The Serial Device Bus

Johan Hovold

Hovold Consulting AB

Embedded Linux Conference Europe
October 23, 2017

Introduction

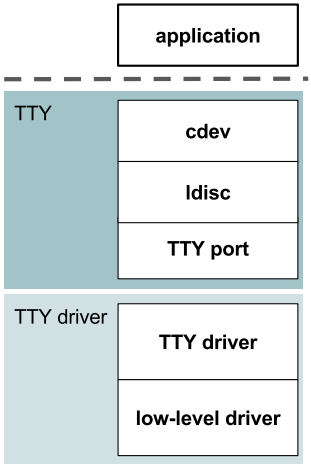
- UARTs and RS-232 have been around since 1960s
- Common interface for Bluetooth, NFC, FM Radio and GPS devices
- TTY layer abstracts serial connection
  - Character-device interface (e.g. `/dev/ttyS1`)
- But no (good) way to model associated resources (e.g. for PM)
  - GPIOs and interrupts
  - Regulators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

Outline

- TTY Layer
- User-space drivers
- Line-discipline drivers
- Serdev implementation
- Serdev driver interface
- Limitations
- Future work

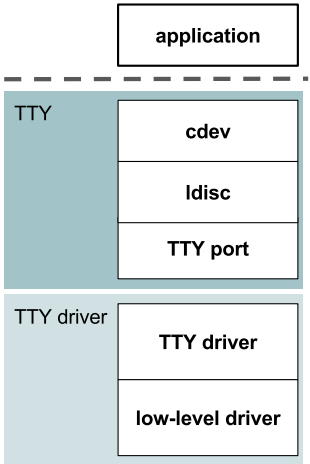
TTY Layer

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



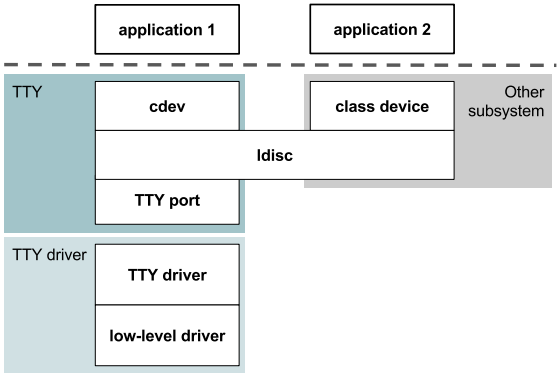
User-Space Drivers

- Using default n_tty line discipline
- Description in user space
  - Port
  - Line speed
- Associated resources?
  - GPIOs and interrupts (accessible)
  - Regulators (N/A)
  - Clocks (N/A)
- Custom power management
  - System-suspend notifications
  - Wakeup interrupts
- Custom firmware management



Line-Discipline Drivers

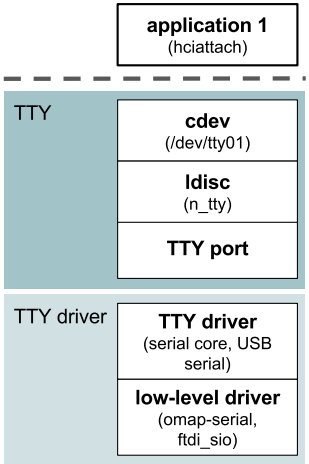
- Interaction with other subsystems (e.g. bluetooth, input, nfc, ppp)
- Ldisc registers further class devices
- Registered while port (ldisc) is open
- User-space daemon to initialise port and switch line discipline
  - `ldattach`
  - `inputattach`
  - `hciattach` (`btattach`)
- Firmware infrastructure available
- But still issues with other resources and PM



