EMBEDDED DEVICE DRIVERS

Linux Device Drivers on Beaglebone Black

Linux device model

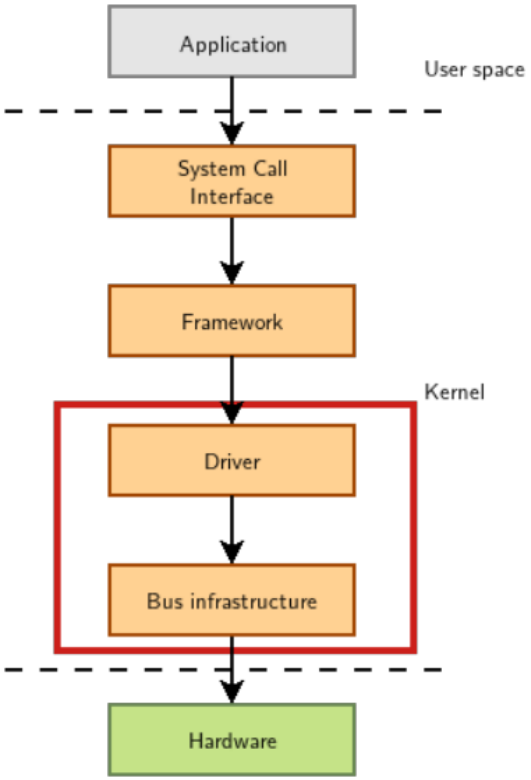
- Linux kernel needs to handle various device connect scenarios
  - The same device needs to use the same driver
  - On multiple CPU architectures
  - But the controllers may be different

  - A single driver
  - Needs to support multiple devices
  - Of the same kind

- This necessitates a separation
  - Controller drivers
    - Control behavior of controllers for complex protocol busses
  - Device drivers
    - Control behavior of specific device instantiations on the bus

Linux device model illustrated

- In Linux, a (device) driver talks to
  - A framework
    - That allows the driver to expose
    - Hardware in a generic manner
  - A bus infrastructure
    - To detect and communicate
    - With the actual hardware



Device model data structures

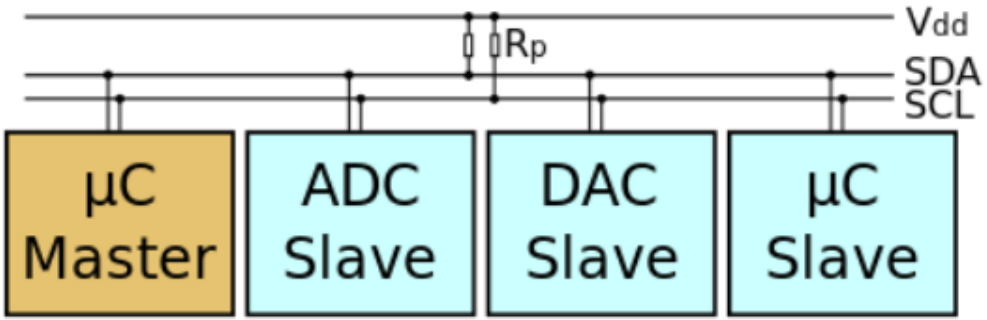
- The device model is organized around:
  - ***struct bus_type***
    - Represents a bus (USB, I2C, SPI, etc.)
  - ***struct device_driver***
    - Represents a driver capable of handling
      - Certain devices on a certain bus
  - ***struct device***
    - Represents one device connected to a bus
- The kernel uses inheritance
  - To create more specialized versions
    - For each bus subsystem

I2C: What?

