Linux I2C

Kaiden Yu

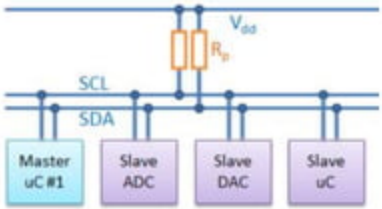
Overview

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

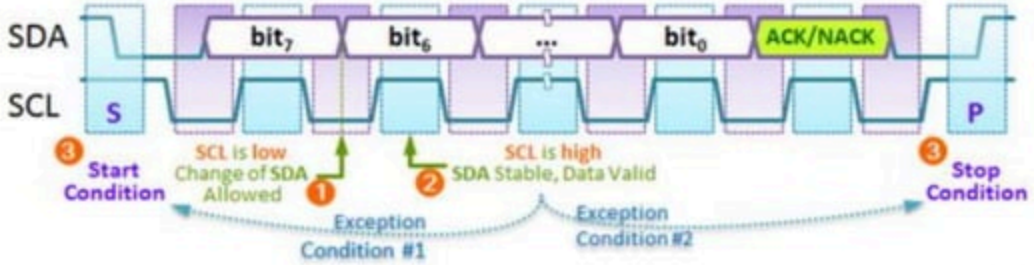
Introduction to I2C – Basics

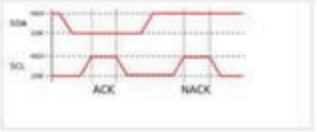
- I2C(Inter-Intergradted Circuit Bus):published by Philip，with licensed trademark and patent (not any more)
- inter-chip communication
- wires
  - SCL(Serial Clock Line)
  - SDA(Serial Data Line)
- with pull-up resistors
- master/slave hierarchy
- bus speed
  - 100KHz/ 400KHz/ 1MHz/ 3.4MHz/ 5MHz



Introduction to I2C – Basics

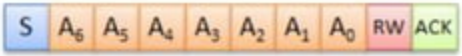
- Signal





Introduction to I2C – Basics

- Address
  - 7 bits
  - 10 bits



Introduction to I2C – Basics

- **TWI**(Two-Wire Interface)：introduced by Atmel to avoid conflicts with trademark issues related to I^2C

- **SMBus**(System Management Bus)：defined by Intel
  - with tighter constraints (100KHz, timeout, format)
  - I2C adapters can support most SMBus protocol, but not the other way around

- **SCCB**(Serial Camera Control Bus)：developed by OmniVision
  - three kind of transaction
    - 3 Phase write
    - 2 Phase read
    - 2 Phase write
  - Don't care bit

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Introduction to I2C – I2C/SMBus Protocol

- I2C protocol
  - S (1 bit) : Start bit
    - Sr (1 bit) : Repeated start bit
  - Addr (7 bits): I2C 7 bit address
  - Rd/Wr (1 bit) : Read/Write bit
  - A, NA (1 bit) : ACK/NACK bit
  - Data (8 bits)
  - P (1 bit) : Stop bit

Introduction to I2C – I2C/SMBus Protocol

- Simple send transaction
  - S Addr Wr [A] Data [A] Data [A] … [A] Data [A] P
- Simple receive transaction
  - S Addr Rd [A] [Data] A [Data] A … A [Data] NA P
- Combined transactions
  - S Addr Wr [A] Data [A] Sr Addr Rd [A] [Data] NA P
  - …

Introduction to I2C – I2C/SMBus Protocol

- SMBus protocol
  - S (1 bit) : Start bit
    - Sr (1 bit) : Repeated start bit
  - Addr (7 bits): I2C 7 bit address
  - Rd/Wr (1 bit) : Read/Write bit.
  - A, NA (1 bit) : ACK/NACK bit
  - Comm (8 bits): Command byte, a data byte which often selects a register on the device.
  - Data (8 bits): A plain data byte
    - Count (8 bits): A data byte containing the length of a block operation.
  - P (1 bit) : Stop bit

Introduction to I2C – I2C/SMBus Protocol

- SMBus Quick Command
  - S Addr Rd/Wr [A] P
- SMBus Receive Byte
  - S Addr Rd [A] [Data] NA P
- SMBus Send Byte
  - S Addr Wr [A] Data [A] P
- SMBus Read Byte
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [Data] NA P
- SMBus Read Word
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [DataLow] A [DataHigh] NA P
- SMBus Write Byte
  - S Addr Wr [A] Comm [A] Data [A] P
- SMBus Write Word
  - S Addr Wr [A] Comm [A] DataLow [A] DataHigh [A] P
- SMBus Process Call
  - S Addr Wr [A] Comm [A] DataLow [A] DataHigh [A] Sr Addr Rd [A] [DataLow] A [DataHigh] NA P
- SMBus Block Read
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [Count] A [Data] A [Data] A … A [Data] NA P
  - reads a block of up to 32 bytes from a device
- SMBus Block Write
  - S Addr Wr [A] Comm [A] Count [A] Data [A] Data [A] … [A] Data [A] P
- SMBus Block Write - Block Read Process Call
  - S Addr Wr [A] Comm [A] Count [A] Data [A] … Sr Addr Rd [A] [Count] A [Data] … A P
  - sned then read 31 bytes
- I2C Block Read
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [Data] A [Data] A … A [Data] NA P
- I2C Block Write
  - S Addr Wr [A] Comm [A] Data [A] Data [A] … [A] Data [A] P

Introduction to I2C – I2C/SMBus Protocol

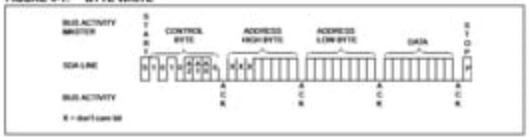
- Check the datasheet of the slave device for its protocol

Single READ from Random Location

Figure 34 shows the typical READ cycle of the host to the [redacted]. The first two bytes sent by the host are an internal 16-bit register address. The following 2-byte READ cycle sends the contents of the registers to host.

