Subsystem drivers using GPIO

Note that standard kernel drivers exist for common GPIO tasks and will provide the right in-kernel and userspace APIs/ABIs for the job, and that these drivers can quite easily interconnect with other kernel subsystems using hardware descriptions such as device tree or ACPI:

- leds-gpio: drivers/leds/leds-gpio.c will handle LEDs connected to GPIO lines, giving you the LED sysfs interface
- ledtrig-gpio: drivers/leds/trigger/ledtrig-gpio.c will provide a LED trigger, i.e. a LED will turn on/off in response to a GPIO line going high or low (and that LED may in turn use the leds-gpio as per above).
- gpio-keys: drivers/input/keyboard/gpio_keys.c is used when your GPIO line can generate interrupts in response to a key press. Also supports debounce.
- gpio-keys-polled: drivers/input/keyboard/gpio_keys_polled.c is used when your GPIO line cannot generate interrupts, so it needs to be periodically polled by a timer.
- gpio_mouse: drivers/input/mouse/gpio_mouse.c is used to provide a mouse with up to three buttons by simply using GPIOs and no mouse port. You can cut the mouse cable and connect the wires to GPIO lines or solder a mouse connector to the lines for a more permanent solution of this type.
- gpio-beeper: drivers/input/misc/gpio-beeper.c is used to provide a beep from an external speaker connected to a GPIO line.
- extcon-gpio: drivers/extcon/extcon-gpio.c is used when you need to read an external connector status, such as a headset line for an audio driver or an HDMI connector. It will provide a better userspace sysfs interface than GPIO.
- restart-gpio: drivers/power/reset/gpio-restart.c is used to restart/reboot the system by pulling a GPIO line and will register a restart handler so userspace can issue the right system call to restart the system.
- poweroff-gpio: drivers/power/reset/gpio-poweroff.c is used to power the system down by pulling a GPIO line and will register a pm_power_off() callback so that userspace can issue the right system call to power down the system.
- gpio-gate-clock: drivers/clk/clk-gpio.c is used to control a gated clock (off/on) that uses a GPIO, and integrated with the clock subsystem.
- i2c-gpio: drivers/i2c/busses/i2c-gpio.c is used to drive an I2C bus (two wires, SDA and SCL lines) by hammering (bitbang) two GPIO lines. It will appear as any other I2C bus to the system and makes it possible to connect drivers for the I2C devices on the bus like any other I2C bus driver.
- spi_gpio: drivers/spi/spi-gpio.c is used to drive an SPI bus (variable number of wires, at least SCK and optionally MISO, MOSI and chip select lines) using GPIO hammering (bitbang). It will appear as any other SPI bus on the system and makes it possible to connect drivers for SPI devices on the bus like any other SPI bus driver. For example any MMC/SD card can then be connected to this SPI by using the mmc_spi host from the MMC/SD card subsystem.
- w1-gpio: drivers/w1/masters/w1-gpio.c is used to drive a one-wire bus using a GPIO line, integrating with the W1 subsystem and handling devices on the bus like any other W1 device.
- gpio-fan: drivers/hwmon/gpio-fan.c is used to control a fan for cooling the system, connected to a GPIO line (and optionally a GPIO alarm line), presenting all the right in-kernel and sysfs interfaces to make your system not overheat.
- gpio-regulator: drivers/regulator/gpio-regulator.c is used to control a regulator providing a certain voltage by pulling a GPIO line, integrating with the regulator subsystem and giving you all the right interfaces.
- gpio-wdt: drivers/watchdog/gpio_wdt.c is used to provide a watchdog timer that will periodically "ping" a hardware connected to a GPIO line by toggling it from 1-to-0-to-1. If that hardware does not receive its "ping" periodically, it will reset the system.
- gpio-nand: drivers/mtd/nand/raw/gpio.c is used to connect a NAND flash chip to a set of simple GPIO lines: RDY, NCE, ALE, CLE, NWP. It interacts with the NAND flash MTD subsystem and provides chip access and partition parsing like any other NAND driving hardware.
- ps2-gpio: drivers/input/serio/ps2-gpio.c is used to drive a PS/2 (IBM) serio bus, data and clock line, by bit banging two GPIO lines. It will appear as any other serio bus to the system and makes it possible to connect drivers for e.g. keyboards and other PS/2 protocol based devices.
- cec-gpio: drivers/media/platform/cec-gpio/ is used to interact with a CEC Consumer Electronics Control bus using only GPIO. It is used to communicate with devices on the HDMI bus.
- gpio-charger: drivers/power/supply/gpio-charger.c is used if you need to do battery charging and all you have to go by to check the presence of the AC charger or more complex tasks such as indicating charging status using nothing but GPIO lines, this driver provides that and also a clearly defined way to pass the charging parameters from hardware descriptions such as the device tree.
- gpio-mux: drivers/mux/gpio.c is used for controlling a multiplexer using n GPIO lines such that you can mux in 2^n different devices by activating different GPIO lines. Often the GPIOs are on a SoC and the devices are some SoC-external entities, such as different components on a PCB that can be selectively enabled.

Apart from this there are special GPIO drivers in subsystems like MMC/SD to read card detect and write protect GPIO lines, and in the TTY serial subsystem to emulate MCTRL (modem control) signals CTS/RTS by using two GPIO lines. The MTD NOR flash has add-ons for extra GPIO lines too, though the address bus is usually connected directly to the flash.

Use those instead of talking directly to the GPIOs from userspace; they integrate with kernel frameworks better than your userspace code could. Needless to say, just using the appropriate kernel drivers will simplify and speed up your embedded hacking in particular by providing ready-made components.