从用户空间的 GPIO Sysfs 接口访问用户空间的 GPIO

原文: https://community.nxp.com/docs/DOC-343344

由 Biyong Sun 于 2019-5-8 上传的文件

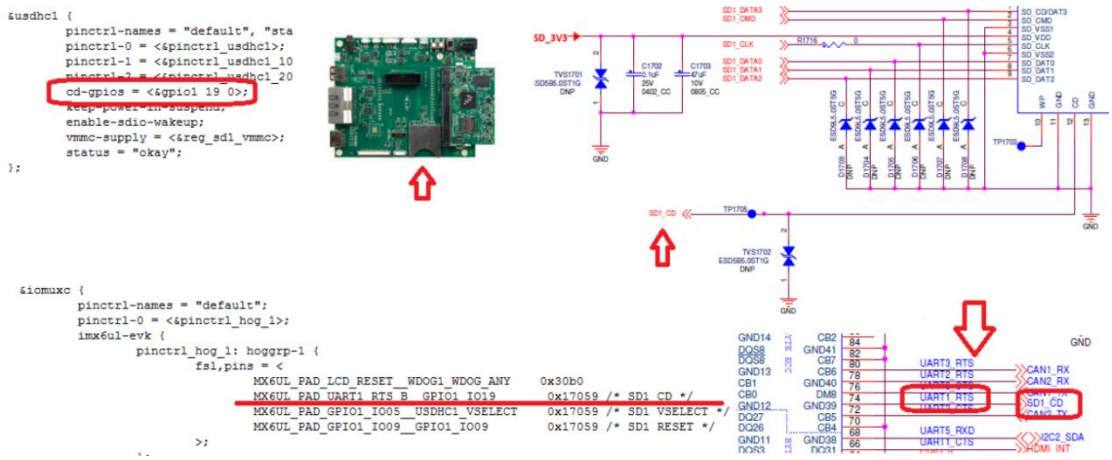
用户空间的 GPIO Sysfs 接口

后面文件的详细信息：

rel_imx_3.14.38_6ul_ga/kernel_imx/Documentation/gpio/sysfs.txt
rel_imx_3.14.28_1.1.0_ga/kernel_imx/Documentation/gpio/sysfs.txt
rel_imx_3.10.53_1.1.0_ga/ernel_imx/Documentation/gpio.txt

i.MX6UL 评估板示例

这个测试以 SD1（usdhc1）上的 cd 引脚为例（GPIO1，bit 19）。



i.MX6UL 评估板示例（续）

目前 CD 引脚（GPIO1，bit 19）属于 usdhc1。
无法导出到系统。 它将显示 busy 的错误类型。

root @ imx6ulevk：~ # echo19> / sys / class / gpio / export

-sh：echo：写错误：设备或资源忙

但是仍然可以通过 debugfs 检查 GPIO 状态

通过以下命令检查 CD 的状态： cat / sys / kernel / debug / gpio in debugfs。

```
SDCRAD plug out:

root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
 gpio-3   (tsc_X-            ) in  hi
 gpio-9   (VSD_3V3           ) out lo
 gpio-19  (2190000.usdhc cd  ) in  hi

GPIOs 32-63, platform/20a0000.gpio, 20a0000.gpio:

GPIOs 64-95, platform/20a4000.gpio, 20a4000.gpio:

GPIOs 96-127, platform/20a8000.gpio, 20a8000.gpio:
```

```
SDCRAD plug in:

root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
 gpio-3   (tsc_X-            ) in  hi
 gpio-9   (VSD_3V3           ) out lo
 gpio-19  (2190000.usdhc cd  ) in  lo

GPIOs 32-63, platform/20a0000.gpio, 20a0000.gpio:

GPIOs 64-95, platform/20a4000.gpio, 20a4000.gpio:

GPIOs 96-127, platform/20a8000.gpio, 20a8000.gpio:
```