Figure 34: Single READ from Random Location

FIGURE 4-1: BYTE WRITE



Single READ from Current Location

Figure 35 shows the single READ cycle without writing the address. The internal address will use the previous address value written to the register.

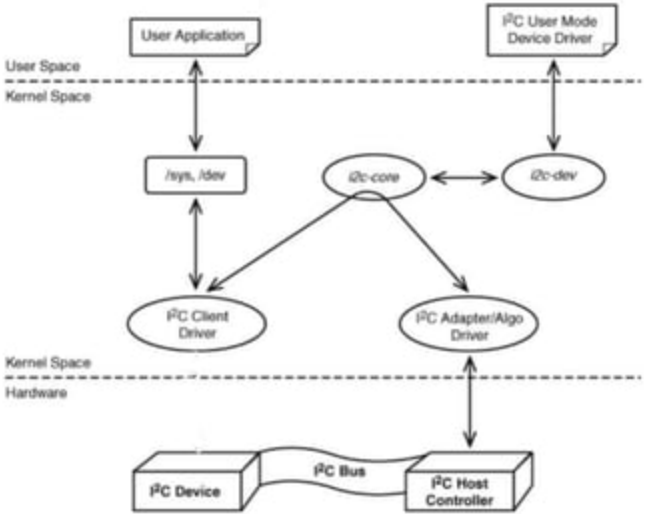
Figure 35: Single Read from Current Location

Read from one control register.

- Start signal
- Slave address byte ($R/\overline{W}$ bit – low)
- Base address byte
- Start signal
- Slave address byte ($R/\overline{W}$ bit – high)
- Data byte from base address
- Stop signal

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – Linux I2C Subsystem



- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – I2C Bus Driver

- The bus driver：i2c-core (i2c-core-*.c)、i2c.h
  - creates and registers the bus_type structure -> i2c_bus_type
  - provides API to
    - register and implement I2C adapter drivers
    - register and implement I2C device drivers
    - matches the device drivers against the devieces detected by the adapter driver
  - define driver and device specific structures
    - struct i2c_adapter
    - struct i2c_algorithm
    - struct i2c_client
    - struct i2c_driver
    - …

Linux I2C – I2C Bus Driver

- API for I2C transaction
  - i2c_master_send()
    - i2c_transfer_buffer_flags()
      - i2c_transfer()
  - i2c_master_recv()
    - i2c_transfer_buffer_flags()
      - i2c_transfer()
  - i2c_transfer()
    - __i2c_transfer()
      - adap->algo->master_xfer()

- Simple send transaction
  - S Addr Wr [A] Data [A] Data [A] … [A] Data [A] P
- Simple receive transaction
  - S Addr Rd [A] [Data] A [Data] A … A [Data] NA P
- Combined transactions
  - S Addr Wr [A] Data [A] Sr Addr Rd [A] [Data] NA P
  - …

Linux I2C – I2C Bus Driver

- API for Smbus transaction – PartI
  - i2c_smbus_read_byte()
  - i2c_smbus_write_byte()
  - i2c_smbus_read_byte_data()
  - i2c_smbus_read_word_data()
  - i2c_smbus_write_byte_data()
  - i2c_smbus_write_word_data()

call

- i2c_smbus_xfer()
  - adapter->algo->smbus_xfer()
  - i2c_smbus_xfer_emulated()
    - i2c_transfer()

- SMBus Quick Command
  - S Addr Rd/Wr [A] P
- SMBus Receive Byte
  - S Addr Rd [A] [Data] NA P
- SMBus Send Byte
  - S Addr Wr [A] Data [A] P
- SMBus Read Byte
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [Data] NA P
- SMBus Read Word
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [DataLow] A [DataHigh] NA P
- SMBus Write Byte
  - S Addr Wr [A] Comm [A] Data [A] P
- SMBus Write Word
  - S Addr Wr [A] Comm [A] DataLow [A] DataHigh [A] P

Linux I2C – I2C Bus Driver

- API for Smbus transaction – PartII
  - i2c_smbus_read_block_data()
  - i2c_smbus_write_block_data()
  - i2c_smbus_read_i2c_block_data()
  - i2c_smbus_write_i2c_block_data()

call

- i2c_smbus_xfer()
  - adapter->algo->smbus_xfer()
  - i2c_smbus_xfer_emulated()
    - i2c_transfer()

- SMBus Process Call
  - S Addr Wr [A] Comm [A] DataLow [A] DataHigh [A] Sr Addr Rd [A] [DataLow] A [DataHigh] NA P
- SMBus Block Read
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [Count] A [Data] A [Data] A ... A [Data] NA P
  - reads a block of up to 32 bytes from a device
- SMBus Block Write
  - S Addr Wr [A] Comm [A] Count [A] Data [A] Data [A] ... [A] Data [A] P
- SMBus Block Write - Block Read Process Call
  - S Addr Wr [A] Comm [A] Count [A] Data [A] ... Sr Addr Rd [A] [Count] A [Data] ... A P
  - snd then read 31 bytes
- I2C Block Read
  - S Addr Wr [A] Comm [A] Sr Addr Rd [A] [Data] A [Data] A ... A [Data] NA P
- I2C Block Write
  - S Addr Wr [A] Comm [A] Data [A] Data [A] ... [A] Data [A] P

