Userland Tools and Techniques for Board Bring Up and Systems Integration

ELC 2012
Agenda

* Introduction
* What?
* Why bother with userland?
* Common SoC interfaces
* Typical Scenario
* Kernel setup
* GPIO/UART/I2C/SPI/Other
* Questions
Introduction

* SoC offer a lot of integrated functionality
* System designs differ by outside parts
* Most mobile systems are SoC
* "CPU boards" for SoCs
* Available BSP for starting
  * Vendor or other sources
* Common Unique components
  * Memory (RAM)
  * Storage ("flash")
  * IO
* Displays
* Power supplies
What?

* IO related items
* I2C
* SPI
* UART
* GPIO
* USB
Why userland?

* Easier for non kernel savvy
* Quicker turn around time
* Easier to debug
* Often times available already
  Sample userland from BSP/LSP vendor
* Kernel driver is not ready
Common SoC interfaces

Most SoC have these and more:
* Pinmux
* UART
* GPIO
* I2C
* SPI
* USB
* Not discussed: Audio/Displays
Typical Scenario

Custom board:
* Load code/Bring up memory
* Setup memory controller for part used
* Load Linux
* Toggle lines on board to verify

Prototype based on demo/eval board:
* Start with board at a shell prompt
* Get newly attached hw to work
Kernel Setup

Additions to typical configs:
* Enable UART support (typically done already)
* Enable I2C support along with drivers for the SoC
  * (CONFIG_I2C + other)
* Enable SPIdev
  * (CONFIG_SPI + CONFIG_SPI_SPIDEV)
  * Add SPIDEV to board file
* Enable GPIO sysfs
  * (CONFIG_GPIO + other + CONFIG_GPIO_SYSFS)
* Enable USB
  * Depends on OTG vs normal, etc.
**GPIO**

* Sysfs supported. sysfs should be mounted:
  
  mount -t sysfs sysfs /sys

* /sys/class/gpio/control
  
  Used to enable access
  
  * export N
  
  * unexport N

* /sys/class/gpio/gpio-N/value
  
  Used to set the state of a pin

* /sys/class/gpio/gpio-N/direction
  
  Used to set input or output

* Kernel may exclusively grab a GPIO
UART

* Basic configuration: stty
  * Port specified by -F (GNU) or stdin (others, works with GNU too)
* Data transfer: /dev/ttyXXX
* Make sure device node exists
* Example:
  stty 4800 < /dev/ttyO2
  cat /dev/ttyO2
  echo "TEXT" > /dev/ttyO2
I2C

* Make sure /dev/i2c-* exists
* i2ctools (from lmsensors project)
  * i2cdetect (i2detect -l)
  * i2cdump
  * i2cset
  * i2cget
* Note bus number!
* Devices may be claimed by kernel
* 7bit vs 8bit addresses
SPI

* `/dev/spidevN`
  * Author recommends udev
  * Can check `/sys/dev` for minor number.
    Look for major 153. i.e. 153:*  
    Then create with `mknod`
  * Need to bind the spidev device
to the right chipselect/bus in your
board file.
  * IO needs to be done with a program.  
No libraries besides libc.
SPI (con't)

static struct omap2_mcspi_device_config spidev_spi_chip_info = {
      .turbo_mode     = 0,
      .single_channel = 1,  /* 0: slave, 1: master */
};
static struct spi_board_info b_spi_board_info[] __initdata = {
      { /* SPIDEV */
         .modalias       = "spidev",
         .bus_num        = 3,
         .chip_select    = 0,
         .max_speed_hz   = 2000000,
         .controller_data = &spidev_spi_chip_info,
         .mode           = SPI_MODE_3,
      },
};
static void __init omap3_beagle_init(void)
{
    ...
    spi_register_board_info(b_spi_brd_info, ARRAY_SIZE(b_spi_brd_info));
    ...
}
Input Devices

* Buttons, keyboards, touch screens, sensors(!), etc.
* /dev/input/eventN
* evtest
* Android specific: getevents
* /sys/class/input
  * Some devices may need to be enabled
* /proc/bus/input/devices
  * List mapping of driver to device instance
USB

* Limited userland debugging
* CONFIG_USB_ANNOUNCE_NEW_DEVICES
* dmesg
* lsusb
* sysfs
* Complicated
LED

* Sysfs
* CONFIG_NEW_LEDS
* echo 1 > /sys/class/led/name/brightness
* Check polarity
* Need board file entry
LED

*static struct gpio_led gpio_leds[] = {

    {                .name                   = "beagleboard::usr0",                .gpio                   = 150,        ... gpio_led_info = {        .leds           = gpio_leds,        .num_leds       = ARRAY_SIZE(gpio_leds),};

    static struct gpio_led_platform_data gpio_led_info = {
        .leds            = gpio_leds,        .num_leds        = ARRAY_SIZE(gpio_leds),  
    };
Other

* The big hammer: devmem2
* Direct physical memory access
* Backdoor for a lot of things
* Useful if there are things hooked up to a memory/general purpose bus.
* lsusb
* sysfs in general
Conclusions

* Many things unique to a system can be tested with just a functional userland.
* Numerous existing utilities.
* Can accelerate development.
* This is just the beginning. More userland support is being added through sysfs and debugfs.
References

* Documentation/gpio.txt
* Documentation/spi/spidev
* Documentation/leds-class.txt
* http://www.lm-sensors.org/wiki/I2CTools
* http://sources.buildroot.net/devmem2.c
* http://elinux.org/File:Evtest.c
Questions?