The Serial Device Bus

Johan Hovold

Hovold Consulting AB

johan@hovoldconsulting.com
johan@kernel.org

September 29, 2017

Introduction

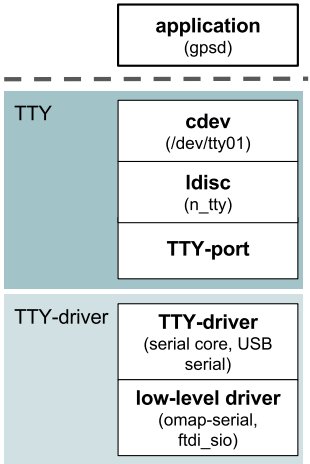
- UARTs and RS-232 have been around since 1960s
- USB, PCIe, Firewire, Thunderbolt
- Common interface for Bluetooth, NFC, FM Radio and GPS
- TTY-layer abstracts serial connection
  - Character-device interface (e.g. `/dev/ttyS1`)
- But no (good) way to model associated resources (PM)
  - GPIOs and interrupts
  - Regulators
  - Clocks
  - Audio interface
- Kernel support limited to line-discipline "drivers"
  - Requires user-space to configure and initialise

Outline

- TTY Layer
- Problems with line-discipline drivers
- Serdev implementation
- Serdev interfaces
- Serdev limitations
- Future work

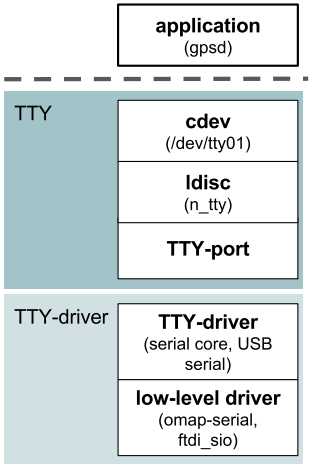
TTY Layer

- Character device
- Line disciplines
  - I/O processing
  - Canonical mode
  - Echos
  - Errors
  - Signals on input
- TTY-ports
  - Input buffering
  - Abstraction layer (e.g. open())
- TTY-drivers



User Space Drivers

- Description
  - In user space
  - Port
- Associated resources
  - GPIOs and interrupts (accessible)
  - Regulators
  - Clocks
- Power management
- Firmware loading



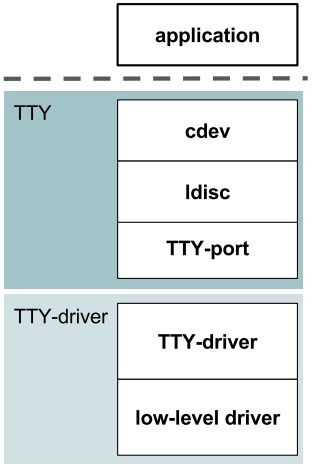
Kernel Drivers

- Interaction with other subsystems
- Line-discipline drivers (e.g. bluetooth, input, nfc, ppp)
- Registers further devices (e.g. hci0)
- User-space daemons to initialise port and switch line-discipline
  - `ldattach`
  - `inputattach`
  - `hciattach`
- Remains registered while port (and ldisc) is open
- Firmware infrastructure available
- But still issues with other resources and PM

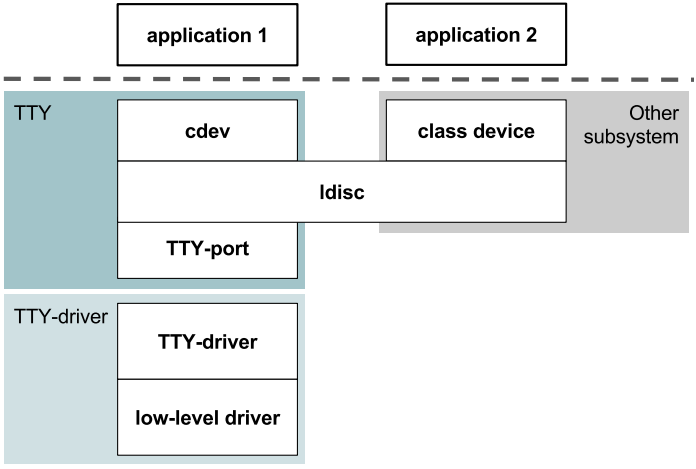
HCI registration example

```
int ldisc = N_HCI;
int proto = HCI_UART_BCM;

fd = open("/dev/ttyO1", ...);

/* configure line settings */

ioctl(fd, TIOCSETD, &ldisc);
ioctl(fd, HCIUARTSETPROTO, proto);
```