Linux I2C – I2C Bus Driver

- Structures for I2C and SMBus transaction messages

```
extern int __i2c_transfer(struct i2c_adapter *adap, struct i2c_msg *msgs,
                          int num);
```

```
struct i2c_msg {
        __u16 addr;     /* slave address                        */
        __u16 flags;
#define I2C_M_RD                0x0001  /* read data, from slave to master */
                                        /* I2C_M_RD is guaranteed to be 0x0001! */
#define I2C_M_TEN               0x0010  /* this is a ten bit chip address */
#define I2C_M_DMA_SAFE          0x0200  /* the buffer of this message is DMA safe */
                                        /* makes only sense in kernelspace */
                                        /* usersapce buffers are copied anyway */
#define I2C_M_RECV_LEN          0x0400  /* length will be first received byte */
#define I2C_M_NO_RD_ACK         0x0800  /* if I2C_FUNC_PROTOCOL_MANGLING */
#define I2C_M_IGNORE_NAK        0x1000  /* if I2C_FUNC_PROTOCOL_MANGLING */
#define I2C_M_REV_DIR_ADDR      0x2000  /* if I2C_FUNC_PROTOCOL_MANGLING */
#define I2C_M_NOSTART           0x4000  /* if I2C_FUNC_NOSTART */
#define I2C_M_STOP              0x8000  /* if I2C_FUNC_PROTOCOL_MANGLING */
        __u16 len;              /* msg length                           */
        __u8 *buf;              /* pointer to msg data                  */
};
```

data buffer

```
extern s32 i2c_smbus_xfer(struct i2c_adapter *adapter, u16 addr,
                          unsigned short flags, char read_write, u8 command,
                          int size, union i2c_smbus_data *data);
```

```
#define I2C_SMBUS_BLOCK_MAX     32      /* A
union i2c_smbus_data {
        __u8 byte;
        __u16 word;
        __u8 block[I2C_SMBUS_BLOCK_MAX + 2];
                                        /* and one mo
};
```

Linux I2C – I2C Bus Driver

- Functionalities

```
/* To determine what functionality is present */

#define I2C_FUNC_I2C                    0x00000001
#define I2C_FUNC_10BIT_ADDR             0x00000002
#define I2C_FUNC_PROTOCOL_MANGLING      0x00000004 /* I2C_M_IGNORE_NAK etc. */
#define I2C_FUNC_SMBUS_PEC              0x00000008
#define I2C_FUNC_NOSTART                0x00000010 /* I2C_M_NOSTART */
#define I2C_FUNC_SLAVE                  0x00000020
#define I2C_FUNC_SMBUS_BLOCK_PROC_CALL  0x00008000 /* SMBus 2.0 */
#define I2C_FUNC_SMBUS_QUICK            0x00010000
#define I2C_FUNC_SMBUS_READ_BYTE        0x00020000
#define I2C_FUNC_SMBUS_WRITE_BYTE       0x00040000
#define I2C_FUNC_SMBUS_READ_BYTE_DATA   0x00080000
#define I2C_FUNC_SMBUS_WRITE_BYTE_DATA  0x00100000
#define I2C_FUNC_SMBUS_READ_WORD_DATA   0x00200000
#define I2C_FUNC_SMBUS_WRITE_WORD_DATA  0x00400000
#define I2C_FUNC_SMBUS_PROC_CALL        0x00800000
#define I2C_FUNC_SMBUS_READ_BLOCK_DATA  0x01000000
#define I2C_FUNC_SMBUS_WRITE_BLOCK_DATA 0x02000000
#define I2C_FUNC_SMBUS_READ_I2C_BLOCK   0x04000000 /* I2C-like block xfer  */
#define I2C_FUNC_SMBUS_WRITE_I2C_BLOCK  0x08000000 /* w/ 1-byte reg. addr. */
#define I2C_FUNC_SMBUS_HOST_NOTIFY      0x10000000
```

combinations

```
#define I2C_FUNC_SMBUS_BYTE             (I2C_FUNC_SMBUS_READ_BYTE | \
                                         I2C_FUNC_SMBUS_WRITE_BYTE)
#define I2C_FUNC_SMBUS_BYTE_DATA        (I2C_FUNC_SMBUS_READ_BYTE_DATA | \
                                         I2C_FUNC_SMBUS_WRITE_BYTE_DATA)
#define I2C_FUNC_SMBUS_WORD_DATA        (I2C_FUNC_SMBUS_READ_WORD_DATA | \
                                         I2C_FUNC_SMBUS_WRITE_WORD_DATA)
#define I2C_FUNC_SMBUS_BLOCK_DATA       (I2C_FUNC_SMBUS_READ_BLOCK_DATA | \
                                         I2C_FUNC_SMBUS_WRITE_BLOCK_DATA)
#define I2C_FUNC_SMBUS_I2C_BLOCK        (I2C_FUNC_SMBUS_READ_I2C_BLOCK | \
                                         I2C_FUNC_SMBUS_WRITE_I2C_BLOCK)

#define I2C_FUNC_SMBUS_EMUL             (I2C_FUNC_SMBUS_QUICK | \
                                         I2C_FUNC_SMBUS_BYTE | \
                                         I2C_FUNC_SMBUS_BYTE_DATA | \
                                         I2C_FUNC_SMBUS_WORD_DATA | \
                                         I2C_FUNC_SMBUS_PROC_CALL | \
                                         I2C_FUNC_SMBUS_WRITE_BLOCK_DATA | \
                                         I2C_FUNC_SMBUS_I2C_BLOCK | \
                                         I2C_FUNC_SMBUS_PEC)
```

What about I2C_FUNC_SMBUS_READ_BLOCK_DATA?