通过修改设备树，从 usdhc1 中删除 GPIO1，bit 19。注释下面红色这一行。然后重新编译生成新的 dtb 文件。

```
&usdhc1 {
pinctrl-names = "default", "state_100mhz", "state_200mhz";
pinctrl-0 = <&pinctrl_usdhc1>;
pinctrl-1 = <&pinctrl_usdhc1_100mhz>;
pinctrl-2 = <&pinctrl_usdhc1_200mhz>;
// cd-gpios = <&gpio1 19 0>;
keep-power-in-suspend;
enable-sdio-wakeup;
vmmc-supply = <®_sd1_vmmc>;
status = "okay";
};
```

用新的 dtb 来启动电路板，并确保 GPIO 不在 debugfs 中显示。

```
root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
 gpio-3   (tsc_X-            ) in  hi
 gpio-9   (VSD_3V3           ) out lo

GPIOs 32-63, platform/20a0000.gpio, 20a0000.gpio:
```

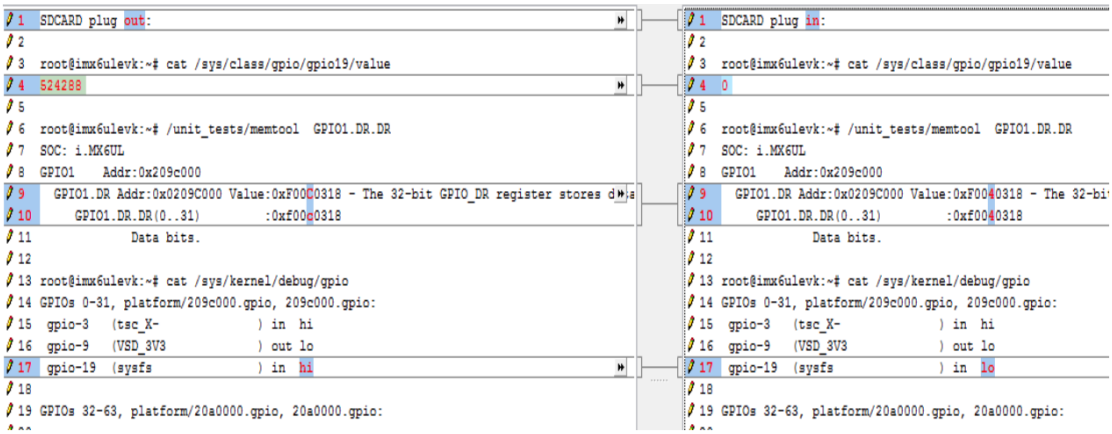
```
root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
 gpio-3   (tsc_X-            ) in  hi
 gpio-9   (VSD_3V3           ) out lo
 gpio-19  (2190000.usdhc cd  ) in  lo

GPIOs 32-63, platform/20a0000.gpio, 20a0000.gpio:
```

将 i.MX6UL GPIO1 bit 19 导出到系统：

```
root@imx6ulevk:~# ls /sys/class/gpio/
export gpiochip0 gpiochip248 gpiochip64 unexport
gpio132 gpiochip128 gpiochip32 gpiochip96
echo 19 > /sys/class/gpio/export
root@imx6ulevk:~# ls /sys/class/gpio/
export gpio19 gpiochip128 gpiochip32 gpiochip96
gpio132 gpiochip0 gpiochip248 gpiochip64 unexport
```

检查状态：



GPIO 输出：

```
root@imx6ulevk:~# echo 19 > /sys/class/gpio/export
root@imx6ulevk:~# echo out > /sys/class/gpio/gpio19/direction
root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
gpio-3 (tsc_X-      ) in hi
gpio-9 (VSD_3V3      ) out lo
gpio-19 (sysfs      ) out lo

GPIOs 32-63,platform/20a0000.gpio,20a0000.gpio:

root@imx6ulevk:~# /unit_tests/memtool GPIO1.DR.DR
SOC: i.MX6UL
GPIO1 Addr:0x209c000
  GPIO1.DR Addr:0x0209C000 Value:0xF0040118 - 32-bit GPIO_DR 寄存器
存储的数据被驱动到输出线。
    GPIO1.DR.DR(0..31) :0xf0040118
        Data bits.

root@imx6ulevk:~# /unit_tests/memtool GPIO1.DR.DR
```