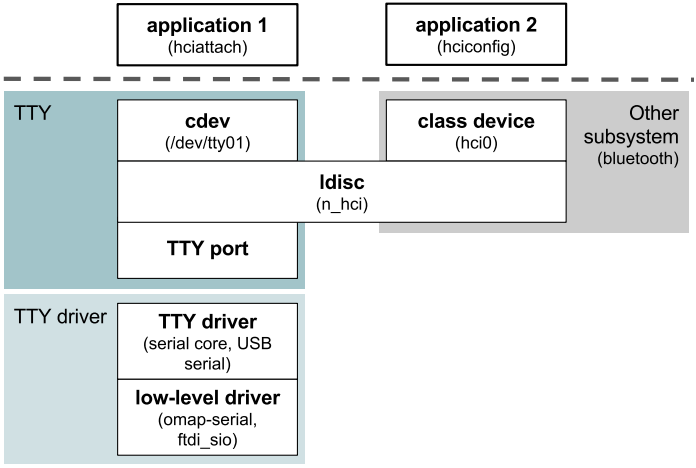
Bluetooth Example

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

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

/* configure line settings */

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

Bluetooth Example



Bluetooth Example



Problems with Line-Discipline 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

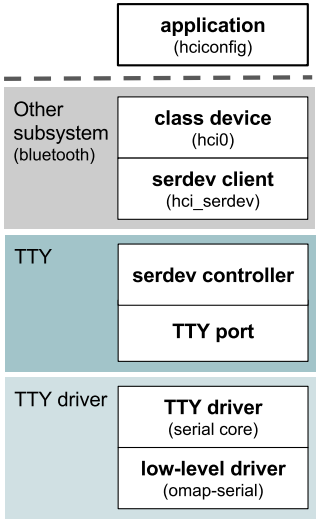
The Serial Device Bus

- 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)

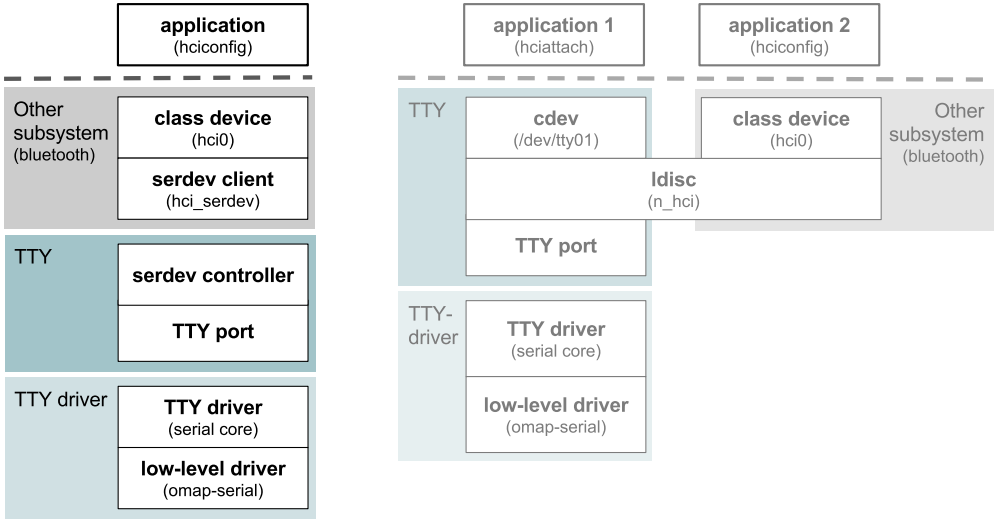
Serdev Overview

- New bus type: `serial`
- Serdev controllers
- Serdev devices (a.k.a. clients or slaves)
- Serdev TTY-port controller
  - Only in-kernel controller implementation
  - Sometimes (incorrectly) identified with Serdev
  - Registered by TTY driver when clients defined
  - Controller replaces TTY character device
- Clients described by firmware (Device Tree or soon ACPI)

Serdev Drivers



Serdev Drivers



TTY-Port Controller Implementation

```
struct device *tty_port_register_device_serdev(...);

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

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

- Registers controller and slaves instead of TTY class device
- Replaces default *TTY-port client* operations
- Controller interface implemented using TTY layer and TTY-driver ops

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
|       `-- tty00
|
`-- 48022000.serial
    |-- driver -> .../omap_uart
    `-- serial0
        `-- serial0-0
            |--bluetooth
            |  `-- hci0
            |-- driver -> .../hci-ti
            `-- subsystem -> .../bus/serial
```