Linux I2C – I2C Bus Driver

- Device Model data structures
  - **Adapter**
    - Bus：struct bus_type **platform_bus_type**
    - Driver：struct **platform_driver**
    - Device：struct **i2c_adapter**
      - Struct **i2c_algorithm**
  - **Client**
    - Bus：struct bus_type **i2c_bus_type**
    - Driver：Struct **i2c_driver**
    - Device：Struct **i2c_client**

※ Adapter could belong to other device driver
- module_usb_driver(/drivers/i2c/busses/i2c-robotfuzz-osif.c)
- module_acpi_driver(/drivers/i2c/busses/i2c-scmi.c)
- module_serio_driver(/i2c/busses/i2c-taos-evm.c)
- isa_driver(/drivers/i2c/busses/i2c-elektor.c)

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – I2C Adapter Driver

- Adapter driver(/drivers/i2c/busses/i2c-cadence.c)
  - struct cdns_i2c
  - struct i2c_algorithm cdns_i2c_algo
  - cdns_i2c_probe()
  - cdns_i2c_remove()
  - struct platform_driver cdns_i2c_drv
  - module_platform_driver(cdns_i2c_drv)

Linux I2C – I2C Adapter Driver

- i2c_adapter structure

```
struct cdns_i2c {
        struct device           *dev;
        void __iomem *membase;
        struct i2c_adapter adap;
        struct i2c_msg *p_msg;
        int err_status;
        struct completion xfer_done;
        unsigned char *p_send_buf;
        unsigned char *p_recv_buf;
        unsigned int send_count;
        unsigned int recv_count;
        unsigned int curr_recv_count;
        int irq;
        unsigned long input_clk;
        unsigned int i2c_clk;
        unsigned int bus_hold_flag;
        struct clk *clk;
        struct notifier_block clk_rate_change_nb;
        u32 quirks;
};
```

```
struct i2c_adapter {
        struct module *owner;
        unsigned int class;                     /* classes to allow probing for */
        const struct i2c_algorithm *algo;       /* the algorithm to access the bus */
        void *algo_data;

        /* data fields that are valid for all devices   */
        const struct i2c_lock_operations *lock_ops;
        struct rt_mutex bus_lock;
        struct rt_mutex mux_lock;

        int timeout;                    /* in jiffies */
        int retries;
        struct device dev;              /* the adapter device */

        int nr;
        char name[48];
        struct completion dev_released;

        struct mutex usepace_clients_lock;
        struct list_head usepace_clients;

        struct i2c_bus_recovery_info *bus_recovery_info;
        const struct i2c_adapter_quirks *quirks;

        struct irq_domain *host_notify_domain;
};
#define to_i2c_adapter(d) container_of(d, struct i2c_adapter, dev)
```

Linux I2C – I2C Adapter Driver

- i2c_algorithm structure

```
struct i2c_algorithm {
        /* If an adapter algorithm can't do I2C-level access, set master_xfer
           to NULL. If an adapter algorithm can do SMBus access, set
           smbus_xfer. If set to NULL, the SMBus protocol is simulated
           using common I2C messages */
        /* master_xfer should return the number of messages successfully
           processed, or a negative value on error */
        int (*master_xfer) struct i2c_adapter *adap, struct i2c_msg *msgs,
                           int num);
        int (*smbus_xfer) struct i2c_adapter *adap, u16 addr,
                          unsigned short flags, char read_write,
                          u8 command, int size, union i2c_smbus_data *data);

        /* To determine what the adapter supports */
        u32 (*functionality) (struct i2c_adapter *);

#if IS_ENABLED(CONFIG_I2C_SLAVE)
        int (*reg_slave)(struct i2c_client *client);
        int (*unreg_slave)(struct i2c_client *client);
#endif
};
```

- **master_xfer**
  - i2c access
  - can emulate Smbus protocol
- **smbus_xfer**
  - smbus access

Linux I2C – I2C Adapter Driver

- Algorithm

```
static int cdns_i2c_master_xfer(struct i2c_adapter *adap, struct i2c_msg *msgs,
                                int num)
{
```

```
static const struct i2c_algorithm cdns_i2c_algo = {
        .master_xfer    = cdns_i2c_master_xfer,
        .functionality  = cdns_i2c_func,
};
```

```
static u32 cdns_i2c_func(struct i2c_adapter *adap)
{
        return I2C_FUNC_I2C | I2C_FUNC_10BIT_ADDR |
                (I2C_FUNC_SMBUS_EMUL & ~I2C_FUNC_SMBUS_QUICK) |
                I2C_FUNC_SMBUS_BLOCK_DATA;
}
```

Linux I2C – I2C Adapter Driver

- Transaction
  - static int cdns_i2c_master_xfer()
    - cdns_i2c_process_msg()
      - cdns_i2c_mrecv()
        - cdns_i2c_readreg()
        - cdns_i2c_writereg()
      - cdns_i2c_msend()
        - cdns_i2c_readreg()
        - cdns_i2c_writereg()

```
i2c0: i2c@ff020000 {
        compatible = "cdns,i2c-r1p14", "cdns,i2c-r1p10";
        status = "disabled";
        interrupt-parent = <&gic>;
        interrupts = <0 17 4>;
        reg = <0x0 0xff020000 0x0 0x1000>;
        #address-cells = <1>;
        #size-cells = <0>;
};
```

```
struct cdns_i2c {
        struct device           *dev;
        void __iomem *membase;
        struct i2c_adapter adap;
```

```
#define cdns_i2c_readreg(offset)        readl_relaxed(id->membase + offset)
#define cdns_i2c_writereg(val, offset) writel_relaxed(val, id->membase + offset)
```

```
#define readl_relaxed(c)        ({ u32 __r = le32_to_cpu((__force __le32)__raw_readl(c)); __r; })
#define writel_relaxed(v,c)     ((void)__raw_writel((__force u32)cpu_to_le32(v),(c)))
```