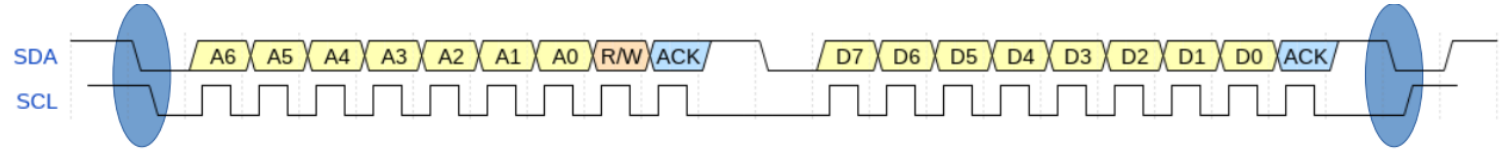
- Serial protocol
  - Roles: Master, Slave
  - 2-wires – SCL (clock), SDA (Data)
    - Synchronous, master-driven clock
  - Multi-drop, multi-master
  - Addressing
    - Masters do not have addresses
    - Slaves have 7-bit / 10-bit addressing
  - Speeds:
    - Original: 100kHz
    - Version 1: 400 kHz Fast
    - Version 2: 3.4MHz High speed
    - Version 3:1MHz Fast+ mode
    - Version 4: 5MHz Ultra fast
  - Connects low-speed controllers, EEPROMs, ADCs, DACs, …

I2C: Wiring

- Wired-OR logic based
- Half-duplex protocol
- Signal wires need pull-up resistors
  - Open-collector / open-drain
  - Pulled up to V_{DD}



I2C: Signalling



Start
SDA goes low before SCL to signal the start of transmission.

Addr
7 bit address that determines the slave device to be accessed.

R/W
Transaction data direction bit. (1 = read, 0 = write)

Data
Byte data read from or written to the slave device. Can be multiple bytes.

ACK
Acknowledge bit. (0 = ack, 1 = nak)

Stop
SDA goes high after SCL to signal the end of transmission.

LKM: I2C subsystem in kernel

- Concepts and kernel perspectives
  - Adapter
    - A master chip is a node that initiates communication
    - Also called ‘adapter’ in the kernel lingo
    - Adapter drivers are in ***drivers/i2c/busses/*** directory
  - Algorithm
    - Code used to implement a class of adapters
    - Each adapter driver depends on one of the algos in the ***drivers/i2c/algos/*** directory
  - Client
    - A slave chip is a node that responds to master commands
    - Also called ‘client’ in the kernel lingo
    - Client drivers are in ***drivers/xx/yy*** for **xx** interface
      - Example:
        - *drivers/misc/eeprom/at24.c*
        - *drivers/media/i2c*

LKM: I2C client – device (1/2)

- Include header:
  #include <linux/i2c.h>
- Get i2c adapter per bus number ***nr***
  *struct i2c_adapter ****i2c_get_adapter****(int nr);*
- Create ***i2c_device_id***
  - Export it using ***MODULE_DEVICE_TABLE***
- Create ***i2c_board_info*** using the slave name and address
  I2C_BOARD_INFO*(SLAVE_NAME, SLAVE_ADDR);*
- Create and register ***i2c_client***
  *struct i2c_client ****i2c_new_client_device****(struct i2c_adapter *adapter, struct i2c_board_info *)*

LKM: I2C client – device (2/2)

- Create an i2c_driver
  *struct i2c_driver **my_i2c_driver**;*
- Register driver with I2C subsystem
  ***i2c_add_driver**(my_i2c_driver);*
  - This automatically runs the ***probe()*** and sets up the device
- Data transfers from/to slave device use the i2c master APIs
  *int **i2c_master_send**(const struct i2c_client, *client, const char *buf, int count);*
  *int **i2c_master_recv**(const struct i2c_client *client, char *buf, int count);*
- When exiting, unregister the client and delete the driver
  *void **i2c_unregister_device**(struct i2c_client *client);*
  *void **i2c_del_driver**(struct i2c_driver *driver);*
- Kernel reference:
  - https://www.kernel.org/doc/html/latest/i2c/writing-clients.html

LKM: I2C adapter (bus/master)

- When *i2c_driver* calls actual transfer functions
  - *i2c_master_send() / i2c_master_recv()*
- It is the *i2c_adapter* (bus driver)
  - Which handles these requests
- Bus driver structures
  - *struct **i2c_algorithm***
    - Represents the actual transfer methods

      *int (*__master_xfer__)(struct i2c_adapter *, struct i2c_msg *, int);*

      *int (*__smbus_xfer__)(struct i2c_adapter *, u16, unsigned short, char, u8, int, unions i2c_smbus_data *);*
  - *struct **i2c_adapter***
    - Represents the actual i2c bus *(/dev/i2c-0*, */dev/i2c-1*, …)