SOC: i.MX6UL
GPIO1 Addr:0x209c000
GPIO1.DR Addr:0x0209C000 Value:0xF00C0118 - 32-bit GPIO_DR 寄存器存储的数据被驱动到输出线。
GPIO1.DR.DR(0..31) :0xf00c0118
Data bits.
由于输出，调试 fs 不能看到更改。
启用 SION 环回系统。

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUXC), also enable SION if need to read loopback pad value through PSR
2. Configure GPIO direction register to output (GPIO_GDIR[GDIR] set to 1b).
3. Write value to data register (GPIO_DR).

A pseudocode description to drive 4'b0101 on [output3:output0] is as follows:

```
// SET PADS TO GPIO MODE VIA IOMUX.
write sw_mux_ctl_pad_<output[0-3]>.mux_mode, <GPIO_MUX_MODE>
// Enable loopback so we can capture pad value into PSR in output mode
write sw_mux_ctl_pad_<output[0-3]>.sion, 1
// SET GDIR=1 TO OUTPUT BITS.
write GDIR[31:4,output3_bit,output2_bit, output1_bit, output0_bit,] 32'hxxxxxxxF
// WRITE OUTPUT VALUE=4'b0101 TO DR.
write DR, 32'hxxxxxxx5
// READ OUTPUT VALUE FROM PSR ONLY.
read_cmp PSR, 32'hxxxxxxx5
```

```
root@imx6ulevk:~#
/unit_tests/memtoolIOMUXC.SW_MUX_CTL_PAD_UART1_RTS_B.SION=1
SOC: i.MX6UL
write 0x00000001 to Bit 4..4 of 0x020E0090

root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
gpio-3 (tsc_X-      ) in hi
gpio-9 (VSD_3V3      ) out lo
gpio-19 (sysfs      ) out hi

GPIOs 32-63, platform/20a0000.gpio, 20a0000.gpio:

root@imx6ulevk:~# echo 0 > /sys/class/gpio/gpio19/value
root@imx6ulevk:~# cat /sys/kernel/debug/gpio
GPIOs 0-31, platform/209c000.gpio, 209c000.gpio:
gpio-3 (tsc_X-      ) in hi
gpio-9 (VSD_3V3      ) out lo
gpio-19 (sysfs      ) out lo
```

如何计算 GPIO 数量？
Sysfs.txt:

```
/sys/class/gpio/gpiochipN/
      "base" ... same as N, the first GPIO managed by this chip
      "label" ... provided for diagnostics (not always unique)
      "ngpio" ... how many GPIOs this manges (N to N + ngpio - 1)
```

i.MX6ul evk:

```
root@imx6ulevk:~# cat /sys/class/gpio/gpiochip0/base
0
root@imx6ulevk:~# cat /sys/class/gpio/gpiochip0/label
209c000.gpio
root@imx6ulevk:~# cat /sys/class/gpio/gpiochip0/ngpio
32
```

对于 linux 系统，i.MX6UL GPIO1 的范围是 N 到 N+gpio-1;0 到 0+32-1;0 到 31
对于 Linux 系统，i.MX6UL GPIO1 的 bit 19 为 0+19=19
命令：echo 19>/sys/class/gpio/export
注意：从 209c000 中我们可以知道它是 i.MX6UL GPIO1

i.MX6UL GPIO3 示例：

```
root@imx6ulevk:~# cat /sys/class/gpio/gpiochip64/base
64
root@imx6ulevk:~# cat /sys/class/gpio/gpiochip64/label
20a4000.gpio
root@imx6ulevk:~# cat /sys/class/gpio/gpiochip0/ngpio
32
```

i.MX6UL GPIO3 bit 5 导出到 linux 系统：
5 + 64 = 69
命令行：
echo 69>/sys/class/gpio/export

附件：Accessing_GPIO_From_UserSpace (3).pdf