Linux I2C – I2C Adapter Driver

- Memory-mapped I/O access

```
#define __raw_readl __raw_readl
static inline u32 __raw_readl(const volatile void __iomem *addr)
{
        u32 val;
        asm volatile(ALTERNATIVE("ldr %w0, [%1]",
                                 "ldar %w0, [%1]",
                                 ARM64_WORKAROUND_DEVICE_LOAD_ACQUIRE)
                     : "=r" (val) : "r" (addr));
        return val;
}
```

```
#define __raw_writel __raw_writel
static inline void __raw_writel(u32 val, volatile void __iomem *addr)
{
        asm volatile("str %w0, [%1]" : : "rZ" (val), "r" (addr));
}
```

Linux I2C – I2C Adapter Driver

- register offsets and bit mask definitions

```c
static void cdns_i2c_mrecv(struct cdns_i2c *id)
{

        ...

        /* Put the controller in master receive mode and clear the FIFO */
        ctrl_reg = cdns_i2c_readreg(CDNS_I2C_CR_OFFSET);
        ctrl_reg |= CDNS_I2C_CR_RW | CDNS_I2C_CR_CLR_FIFO;

        ...

        cdns_i2c_writereg(ctrl_reg, CDNS_I2C_CR_OFFSET);
```

```c
/* Register offsets for the I2C device. */
#define CDNS_I2C_CR_OFFSET              0x00
#define CDNS_I2C_SR_OFFSET              0x04
#define CDNS_I2C_ADDR_OFFSET            0x08
#define CDNS_I2C_DATA_OFFSET            0x0C
#define CDNS_I2C_ISR_OFFSET             0x10
#define CDNS_I2C_XFER_SIZE_OFFSET       0x14
#define CDNS_I2C_TIME_OUT_OFFSET        0x1C
#define CDNS_I2C_IER_OFFSET             0x24
#define CDNS_I2C_IDR_OFFSET             0x28
```

```c
/* Control Register Bit mask definitions */
#define CDNS_I2C_CR_HOLD                BIT(4) /* Hold Bus bit */
#define CDNS_I2C_CR_ACK_EN              BIT(3)
#define CDNS_I2C_CR_NEA                 BIT(2)
#define CDNS_I2C_CR_MS                  BIT(1)
/* Read or Write Master transfer 0 - Transmitter, 1 - Receiver */
#define CDNS_I2C_CR_RW                  BIT(0)
/* Clear data in FIFO to zeroes */
#define CDNS_I2C_CR_CLR_FIFO            BIT(6)
```

Linux I2C – I2C Adapter Driver

- Probe / remove

```
static int cdns_i2c_probe(struct platform_device *pdev)
{
        struct resource *r_mem;
        struct cdns_i2c *id;
        int ret;
        const struct of_device_id *match;

        id = devm_kzalloc(&pdev->dev, sizeof(*id), GFP_KERNEL);
        if (!id)
                return -ENOMEM;
        ...
        id->dev = &pdev->dev;
        platform_set_drvdata(pdev, id);
        ...
        id->adap.owner = THIS_MODULE;
        id->adap.dev.of_node = pdev->dev.of_node;
        id->adap.algo = &cdns_i2c_algo;
        id->adap.timeout = CDNS_I2C_TIMEOUT;
        id->adap.retries = 3;           /* Default retry value. */
        id->adap.algo_data = id;
        id->adap.dev.parent = &pdev->dev;
        ...
        ret = devm_request_irq(&pdev->dev, id->irq, cdns_i2c_isr, 0,
                               DRIVER_NAME, id);
        ...
        ret = i2c_add_adapter(&id->adap);
        if (ret < 0)
                goto err_clk_dis;
```

```
static int cdns_i2c_remove(struct platform_device *pdev)
{
        struct cdns_i2c *id = platform_get_drvdata(pdev);

        i2c_del_adapter(&id->adap);
        clk_notifier_unregister(id->clk, &id->clk_rate_change_nb);
        clk_disable_unprepare(id->clk);
        pm_runtime_disable(&pdev->dev);

        return 0;
}
```

Linux I2C – I2C Adapter Driver

- Get/set driver data

```
static inline void platform_set_drvdata(struct platform_device *pdev,
                                        void *data)
{
        dev_set_drvdata(&pdev->dev, data);
}
```

```
static inline void dev_set_drvdata(struct device *dev, void *data)
{
        dev->driver_data = data;
}
```

```
static inline void *platform_get_drvdata(const struct platform_device *pdev)
{
        return dev_get_drvdata(&pdev->dev);
}
```

```
static inline void *dev_get_drvdata(const struct device *dev)
{
        return dev->driver_data;
}
```

Linux I2C – I2C Adapter Driver

- i2c_add_adapter() / i2c_del_adapter()

- i2c_add_adapter()
  - i2c_register_adapter()
    - device_register(&adap->dev)
      - device_add()
  - of_i2c_register_devices()
    - of_i2c_register_device()
      - i2c_new_device()
        - device_register(&client->dev)
          - device_add()

- i2c_del_adapter()
  - i2c_unregister_device(client)
    - device_unregister(&client->dev)
      - device_del()
  - device_unregister(&adap->dev)
    - device_del()

Linux I2C – I2C Adapter Driver

- Matching

```
static const struct of_device_id cdns_i2c_of_match[] = {
        { .compatible = "cdns,i2c-r1p10", .data = &r1p10_i2c_def },
        { .compatible = "cdns,i2c-r1p14" },
        { /* end of table */ }
};
MODULE_DEVICE_TABLE(of, cdns_i2c_of_match);
```

/arch/arm64/boot/dts/xilinx/zynqmp.dtsi

```
i2c0: i2c@ff020000 {
        compatible = "cdns,i2c-r1p14", "cdns,i2c-r1p10";
        status = "disabled";
        interrupt-parent = <&gic>;
        interrupts = <0 17 4>;
        reg = <0x0 0xff020000 0x0 0x1000>;
        #address-cells = <1>;
        #size-cells = <0>;
};

i2c1: i2c@ff030000 {
        compatible = "cdns,i2c-r1p14", "cdns,i2c-r1p10";
        status = "disabled";
        interrupt-parent = <&gic>;
        interrupts = <0 18 4>;
        reg = <0x0 0xff030000 0x0 0x1000>;
        #address-cells = <1>;
        #size-cells = <0>;
};
```

