GPIO_18__GPIO_7_13,

10.2 配置内核

#make menuconfig

Device Drivers --->

Character devices --->

ws_add_drivers --->

[*] DS18B20 support

11.RS485 的添加

11.1 添加管脚配置

1 ） 3.2 配置内核章节拷贝的 ws_driver 文件夹中包含动源码

RS485.c。

在 board-mx6q_marsboard.c 文件

static iomux_v3_cfg_t mx6q_marsboard_pads[] = {}结构体，

第 370 行，添加管脚的初始化配置。

#vim arch/arm/mach-mx6/board-mx6q_marsboard.c

MX6Q_PAD_EIM_D23__GPIO_3_23

MX6Q_PAD_SD4_CLK__GPIO_7_10,

11.2 配置内核

#make menuconfig

Device Drivers --->

Character devices --->

ws_add_drivers --->

[*] RS485 support

12.SPI CS 的添加

12.1 添加管脚配置

1 ） 3.2 配置内核章节拷贝的 ws_driver 文件夹中包含动源码

spi_io.c。

2 ）配置内核

#make menuconfig

```
Device Drivers    --->
    Character devices    --->
        ws_add_drivers    --->
            [*]    spi2_cs
```