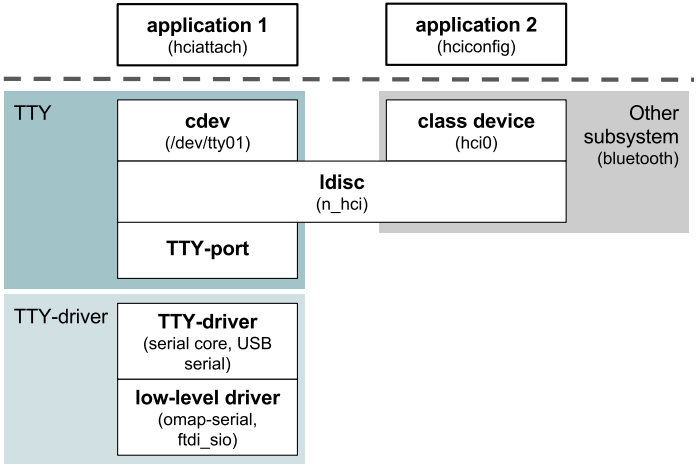
Kernel Drivers



Kernel Drivers



Kernel Drivers



Problems with ldisc drivers

- Description (what, where, how?) and discovery
  - Encoded in user space rather than firmware (DT, ACPI)
  - User-space daemons
- Description and lookup of associated resources
  - GPIOs and interrupts (e.g. reset, wakeup)
  - Pinctrl
  - Regulators
  - Clocks
- Power management
  - GPIOs, regulators, clocks...
  - Open port may prevent underlying device from runtime suspending
- Firmware loading
  - GPIO (e.g. reset) interaction

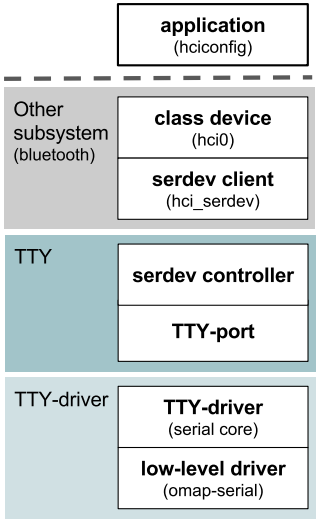
The Serial Device Bus

- The Serial Device Bus (serdev)
- By Rob Herring (Linaro)
- Bus for UART-attached devices
  - Replace ti-st driver and UIM daemon
  - Earlier efforts (power management)
- Merged in 4.11
- Enabled for serial core only in 4.12 (due to lifetime issues)