Linux I2C – I2C Adapter Driver

- Device driver structure

```
static struct platform_driver cdns_i2c_drv = {
        .driver = {
                .name  = DRIVER_NAME,
                .of_match_table = cdns_i2c_of_match,
                .pm = &cdns_i2c_dev_pm_ops,
        },
        .probe  = cdns_i2c_probe,
        .remove = cdns_i2c_remove,
};

module_platform_driver(cdns_i2c_drv);
```

Linux I2C – I2C Adapter Driver

- Driver registration/unregistration

```
#define module_platform_driver(__platform_driver) \
        module_driver(__platform_driver, platform_driver_register, \
                      platform_driver_unregister)
```

```
#define module_driver(__driver, __register, __unregister, ...) \
static int __init __driver##_init(void) \
{ \
        return __register(&(__driver) , ##__VA_ARGS__); \
} \
module_init(__driver##_init); \
static void __exit __driver##_exit(void) \
{ \
        __unregister(&(__driver) , ##__VA_ARGS__); \
} \
module_exit(__driver##_exit);
```

```
static int __init cdns_i2c_drv_init(void)
{
        return platform_driver_register(&(cdns_i2c_drv));
}
module_init(cdns_i2c_drv_init);
static void __exit cdns_i2c_drv_exit(void)
{
        platform_driver_unregister(&(cdns_i2c_drv));
}
module_exit(cdns_i2c_drv_exit);;
```

Linux I2C – I2C Adapter Driver

- platform_driver_register() / platrom_driver_unregister()

```
#define platform_driver_register(drv) \
        __platform_driver_register(drv, THIS_MODULE)
```

```
int __platform_driver_register(struct platform_driver *drv,
                               struct module *owner)
{
        drv->driver.owner = owner;
        drv->driver.bus = &platform_bus_type;
        drv->driver.probe = platform_drv_probe;
        drv->driver.remove = platform_drv_remove;
        drv->driver.shutdown = platform_drv_shutdown;

        return driver_register(&drv->driver);
}
EXPORT_SYMBOL_GPL(__platform_driver_register);
```

```
void platform_driver_unregister(struct platform_driver *drv)
{
        driver_unregister(&drv->driver);
}
EXPORT_SYMBOL_GPL(platform_driver_unregister);
```

Linux I2C – I2C Adapter Driver

- driver_register()
  - bus_add_driver()
    - driver_attach()
      - __driver_attach()
        - driver_probe_device()
          - really_probe()

```
static int really_probe(struct device *dev, struct device_driver *drv)
{
        ...
        if (dev->bus->probe) {
                ret = dev->bus->probe(dev);
                if (ret)
                        goto probe_failed;
        } else if (drv->probe) {
                ret = drv->probe(dev);
                if (ret)
                        goto probe_failed;
        }
```

```
struct bus_type platform_bus_type = {
        .name           = "platform",
        .dev_groups     = platform_dev_groups,
        .match          = platform_match,
        .uevent         = platform_uevent,
        .dma_configure  = platform_dma_configure,
        .pm             = &platform_dev_pm_ops,
};
EXPORT_SYMBOL_GPL(platform_bus_type);
```

probe not assigned in Platform_bus_type
so drv->probe will be called

Linux I2C – I2C Adapter Driver

- driver_unregister()
  - Bus_remove_driver()
    - Driver_detach()
      - Device_release_driver_internal()
        - __device_release_driver()

```
static void __device_release_driver(struct device *dev, struct device *parent)
{
        struct device_driver *drv;
        ...
        if (dev->bus && dev->bus->remove)
                dev->bus->remove(dev);
        else if (drv->remove)
                drv->remove(dev);
```

```
struct bus_type platform_bus_type = {
        .name           = "platform",
        .dev_groups     = platform_dev_groups,
        .match          = platform_match,
        .uevent         = platform_uevent,
        .dma_configure  = platform_dma_configure,
        .pm             = &platform_dev_pm_ops,
};
EXPORT_SYMBOL_GPL(platform_bus_type);
```

remove not assigned in platform_bus_type so drv->remove will be called

Linux I2C – I2C Adapter Driver

- platform_drv_probe() / platform_drv_remove()

```
static int platform_drv_probe(struct device *_dev)
{
        struct platform_driver *drv = to_platform_driver(_dev->driver);
        struct platform_device *dev = to_platform_device(_dev);
        int ret;

        ret = of_clk_set_defaults(_dev->of_node, false);
        if (ret < 0)
                return ret;

        ret = dev_pm_domain_attach(_dev, true);
        if (ret)
                goto out;

        if (drv->probe) {
                ret = drv->probe(dev);
                if (ret)
                        dev_pm_domain_detach(_dev, true);
        }

out:
        if (drv->prevent_deferred_probe && ret == -EPROBE_DEFER) {
                dev_warn(_dev, "probe deferral not supported\n");
                ret = -ENXIO;
        }

        return ret;
}
```

```
static int platform_drv_remove(struct device *_dev)
{
        struct platform_driver *drv = to_platform_driver(_dev->driver);
        struct platform_device *dev = to_platform_device(_dev);
        int ret = 0;

        if (drv->remove)
                ret = drv->remove(dev);
        dev_pm_domain_detach(_dev, true);

        return ret;
}
```

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – I2C-Dev Driver

- I2c-dev.c
  - i2cdev_read()
  - i2cdev_write()
  - I2cdev_ioctl()
  - I2cdev_open()
  - I2cdev_release()
  - Struct file_operations i2cdev_fops
  - i2c_dev_init()
  - i2c_dev_exit()