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(struct serdev_device *);
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(...);
void     serdev_device_write_flush(...);
int      serdev_device_write_room(...);
int      serdev_device_get_tiocm(...);
int      serdev_device_set_tiocm(...);
```

- No write serialisation (should not be a problem)
- No operation ordering enforced (by core)
- All but `write_buf()` and `write_room()` may sleep

Driver Interface Callbacks

```
struct serdev_device_ops {
        int (*receive_buf)(struct serdev_device *,
                           const unsigned char *, size_t);
        void (*write_wakeup)(struct serdev_device *);
};
```

- `receive_buf()`
  - Workqueue context
  - Returns number of bytes processed
- `write_wakeup()`
  - Typically atomic context
  - Must not sleep

Example Driver

```
static struct serdev_device_driver slave_driver = {
    .driver = {
        .name           = "serdev-slave",
        .of_match_table = of_match_ptr(slave_of_match),
        .pm             = &slave_pm_ops,
    },
    .probe  = slave_probe,
    .remove = slave_remove,
};

module_serdev_device_driver(slave_driver);
```

Example Driver Probe

```
static struct serdev_device_ops slave_ops;

static int slave_probe(struct serdev_device *serdev)
{
    ...
    priv->clk = clk_get(&serdev->dev, "clk");
    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;
}
```

Limitations

- Serial-core only (for now)
- No hotplug support
- Single slave
- No input flow control
  - No push back
  - Data silently dropped if client can't keep up
- No input processing (cf. raw terminal mode)
  - No software flow control (XON/XOFF)
  - No parity, framing, or overrun errors
  - No break signalling

Serial Port Hotplugging

- Implemented using TTY hangups and file operations
- 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
  - Pass overlays (`compatible` strings) from TTY drivers?
- Example
  - Pulse Eight HDMI CEC USB device (ACM, serio driver)

Quirks

- Line-discipline allocated (and used)
- Controller always registered
- No character device (feature)
- No operation ordering
- No controller runtime PM (client-device status not propagated)
- Code duplication and backwards compatibility
- Some naming inconsistencies
  - *serial* bus (not *serdev*)
  - *serdev device/client/slave*

Kconfig Notice

```
Device drivers --->
    Character devices --->
        <*> Serial device bus --->
            <*> Serial device TTY port controller
```

- SERIAL_DEV_BUS [=y]
  - Tristate
  - Driver dependency
- SERIAL_DEV_CTRL_TTYPORT [=y]
  - Boolean
  - Only in-kernel controller implementation
  - Should default to y (patch posted)
  - Depends on TTY and SERIAL_DEV_BUS != m

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 and PM support merged for 4.15
- Regression risk
- Similar problems with hci_intel

In the Works

- ACPI support merged for 4.15
  - "[PATCH v3 0/2] ACPI serdev support" (October 11)
  - Potential hci_bcm and hci_intel breakage

In the Works

- ACPI support merged for 4.15
  - "[PATCH v3 0/2] ACPI serdev support" (October 11)
  - Potential hci_bcm and hci_intel breakage
- 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 merged for 4.15
  - "[PATCH v3 0/2] ACPI serdev support" (October 11)
  - Potential hci_bcm and hci_intel breakage
- 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 v8 0/5] ZII RAVE platform driver" (October 18)
  - MFD driver for supervisory processor (watchdog, backlight, LED, etc.)

In the Works

- ACPI support merged for 4.15
  - "[PATCH v3 0/2] ACPI serdev support" (October 11)
  - Potential hci_bcm and hci_intel breakage
- 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 v8 0/5] ZII RAVE platform driver" (October 18)
  - MFD driver for supervisory processor (watchdog, backlight, LED, etc.)
- 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
  - Add hotplug support
  - Enable for more TTY drivers (USB serial)
  - Multiple slaves (mux and RS-485)?
- Further Bluetooth protocol drivers (e.g. hci_intel)
- Other line-discipline drivers
  - NFC
  - CAN
  - ti-st driver
  - Some serio drivers (e.g. 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