**Request a GPIO**

#include <asm/gpio.h>

**int gpio\_request(unsigned gpio, const char \*label);**

gpio: gpio number to request, must be a valid number

label: text string for diagnostic

return: 0 on success

<0 on error

Ex: ret = gpio\_request(16, "gpio\_out"); //request GPIO#15 label "gpio\_out"

**Release a previously requested GPIO given its GPIO number**

#include <asm/gpio.h>

**void gpio\_free(unsigned gpio);**

gpio: GPIO number to free, previously requested

return: nothing

Ex: gpio\_free(16);

**Set the direction of a previously requested GPIO**

#include <asm/gpio.h>

**int gpio\_direction\_input(unsigned gpio);**

**int gpio\_direction\_output(unsigned gpio, int value);**

gpio: specifies the number of a successfully requested GPIO pin

value: initialization level after configuration

boolean: zero means low, non-zero means high

return: 0 on success

negative errno on error

**Access a previously configured GPIO given its GPIO number**

#include <asm/gpio.h>

**int gpio\_get\_value(unsigned gpio);**

gpio: specified number of a configured GPIO

return: zero for low

non-zero for high

**void gpio\_set\_value(unsigned gpio, int value);**

gpio: specifies number of a configured GPIO

value: zero for low

non-zero for high

Ex: gpio\_set\_value(16, 1);

**Request and configure a single GPIO**

#include <asm/gpio.h>

**int gpio\_request\_one(unsigned gpio, unsigned long flags, const char \*label);**

gpio, label, return value are the same as gpio\_request()

flags: GPIOF\_DIR\_IN configure direction as input

GPIOF\_DIR\_OUT configure direction as output

GPIOF\_OUT\_INIT\_LOW configure GPIO as output, logic level low

GPIOF\_OUT\_INIT\_HIGH configure GPIO as output, logic level high

Ex:

int ret;

ret = gpio\_request\_one(16, GPIOF\_OUT\_LOW, “out\_0”);

**Request/free and configure multiple GPIOs in a single call**

#include <asm/gpio.h>

struct gpio {

unsigned gpio; // GPIO number

unsigned long flags;

const char \*label; // Text string

}

**int gpio\_request\_array(struct gpio \*a, size\_t n);**

**void gpio\_free\_array(struct gpio \*a, size\_t n);**

n: size of array passed in argument a, used ARRAY\_SIZE to find n

return: 0 on success

negative errno on error

Ex:

#include <asm/gpio.h>

static struct gpio gpio\_conf[] = {

{16, GPIOF\_OUT\_INIT\_LOW, “gpiout0”},

{17, GPIOF\_OUT\_INIT\_LOW, “gpiout1”},

{11, GPIOF\_OUT\_INIT\_LOW, “gpiout2”},

{10, GPIOF\_OUT\_INIT\_LOW, “gpiout3”},

}

int ret;

ret = gpio\_request\_array(gpio\_conf, ARRAY\_SIZE(gpio\_conf);