Linux I2C – I2C-Dev Driver

- i2cdev_open()

```
static int i2cdev_open(struct inode *inode, struct file *file)
{
        unsigned int minor = iminor(inode);
        struct i2c_client *client;
        struct i2c_adapter *adap;

        adap = i2c_get_adapter(minor);
        if (!adap)
                return -ENODEV;

        /* This creates an anonymous i2c_client, which may later be
         * pointed to some address using I2C_SLAVE or I2C_SLAVE_FORCE.
         *
         * This client is ** NEVER REGISTERED ** with the driver model
         * or I2C core code!!  It just holds private copies of addressing
         * information and maybe a PEC flag.
         */
        client = kzalloc(sizeof(*client), GFP_KERNEL);
        if (!client) {
                i2c_put_adapter(adap);
                return -ENOMEM;
        }
        snprintf(client->name, I2C_NAME_SIZE, "i2c-dev %d", adap->nr);

        client->adapter = adap;
        file->private_data = client;

        return 0;
}
```

- i2c_get_adapter()
  - idr_find()
  - try_module_get()
  - get_device()
- creates an anonymous i2c_client
- can be pointed to address using I2C_SLAVE or I2C_SLAVE_FORCE
- assign file->private_data to client

Linux I2C – I2C-Dev Driver

- i2cdev_release()

```
static int i2cdev_release(struct inode *inode, struct file *file)
{
        struct i2c_client *client = file->private_data;

        i2c_put_adapter(client->adapter);
        kfree(client);
        file->private_data = NULL;

        return 0;
}
```

- i2c_put_adapter()
  - put_device()
  - module_put()
- kfree client
- assign file->private_data to NULL

Linux I2C – I2C-Dev Driver

- i2cdev_write()
  - medup_user()
    - copy_from_user()
  - i2c_master_send()
    - i2c_transfer_buffer_flasgs(..., 0)
      - i2c_transfer()
        - __i2c_transfer()
          - adap->algo->master_xfer()

- i2cdev_read()
  - i2c_master_recv()
    - i2c_transfer_buffer_flags(..., I2C_M_RD)
      - i2c_transfer()
        - __i2c_transfer()
          - adap->algo->master_xfer()
  - copy_to_user()

eventually calls algo->master_xfer()
which means SMBus adapters won't support
these functions

Linux I2C – I2C-Dev Driver

- i2cdev_ioctl() – set slave address

```
static long i2cdev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
        struct i2c_client *client = file->private_data;
        unsigned long funcs;

        dev_dbg(&client->adapter->dev, "ioctl, cmd=0x%02x, arg=0x%02lx\n",
                cmd, arg);

        switch (cmd) {
        case I2C_SLAVE:
        case I2C_SLAVE_FORCE:
                if ((arg > 0x3ff) ||
                    (((client->flags & I2C_M_TEN) == 0) && arg > 0x7f))
                        return -EINVAL;
                if (cmd == I2C_SLAVE && i2cdev_check_addr(client->adapter, arg))
                        return -EBUSY;
                /* REVISIT: address could become busy later */
                client->addr = arg;
                return 0;
```

- if cmd==I2C_SLAVE_FORCE, i2cdev_check_addr won't be called
- i2cdev_check_addr()
  - i2cdev_check()
- busy if driver bound to it

```
static int i2cdev_check(struct device *dev, void *addrp)
{
        struct i2c_client *client = i2c_verify_client(dev);

        if (!client || client->addr != *(unsigned int *)addrp)
                return 0;

        return dev->driver ? -EBUSY : 0;
}
```

Linux I2C – I2C-Dev Driver

- i2cdev_ioctl() – ten bit address

```
static long i2cdev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
        ...
        case I2C_TENBIT:
                if (arg)
                        client->flags |= I2C_M_TEN;
                else
                        client->flags &= ~I2C_M_TEN;
                return 0;
```

Linux I2C – I2C-Dev Driver

- i2cdev_ioctl() – get functionality

```
static long i2cdev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
        ...
        case I2C_FUNCS:
                funcs = i2c_get_functionality(client->adapter);
                return put_user(funcs, (unsigned long __user *)arg);
```

```
static inline u32 i2c_get_functionality(struct i2c_adapter *adap)
{
        return adap->algo->functionality(adap);
}
```

Linux I2C – I2C-Dev Driver

- i2cdev_ioctl() – I2C transaction

```
static long i2cdev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
  ...
  case I2C_RDWR: {
        struct i2c_rdwr_ioctl_data rdwr_arg;
        struct i2c_msg *rdwr_pa;

        if (copy_from_user(&rdwr_arg,
                           (struct i2c_rdwr_ioctl_data __user *)arg,
                           sizeof(rdwr_arg)))
                return -EFAULT;
        if (rdwr_arg.nmsgs > I2C_RDWR_IOCTL_MAX_MSGS)
                return -EINVAL;

        rdwr_pa = memdup_user(rdwr_arg.msgs,
                              rdwr_arg.nmsgs * sizeof(struct i2c_msg));
        if (IS_ERR(rdwr_pa))
                return PTR_ERR(rdwr_pa);

        return i2cdev_ioctl_rdwr(client, rdwr_arg.nmsgs, rdwr_pa);
  }
```

- copy i2c_rdwr_ioctl_data
- copy i2c_msg
- i2cdev_ioctl_rdwr()