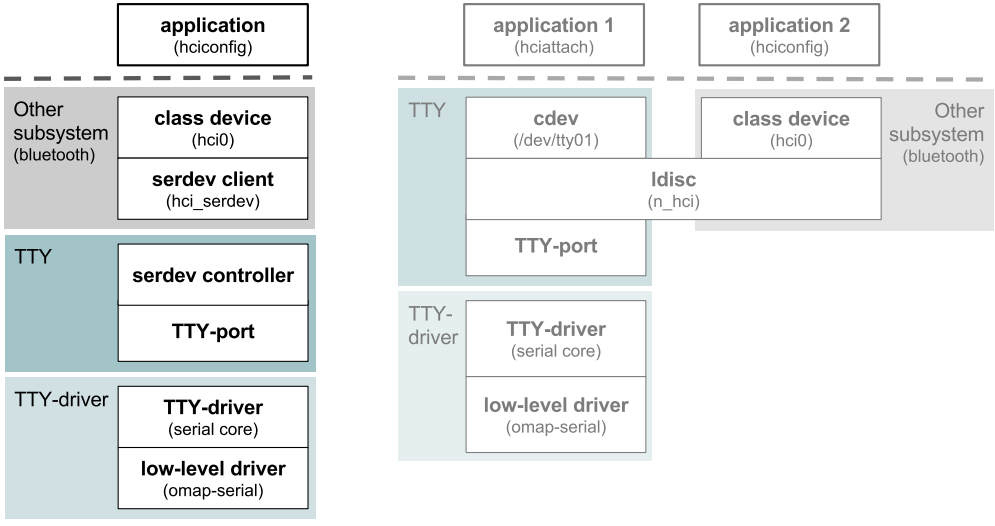
Serdev Overview

- New "serial" bus type
- Serdev controllers
- Serdev clients (a.k.a. slaves or devices)
- Serdev TTY-port controller
  - The only controller implementation
  - Registered by TTY-driver when there are clients
  - Controller replaces TTY character device
- Clients described by firmware (Device Tree)

Serdev Drivers



Serdev Drivers



Serdev Implementation

- TTY-port clients
  - Default client (ldisc)
  - Serdev TTY-port client (controller)
- Register controller and slaves when registering TTY-port

```
struct tty_port {
    ...
    struct tty_port_client_operations *client_ops;
    void *client_data;
};

struct tty_port_client_operations {
    int (*receive_buf)(...);
    void (*write_wakeup)(...);
};
```

Device Tree Bindings

- Child of serial-port node
- `compatible` property
- `max-speed` property (optional)
- Additional resources

```
&uart1 {
    bluetooth {
        compatible = "ti,wl1835-st";
        enable-gpios = <&gpio1 7 0>;
        clocks = <&clk32k_wl18xx>;
        clock-names = "ext_clock";
    };
};
```

Sysfs Example

```
/sys/bus/platform/devices/
|-- 44e09000.serial
|    |-- driver -> .../omap_uart
|    `-- tty
|        `-- tty0
`-- 48022000.serial
    |-- driver -> .../omap_uart
    `-- serial0
        `-- serial0-0
            |--bluetooth
            |  `-- hci0
            |-- subsystem -> .../bus/serial
            `-- driver -> .../hci-ti
```

Driver Interface

- Resembles line-discipline operations
  - open and close
  - terminal settings
  - write
  - modem control
  - read (callback)
  - write wakeup (callback)
- A few additional helpers

Driver Interface Functions

```
int      serdev_device_open(...);
void     serdev_device_close(...);
unsigned serdev_device_set_baudrate(...);
void     serdev_device_set_flow_control(...);
int      serdev_device_write_buf(...);
void     serdev_device_wait_until_sent(...);
int      serdev_device_get_tiocm(...);
int      serdev_device_set_tiocm(...);
void     serdev_device_write_flush(...);
int      serdev_device_write_room(...);
```

- No write serialisation
- No operation ordering enforced

Driver Interface Callbacks

```
struct serdev_device_ops {
        int (*receive_buf)(...);
        void (*write_wakeup)(...);
};
```

Slave Driver Example

```
static struct serdev_device_driver slave_driver = {
    .driver     = {
        .name           = "serdev-slave",
        .of_match_table = slave_of_match,
    },
    .probe  = slave_probe,
    .remove = slave_remove,
};

module_serdev_device_driver(slave_driver);
```

Slave Driver Probe Example

```
static struct serdev_device_ops slave_ops {
    .receive_buf  = slave_receive_buf,
    .write_wakeup = slave_write_wake_up,
};

static int slave_probe(struct serdev_device *serdev)
{
    ...
    priv->serdev = serdev;
    serdev_device_set_drvdata(serdev, priv);
    serdev_device_set_client_ops(serdev, &slave_ops);
    serdev_device_open(serdev);
    serdev_device_set_baudrate(serdev, 115200);
    device_add(&priv->dev);
    return 0;
}
```

Merged Drivers

- Bluetooth
  - hci_serdev (library based on hci_ldisc.c)
  - hci_bcm (4.14)
  - hci_ll (4.12)
  - hci_nokia (4.12)
- Ethernet
  - qca_uart (4.13)