LKM: I2C Adapter (usage)

- Creation:
  - *int **i2c_add_adapter**(struct i2c_adapter *adapter);*
  - *int **i2c_add_numbered_adapter**(struct i2c_adapter *adapter);*
    - ***adapter->nr*** contains the bus number
- Deletion
  - *void **i2c_del_adapter**(struct i2c_adapter *adapter);*

I2C: Userspace access control

- ***i2c-tools***
  - Package for user space interaction
  - With i2c devices connected on a bus

- Commands
  - *i2cdetect*
    - Detect i2c devices responding on a bus
  - *i2cset*
    - Write data to a location in i2c slave (address) on a bus
  - *i2cget*
    - Read data from a location in i2c slave on a bus
  - *i2cdump*
    - Dump data from range of addresses in i2c slave on a bus

I2C: i2cdetect

- Usage:
  - Get buses on a system

```
root@BeagleBone:/home/debian# i2cdetect -l
i2c-1   i2c             OMAP I2C adapter                        I2C adapter
i2c-2   i2c             OMAP I2C adapter                        I2C adapter
i2c-0   i2c             OMAP I2C adapter                        I2C adapter
```

  - Detect devices on bus 0 – who are these?

```
root@BeagleBone:/home/debian# i2cdetect -y 0
Warning: Can't use SMBus Quick Write command, will skip some addresses
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:
10:
20:
30: -- -- -- -- 34 -- -- --
40:
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:
70:
```

I2C: i2c-stub

- Linux kernel has a kernel module
  - i2c-stub
    - To create a dummy slave **SMBUS** device
    - On the i2c bus
- Used for
  - Learning i2c device behaviour
    - Specially i2c-tools usage
  - Testing i2c device drivers (client) code
- Not compiled by default for BBB kernel
  - So we will compile it as a module and load it at runtime
  - File: ***<linux src>/drivers/i2c/i2c-stub.c***
    - Also copied in **mod13** directory for reference

LKM: I2C exercise #1 (1/2)

- Refer ***mod13*** directory
  - Compile ***i2c-stub.c*** and load the module on BBB

    # insmod i2c-stub.ko chip_addr=0x25
    - It creates a new i2c-bus
      - And populates it with the slave device with address 0x25
    - Verify this from ***dmesg*** output
    - Also, check using ***i2cdetect –l*** and **i2cdetect –y 3**

```
root@BeagleBone:/home/debian# insmod i2c-stub.ko chip_addr=0x25
root@BeagleBone:/home/debian# i2cdetect -l
i2c-3   smbus           SMBus stub driver                       SMBus adapter
i2c-1   i2c             OMAP I2C adapter                        I2C adapter
i2c-2   i2c             OMAP I2C adapter                        I2C adapter
i2c-0   i2c             OMAP I2C adapter                        I2C adapter
root@BeagleBone:/home/debian# i2cdetect -y 3
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- 25 -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

LKM: I2C exercise #1 (2/2)

- Use i2c-tools utils to write, read and dump data from this slave device

```
root@BeagleBone:/home/debian# i2cdump -y -r 0x0-0x10 -a 3 0x25
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
10: 00                                                 .
root@BeagleBone:/home/debian# i2cset -y -a 3 0x25 0x00 0x55
root@BeagleBone:/home/debian# i2cget -y -a 3 0x25 0x00
0x55
root@BeagleBone:/home/debian# i2cdump -y -r 0x0-0x10 -a 3 0x25
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    U...............
10: 00                                                 .
```

LKM: I2C exercise #2

- We will now access the slave device
  - Using a kernel module – client-driver
- Refer ***mod13.c*** which contains the driver code
  - Notice the driver code steps
    - To get hold of the underlying i2c_adapter
    - To create the i2c_client
    - And the i2c_driver
    - On addition of which the probe() is called
      - Which in turn does an SMBUS read (not i2C read)
        - From the slave device
  - Compile and load the module on BBB
    - Observe the ***dmesg*** output
- Unload the modules
  - First mod13.ko then i2c-stub.ko!

THANK YOU!