SAMSUNG
Open Source Group

Rapid SPI Device Driver Development over USB

FOSDEM

2018-02-04, Brussels

Stefan Schmidt
stefan@osg.samsung.com
Samsung Open Source Group

SAMSUNG
Open Source Group

Who am I

- FOSS Contributor since 2006
- EFL Developer and Release Manager
- Linux Kernel ieee802154 subsystem maintainer
- Team lead Samsung Open Source Group SRUK

Agenda

- Problem Statement
- Hardware Selection
- MCP2210
- Rapid Test Cycle

Problem Statement

- I maintain a small Linux Kernel subsystem (IEEE 802.15.4)
- A typical hardware transceiver for this subsystem uses SPI for data and some extra GPIOs for signalling
- Getting them all hooked up to a Raspberry Pi for driver development and testing can be annoying:
  - Problems to build a working mainline kernel for the Pi
  - Cross compilation needed
  - Flashing kernel and modules, reboot, test cycle is time consuming

Dream Solution

- Small PCB with flexibility to connect SPI as well as GPIOs over USB
- Connect to my workstation or laptop while travelling
- SPI master driver inside the mainline kernel
- Quickly test mainline kernels without reboot and chance of system screw up

- No embedded board, no cross-compile, no non-mainline kernels
- No user-space SPI library
- No USB HID device
- No SPIDEV

Hardware Selection

- Various USB-SPI bridge chips available
- For my use case a basic feature set is enough
  - SPI interface & some extra GPIO pins
  - Public datasheet must be available
  - A cheap board must be available (<30€)
  - An existing driver would be nice-to-have
  - Maybe some EEPROM to store the config
- If you have a need for SPI at its performance limits some chips might perform better than the one I selected

SAMSUNG
Open Source Group

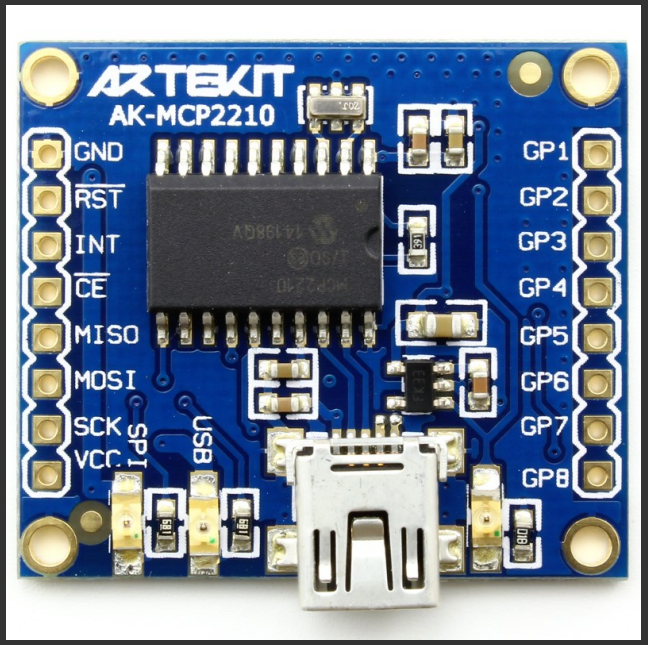
Hardware

Chip	Public Datasheet	Mainline Driver*	Out-of-tree Driver*	Cheap Board
Microchip MCP2210	✔	✘	✔	✔
Silicon Labs CP2130	✔	✘	✔	✘ Board from Silicon Labs
FTDIXXX	✔	✘	✘	✔
Cypress CY7C65211	✘	✘	✘	✔

* A driver with the functionality I am looking for. Some of these devices already have drivers (like USBHID) in mainline.

MCP2210

- Data sheet available
- Cheap (12€) and small board available (only annoying part is the mini USB socket instead of micro USB or USB-C)
- Two drivers available on GitHub
- The one from Daniel Santos has progressed quite far, but sadly never made it into mainline



MCP2210 - USB

- Linux uses the default USB ID for MCP2210 with the USB HID driver (0x04d8:0x00de)
- One can work around this with a udev rule to unbind the driver
- Another solution is to assign a different set of USB PID/VID to the MCP2210
- Stored in the on-chip EEPROM and thus not lost during resets

MCP2210 - SPI

- Connect to the Kernel SPI subsystem
- Setup a SPI master driver and register it
- spi_alloc_master()
- spi_register_master()

- Setup struct spi_board_info and register new device
- spi_new_device()

SAMSUNG
Open Source Group

MCP2210 - Configuration

- What configuration from user-space should be possible?
- Only provisioning or also run-time configuration?
- Leaning towards no runtime configuration
  - Simplification
  - No user-space configuration interface (ioctl’s, sysfs)
  - Hard to actually come up with run-time use cases

Provisioning

- The SPI bus gets not auto-probed so we need a mechanism to let the kernel know what device is connected and how
- struct spi_board_info works for a static configuration
- To be more dynamic and avoid re-compiles of the module another solution is needed
- Devicetree used in some scenarios (not the case on a x86_64 laptop)
- Simply put the dts into the EEPROM of the MCP2210 and let the device driver inform the rest of the kernel?

MCP2210 - Mainlining

- With almost 9000 lines of code the out of tree driver is huge for such a simple piece of hardware
- Work towards mainline needs simplification
  - Remove extra debug infrastructure and use the one from mainline
  - Remove out-of-tree version checks
  - Remove extra ioctl and syfs interfaces
  - To many build options (e.g. without SPI or GPIO)

Rapid Test cycle

- “Virtme is a set of simple tools to run a virtualized Linux kernel that uses the host Linux distribution or a simple rootfs instead of a whole disk image.” --Virtme README

- Python scripts around Qemu/KVM
- Boot installed kernel or non-modular kernel image directly
- No need to have a full disk image, host system is used read-only

Rapid Test cycle

- My work flow:
- Boot a non-modular kernel with -kimg
- Access the USB device with Qemu USB pass-through
- Use my normal workdir for running test scripts and a shared read-write folders if needed

```
virtme-run --pwd --kimg arch/x86_64/boot/bzImage \
–-qemu-opts -usb -usbdevice host:XXXX:XXXX
```

SAMSUNG
Open Source Group

Status Summary

- Board decided: MCP2210 ✔
- Rapid test cycle: Virtme ✔
- Cheap board: Artekit ✔
- Out of tree drivers detects the board ✔
- Hookup 15.4 transceiver for testing ✘ (partly)
- Simplify driver ✘
- Provisioning solution ✘
- Patchset for mainline ✘

SAMSUNG
Open Source Group

Thanks!

Questions?

References

- https://www.silabs.com/products/interface/usb-bridges/classic-usb-bridges/device.cp2130
- https://github.com/Henneberg-Systemdesign/cp2130
- MCP210
- http://www.microchip.com/wwwproducts/en/en556614
- https://github.com/daniel-santos/mcp2210-linux
- https://www.artekit.eu/products/accessories/