S3C2410 GPIO Control

Introduction

The s3c2410 kernel provides an interface to configure and manipulate the state of the GPIO pins, and find out other information about them.

There are a number of conditions attached to the configuration of the s3c2410 GPIO system, please read the Samsung provided data-sheet/users manual to find out the complete list.

See Documentation/arm/Samsung/GPIO.txt for the core implemetation.

GPIOLIB

With the event of the GPIOLIB in drivers/gpio, support for some of the GPIO functions such as reading and writing a pin will be removed in favour of this common access method.

Once all the extant drivers have been converted, the functions listed below will be removed (they may be marked as __deprecated in the near future).

The following functions now either have a s3c_ specific variant or are merged into gpiolib. See the definitions in arch/arm/plat-samsung/include/plat/gpio-cfg.h:

s3c2410_gpio_setpin()	gpio_set_value() or gpio_direction_output()
s3c2410_gpio_getpin()	gpio_get_value() or gpio_direction_input()
s3c2410_gpio_getirq()	gpio_to_irq()
s3c2410_gpio_cfgpin()	s3c_gpio_cfgpin()
s3c2410_gpio_getcfg()	s3c_gpio_getcfg()
s3c2410_gpio_pullup()	s3c_gpio_setpull()

GPIOLIB conversion

If you need to convert your board or driver to use gpiolib from the exiting s3c2410 api, then here are some notes on the process.

1) If your board is exclusively using an GPIO, say to control peripheral power, then it will require to claim the gpio with gpio_request() before it can use it.

   It is recommended to check the return value, with at least WARN_ON() during initialisation.

2) The s3c2410_gpio_cfgpin() can be directly replaced with s3c_gpio_cfgpin() as they have the same arguments, and can either take the pin specific values, or the more generic special-function-number arguments.

3) s3c2410_gpio_pullup() changs have the problem that whilst the
   s3c2410_gpio_pullup(x, 1) can be easily translated to the
   s3c_gpio_setpull(x, S3C_GPIO_PULL_NONE), the s3c2410_gpio_pullup(x, 0)
   are not so easy.

   The s3c2410_gpio_pullup(x, 0) case enables the pull-up (or in the case
   of some of the devices, a pull-down) and as such the new API distinguishes
   between the UP and DOWN case. There is currently no 'just turn on' setting
   which may be required if this becomes a problem.

4) s3c2410_gpio_setpin() can be replaced by gpio_set_value(), the old call
   does not implicitly configure the relevant gpio to output. The gpio
   direction should be changed before using gpio_set_value().

5) s3c2410_gpio_getpin() is replaceable by gpio_get_value() if the pin
   has been set to input. It is currently unknown what the behaviour is
   when using gpio_get_value() on an output pin (s3c2410_gpio_getpin
   would return the value the pin is supposed to be outputting).

6) s3c2410_gpio_getirq() should be directly replacable with the
   gpio_to_irq() call.

The s3c2410_gpio and gpio_ calls have always operated on the same gpio
numberspace, so there is no problem with converting the gpio numbering
between the calls.

Headers

See arch/arm/mach-s3c2410/include/mach/regs-gpio.h for the list
of GPIO pins, and the configuration values for them. This
is included by using #include <mach/regs-gpio.h>

The GPIO management functions are defined in the hardware
header arch/arm/mach-s3c2410/include/mach/hardware.h which can be
included by #include <mach/hardware.h>

A useful amount of documentation can be found in the hardware
header on how the GPIO functions (and others) work.

Whilst a number of these functions do make some checks on what
is passed to them, for speed of use, they may not always ensure
that the user supplied data to them is correct.

PIN Numbers

Each pin has an unique number associated with it in regs-gpio.h,
eg S3C2410_GPA(0) or S3C2410_GPF(1). These defines are used to tell
the GPIO functions which pin is to be used.

With the conversion to gpiolib, there is no longer a direct conversion
from gpio pin number to register base address as in earlier kernels. This

is due to the number space required for newer SoCs where the later
GPIOs are not contiguous.

Configuring a pin

The following function allows the configuration of a given pin to
be changed.

```
void s3c2410_gpio_cfgpin(unsigned int pin, unsigned int function);
```

Eg:

```
s3c2410_gpio_cfgpin(S3C2410_GPA(0), S3C2410_GPA0_ADDR0);
s3c2410_gpio_cfgpin(S3C2410_GPE(8), S3C2410_GPE8_SDDAT1);
```

which would turn GPA(0) into the lowest Address line A0, and set
GPE(8) to be connected to the SDIO/MMC controller's SDDAT1 line.

The s3c_gpio_cfgpin() call is a functional replacement for this call.

Reading the current configuration

The current configuration of a pin can be read by using:

```
s3c2410_gpio_getcfg(unsigned int pin);
```

The return value will be from the same set of values which can be
passed to s3c2410_gpio_cfgpin().

The s3c_gpio_getcfg() call should be a functional replacement for
this call.

Configuring a pull-up resistor

A large proportion of the GPIO pins on the S3C2410 can have weak
pull-up resistors enabled. This can be configured by the following
function:

```
void s3c2410_gpio_pullup(unsigned int pin, unsigned int to);
```

Where the to value is zero to set the pull-up off, and 1 to enable
the specified pull-up. Any other values are currently undefined.

The s3c_gpio_setpull() offers similar functionality, but with the
ability to encode whether the pull is up or down. Currently there
is no 'just on' state, so up or down must be selected.

Getting the state of a PIN

  The state of a pin can be read by using the function:

```
    unsigned int s3c2410_gpio_getpin(unsigned int pin);
```

  This will return either zero or non-zero. Do not count on this
  function returning 1 if the pin is set.

  This call is now implemented by the relevant gpiolib calls, convert
  your board or driver to use gpiolib.

Setting the state of a PIN

  The value an pin is outputing can be modified by using the following:

```
    void s3c2410_gpio_setpin(unsigned int pin, unsigned int to);
```

  Which sets the given pin to the value. Use 0 to write 0, and 1 to
  set the output to 1.

  This call is now implemented by the relevant gpiolib calls, convert
  your board or driver to use gpiolib.

Getting the IRQ number associated with a PIN

  The following function can map the given pin number to an IRQ
  number to pass to the IRQ system.

```
   int s3c2410_gpio_getirq(unsigned int pin);
```

  Note, not all pins have an IRQ.

  This call is now implemented by the relevant gpiolib calls, convert
  your board or driver to use gpiolib.

Authour

Ben Dooks, 03 October 2004
Copyright 2004 Ben Dooks, Simtec Electronics