A Word on hci_bcm

- Precursor to serdev
- Hack for additional resources and PM
- Platform companion device
  - Described by ACPI or platform code
  - Child of serial device
  - Manages GPIOs and clocks
  - Registered in driver list at probe
  - Looked up in list from HCI callbacks
    - Matches on parent device
- Serdev ACPI support?
- Regression risk

Limitations

- Serial-core only
- No hotplug
- Device-tree only
- Single slave
- Raw mode only (error flags discarded)

Hotplug

- Implemented in TTY layer using file operations and hangups
- But serdev does not use file abstraction
- Requires changes to TTY layer
- Partial reason for initial revert
- PCI hotplug...
- Description of dynamic buses
  - Only USB has rudimentary support for Device Tree
  - Device tree overlays
  - No in-kernel user-space interface for overlays
- Example
  - Pulse Eight HDMI CEC USB device (ACM, serio driver)

Quirks

- Line-discipline allocated (and used)
- Controller always registered
- No character device (feature)
- No bus PM (`power.ignore_children` ?)
- No operation ordering
- Code duplication and backwards compatibility
- Ungraceful handling of second slave
- Inconsistent naming
  - serdev vs. serial bus
  - serdev device
  - device, client, slave

In the Works

- ACPI support
  - "[RFC 0/3] ACPI serdev support" (September 7)
  - BCM2E39 Bluetooth device (regression?)

In the Works

- ACPI support
  - "[RFC 0/3] ACPI serdev support" (September 7)
  - BCM2E39 Bluetooth device (regression?)
- Mux support
  - "[PATCH 0/6] serdev multiplexing support" (August 16)
  - Utilising new mux subsystem
  - Adds `reg` property (mux index)
  - Has issues (no flushing, and no locking?!)
  - max9260 I2C-controller slave driver
  - Basic parity support (no error handling)

In the Works

- ACPI support
  - "[RFC 0/3] ACPI serdev support" (September 7)
  - BCM2E39 Bluetooth device (regression?)
- Mux support
  - "[PATCH 0/6] serdev multiplexing support" (August 16)
  - Utilising new mux subsystem
  - Adds `reg` property (mux index)
  - Has issues (no flushing, and no locking?!)
  - max9260 I2C-controller slave driver
  - Basic parity support (no error handling)
- RAVE slave driver
  - "[PATCH v7 1/1] platform: Add driver for RAVE Supervisory Processor" (September 6)

In the Works

- ACPI support
  - "[RFC 0/3] ACPI serdev support" (September 7)
  - BCM2E39 Bluetooth device (regression?)
- Mux support
  - "[PATCH 0/6] serdev multiplexing support" (August 16)
  - Utilising new mux subsystem
  - Adds `reg` property (mux index)
  - Has issues (no flushing, and no locking?!)
  - max9260 I2C-controller slave driver
  - Basic parity support (no error handling)
- RAVE slave driver
  - "[PATCH v7 1/1] platform: Add driver for RAVE Supervisory Processor" (September 6)
- w2sg and w2cbw GPS and WiFi/BT slave drivers
  - "[RFC 0/3] misc: new serdev based drivers for w2sg00x4 GPS module and w2cbw003 wifi/bluetooth" (May 21)

Future Work

- Address quirks and limitations, including
  - ACPI
  - Hotplug
  - Enable for more TTY drivers (USB serial)
  - Mux and RS-485?
  - Bus PM?
- Convert line-discipline drivers
  - NFC
  - CAN
  - ti-st driver
  - Some serio drivers (pulse8-cec)?

Further Reading

- include/linux/serdev.h
- drivers/tty/serdev/
  - core.c
  - serdev-ttyport.c
- Documentation/devicetree/bindings/
  - serial/slave-device.txt
  - net/broadcom-bluetooth.txt
  - net/nokia-bluetooth.txt
  - net/qca,qca7000.txt
  - net/ti,wilink-st.txt
- "The need for TTY slave devices" by Neil Brown
  - https://lwn.net/Articles/700489/

Thanks!

johan@hovoldconsulting.com
johan@kernel.org