Remote Controller devices

Remote Controller core

The remote controller core implements infrastructure to receive and send remote controller keyboard keystrokes and mouse events.

Every time a key is pressed on a remote controller, a scan code is produced. Also, on most hardware, keeping a key pressed for more than a few dozens of milliseconds produce a repeat key event. That's somewhat similar to what a normal keyboard or mouse is handled internally on Linux[1]. So, the remote controller core is implemented on the top of the linux input/evdev interface.

[1] The main difference is that, on keyboard events, the keyboard controller produces one event for a key press and another one for key release. On infrared-based remote controllers, there's no key release event. Instead, an extra code is produced to indicate key repeats.

However, most of the remote controllers use infrared (IR) to transmit signals. As there are several protocols used to modulate infrared signals, one important part of the core is dedicated to adjust the driver and the core system to support the infrared protocol used by the emitter.

The infrared transmission is done by blinking a infrared emitter using a carrier. The carrier can be switched on or off by the IR transmitter hardware. When the carrier is switched on, it is called *PULSE*. When the carrier is switched off, it is called *SPACE*.

In other words, a typical IR transmission can be viewed as a sequence of *PULSE* and *SPACE* events, each with a given duration.

The carrier parameters (frequency, duty cycle) and the intervals for *PULSE* and *SPACE* events depend on the protocol. For example, the NEC protocol uses a carrier of 38kHz, and transmissions start with a 9ms *PULSE* and a 4.5ms SPACE. It then transmits 16 bits of scan code, being 8 bits for address (usually it is a fixed number for a given remote controller), followed by 8 bits of code. A bit "1" is modulated with 560Âµs *PULSE* followed by 1690Âµs *SPACE* and a bit "0" is modulated with 560Âµs *PULSE* followed by 560Âµs *SPACE*.

At receiver, a simple low-pass filter can be used to convert the received signal in a sequence of *PULSE/SPACE* events, filtering out the carrier frequency. Due to that, the receiver doesn't care about the carrier's actual frequency parameters: all it has to do is to measure the amount of time it receives *PULSE/SPACE* events. So, a simple IR receiver hardware will just provide a sequence of timings for those events to the Kernel. The drivers for hardware with such kind of receivers are identified by `RC_DRIVER_IR_RAW`, as defined by :c:type:`rc_driver_type`[2]. Other hardware come with a microcontroller that decode the *PULSE/SPACE* sequence and return scan codes to the Kernel. Such kind of receivers are identified by `RC_DRIVER_SCANCODE`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]rc-core.rst, line 48); *backlink***
>
> Unknown interpreted text role "c:type".

[2] The RC core also supports devices that have just IR emitters, without any receivers. Right now, all such devices work only in raw TX mode. Such kind of hardware is identified as `RC_DRIVER_IR_RAW_TX`.

When the RC core receives events produced by `RC_DRIVER_IR_RAW` IR receivers, it needs to decode the IR protocol, in order to obtain the corresponding scan code. The protocols supported by the RC core are defined at enum :c:type:`rc_proto`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]rc-core.rst, line 68); *backlink***
>
> Unknown interpreted text role "c:type".

When the RC code receives a scan code (either directly, by a driver of the type `RC_DRIVER_SCANCODE`, or via its IR decoders), it needs to convert into a Linux input event code. This is done via a mapping table.

The Kernel has support for mapping tables available on most media devices. It also supports loading a table in runtime, via some sysfs nodes. See the :ref:`RC userspace API <Remote_controllers_Intro>` for more details.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]rc-core.rst, line 78); *backlink***
>
> Unknown interpreted text role "ref".

Remote controller data structures and functions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]rc-core.rst, line 86)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/media/rc-core.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]rc-core.rst, line 88)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/media/rc-map.h
```