The Serial Device Bus

Johan Hovold

Hovold Consulting AB

Linux Piter, Saint Petersburg
November 2, 2018

Introduction

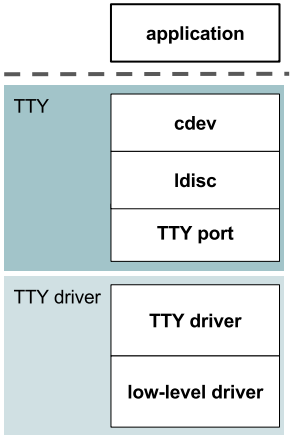
- UARTs and RS-232 have been around since 1960s
- Common interface for Bluetooth, NFC, FM Radio and GPS devices
- TTY layer abstracts serial connection
  - Character-device interface (e.g. `/dev/ttyS1`)
- But no (good) way to model associated resources (e.g. for PM)
  - GPIOs and interrupts
  - Regulators
  - 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

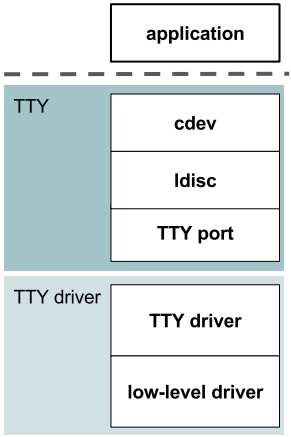
TTY layer

- 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`



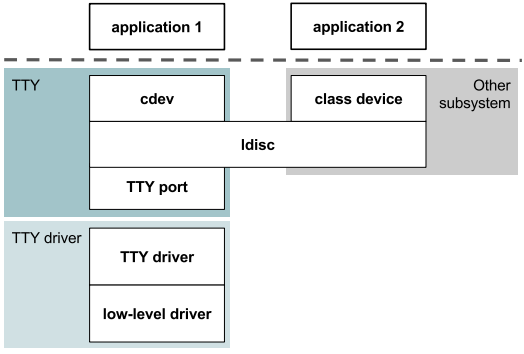
User-space drivers

- 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

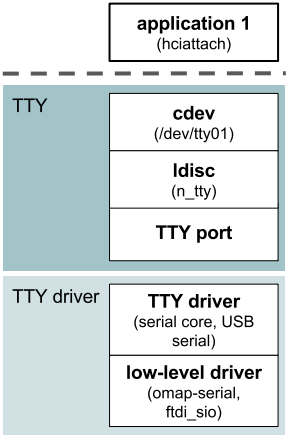
- Interaction with other subsystems (e.g. bluetooth, input, nfc, ppp)
- Ldisc registers further class devices
- Registered while port (ldisc) is open
- User-space daemon to initialise port and switch line discipline
  - `ldattach`
  - `inputattach`
  - `hciattach` (`btattach`)
- Firmware infrastructure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



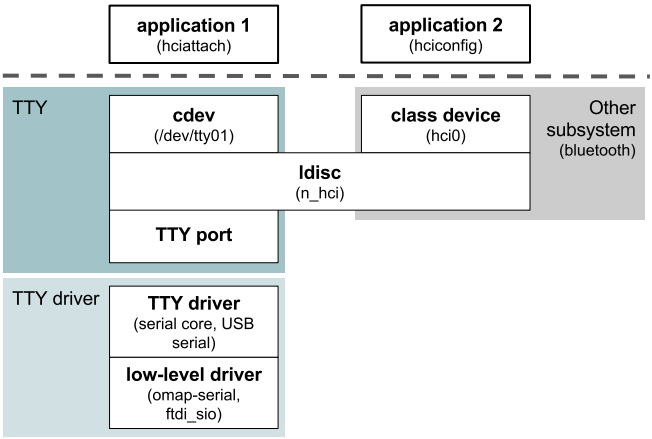
Bluetooth example

```
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 (Devicetree, 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

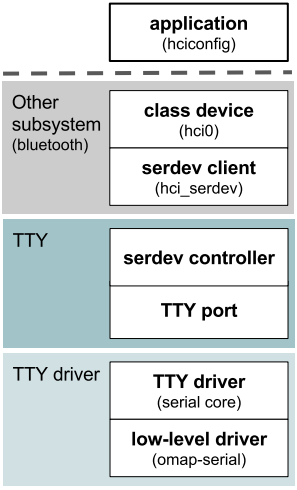
The Serial device bus

- The Serial device bus (Serdev)
- By Rob Herring (Linaro)
- Bus for UART-attached devices
  - Replace Bluetooth line disciplines and hciattach
  - Earlier efforts (GPS power management)
- Merged in 4.11
- Enabled for serial core only in 4.12 (due to lifetime issues)

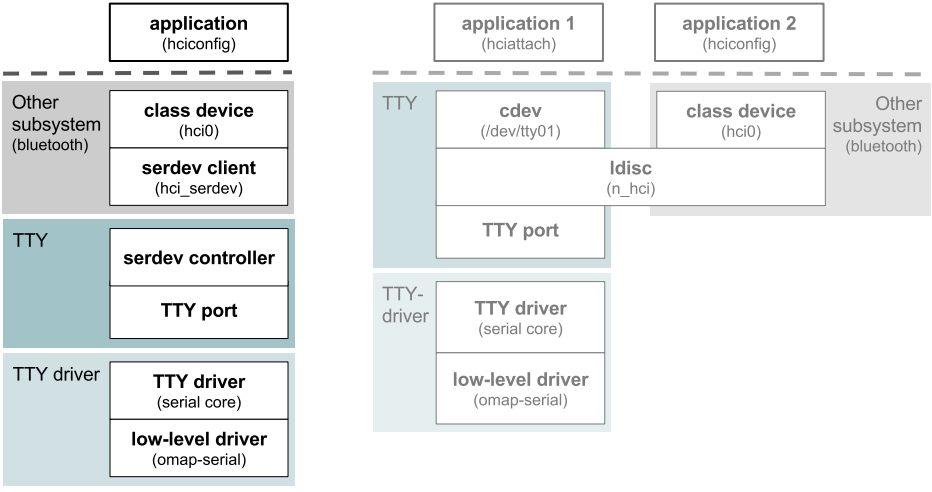
Serdev overview

- 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 (Devicetree or 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

Devicetree bindings

- Child node of serial-port node
- Generic properties
  - `compatible` (required)
  - `max-speed` (optional)
  - `current-speed` (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_set_parity(...);
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),
        .acpi_match_table = ACPI_PTR(slave_acpi_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;
}
```

Power management

- Controller runtime power management (RPM) added in 4.19
- Client RPM status not propagated
  - `power.ignore_children = true`
- Controller generally active while open
- Allows for aggressive controller RPM
  - Driver can put RPM reference taken by core at open()
  - Resume controller when expecting input (e.g. based on OOB signalling or protocol)
- Only implemented by omap-serial and some USB-serial drivers
  - OMAP currently broken (controller always active)
  - Need generic user-space interface to enable (PM QoS?)

Limitations

- Serial-core only (for now)
- No hotplug support
- Single slave
- No input flow control
  - No pushback
  - 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 support for Devicetree (4.16)
  - Devicetree 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
- 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
  - Defaults to y (since 4.15)
  - Depends on TTY and SERIAL_DEV_BUS != m

Merged drivers

- Bluetooth
  - Broadcom (4.14, ACPI in 4.15)
  - Mediatek (4.19)
  - Nokia (4.12)
  - Qualcomm Atheros (4.18)
  - Realtek (3-wire, ACPI, 4.19)
  - Texas Instruments (4.12)
- Ethernet
  - Qualcomm Atheros (4.13)
- GNSS
  - SiRFStar (4.19)
  - u-blox (4.19)
- MFD
  - RAVE supervisory processor (4.16)

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 in 4.15
- Regression risk
- Similar hacks in hci_intel, which is now broken

In the works

- NXP PN533 NFC driver (v3, October 25)
  - UART-interface support for existing driver

In the works

- NXP PN533 NFC driver (v3, October 25)
  - UART-interface support for existing driver
- LoRa socket API (RFC, July 1)
  - Long-range, low-power wireless technology
  - SPI or UART interfaces, currently handled in user space

In the works

- NXP PN533 NFC driver (v3, October 25)
  - UART-interface support for existing driver
- LoRa socket API (RFC, July 1)
  - Long-range, low-power wireless technology
  - SPI or UART interfaces, currently handled in user space
- Dynamic descriptions through sysfs? (v2, June 11)
  - More of an RFC
  - Interface and implementation issues

In the works

- NXP PN533 NFC driver (v3, October 25)
  - UART-interface support for existing driver
- LoRa socket API (RFC, July 1)
  - Long-range, low-power wireless technology
  - SPI or UART interfaces, currently handled in user space
- Dynamic descriptions through sysfs? (v2, June 11)
  - More of an RFC
  - Interface and implementation issues
- USB-serial Devicetree and Serdev support (RFC, May 25)
  - Depends on Serdev hotplug support
  - Usable for static topologies

In the works (cont.)

- ACPI regression in 4.15 (April 24)
  - Serdev claims serial port even when no driver
  - Need to black- or whitelist slave devices
  - Still not fixed, new bug report last week (October 25)

In the works (cont.)

- ACPI regression in 4.15 (April 24)
  - Serdev claims serial port even when no driver
  - Need to black- or whitelist slave devices
  - Still not fixed, new bug report last week (October 25)
- Dell AIO backlight driver (v1, October 26, 2017)
  - File I/O from kernel
  - Serdev rewrite pending

In the works (cont.)

- ACPI regression in 4.15 (April 24)
  - Serdev claims serial port even when no driver
  - Need to black- or whitelist slave devices
  - Still not fixed, new bug report last week (October 25)
- Dell AIO backlight driver (v1, October 26, 2017)
  - File I/O from kernel
  - Serdev rewrite pending
- Mux support? (v1, August 16, 2017)
  - Utilising new mux subsystem
  - Adds `reg` property (mux index)
  - Has issues (no flushing, no locking)

In the works (cont.)

- ACPI regression in 4.15 (April 24)
  - Serdev claims serial port even when no driver
  - Need to black- or whitelist slave devices
  - Still not fixed, new bug report last week (October 25)
- Dell AIO backlight driver (v1, October 26, 2017)
  - File I/O from kernel
  - Serdev rewrite pending
- Mux support? (v1, August 16, 2017)
  - Utilising new mux subsystem
  - Adds `reg` property (mux index)
  - Has issues (no flushing, no locking)
- max9260 I2C-controller driver (v1, August 16, 2017)

Future work

- Address quirks and limitations, including
  - Add hotplug support
  - Enable for more TTY drivers (usbserial)
  - 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