Linux I2C – I2C-Dev Driver

- i2cdev_ioctl() – I2C transaction

```
static noinline int i2cdev_ioctl_rdwr(struct i2c_client *client,
                unsigned nmsgs, struct i2c_msg *msgs)
{
        u8 __user **data_ptrs;
        int i, res;
        ...
        data_ptrs = kmalloc_array(nmsgs, sizeof(u8 __user *), GFP_KERNEL);
        ...
        for (i = 0; i < nmsgs; i++) {
                /* Limit the size of the message to a sane amount */

                data_ptrs[i] = (u8 __user *)msgs[i].buf;
                msgs[i].buf = memdup_user(data_ptrs[i], msgs[i].len);
        ...
        res = i2c_transfer(client->adapter, msgs, nmsgs);
        while (i-- > 0) {
                if (res >= 0 && (msgs[i].flags & I2C_M_RD)) {
                        if (copy_to_user(data_ptrs[i], msgs[i].buf,
                                         msgs[i].len))
                                res = -EFAULT;
                }
```

- copy i2c_msg buf
- i2c_transfer()
- copy to user if I2C_M_RD is set

Linux I2C – I2C-Dev Driver

- Structures for I2C transaction

```
struct i2c_rdwr_ioctl_data {
        struct i2c_msg __user *msgs;
        __u32 nmsgs;
};
```

```
struct i2c_msg {
        __u16 addr;     /* slave address                        */
        __u16 flags;
#define I2C_M_RD                0x0001  /* read data, from slave to master */
                                        /* I2C_M_RD is guaranteed to be 0x0001! */
#define I2C_M_TEN               0x0010  /* this is a ten bit chip address */
#define I2C_M_DMA_SAFE          0x0200  /* the buffer of this message is DMA safe */
                                        /* makes only sense in kernelspace */
                                        /* userspace buffers are copied anyway */
#define I2C_M_RECV_LEN          0x0400  /* length will be first received byte */
#define I2C_M_NO_RD_ACK         0x0800  /* if I2C_FUNC_PROTOCOL_MANGLING */
#define I2C_M_IGNORE_NAK        0x1000  /* if I2C_FUNC_PROTOCOL_MANGLING */
#define I2C_M_REV_DIR_ADDR      0x2000  /* if I2C_FUNC_PROTOCOL_MANGLING */
#define I2C_M_NOSTART           0x4000  /* if I2C_FUNC_NOSTART */
#define I2C_M_STOP              0x8000  /* if I2C_FUNC_PROTOCOL_MANGLING */
        __u16 len;              /* msg length                           */
        __u8 *buf;              /* pointer to msg data                  */
};
```

data buffer

Linux I2C – I2C-Dev Driver

- SMBus protocol transaction

```
static long i2cdev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
    ...
    case I2C_SMBUS: {
            struct i2c_smbus_ioctl_data data_arg;
            if (copy_from_user(&data_arg,
                               (struct i2c_smbus_ioctl_data __user *) arg,
                               sizeof(struct i2c_smbus_ioctl_data)))
                    return -EFAULT;
            return i2cdev_ioctl_smbus(client, data_arg.read_write,
                                      data_arg.command,
                                      data_arg.size,
                                      data_arg.data);
    }
```

- copy i2c_smbus_ioctl_data
- i2cdev_ioctl_smbus()

Linux I2C – I2C-Dev Driver

- SMBus protocol transaction

```
static noinline int i2cdev_ioctl_smbus(struct i2c_client *client,
                u8 read_write, u8 command, u32 size,
                union i2c_smbus_data __user *data)
{
        union i2c_smbus_data temp = {};
        int datasize, res;
        ...
        if ((size == I2C_SMBUS_BYTE_DATA) ||
            (size == I2C_SMBUS_BYTE))
                datasize = sizeof(data->byte);
        ...
                if (copy_from_user(&temp, data, datasize))
                        return -EFAULT;
        }
        ...
        res = i2c_smbus_xfer(client->adapter, client->addr, client->flags,
              read_write, command, size, &temp);
        ...
                if (copy_to_user(data, &temp, datasize))
                        return -EFAULT;
        }
```

- copy i2c_smbus_data data
- i2c_smbus_xfer()
- copy to user if I2C_SMBUS_READ or PROC_CALL is set

Linux I2C – I2C-Dev Driver

- Structures for SMBus transaction

```
struct i2c_smbus_ioctl_data {
        __u8 read_write;
        __u8 command;
        __u32 size;
        union i2c_smbus_data __user *data;
};
```

```
#define I2C_SMBUS_BLOCK_MAX     32      /* A
union i2c_smbus_data {
        __u8 byte;
        __u16 word;
        __u8 block[I2C_SMBUS_BLOCK_MAX + 2];
                                /* and one ma
};
```

```
/* i2c_smbus_xfer read or write markers */
#define I2C_SMBUS_READ  1
#define I2C_SMBUS_WRITE 0
```

```
#define I2C_SMBUS_QUICK             0
#define I2C_SMBUS_BYTE              1
#define I2C_SMBUS_BYTE_DATA         2
#define I2C_SMBUS_WORD_DATA         3
#define I2C_SMBUS_PROC_CALL         4
#define I2C_SMBUS_BLOCK_DATA        5
#define I2C_SMBUS_I2C_BLOCK_BROKEN  6
#define I2C_SMBUS_BLOCK_PROC_CALL   7
#define I2C_SMBUS_I2C_BLOCK_DATA    8
```

Linux I2C – I2C-Dev Driver

- struct file_operations i2cdev_fops

```
static const struct file_operations i2cdev_fops = {
        .owner          = THIS_MODULE,
        .llseek         = no_llseek,
        .read           = i2cdev_read,
        .write          = i2cdev_write,
        .unlocked_ioctl = i2cdev_ioctl,
        .compat_ioctl   = compat_i2cdev_ioctl,
        .open           = i2cdev_open,
        .release        = i2cdev_release,
};
```

Userspace

- read()
- write()
- Ioctl()
- open()
- close()

Linux I2C – I2C-Dev Driver

- i2c_dev_init()

```
static int __init i2c_dev_init(void)
{
        ...
        res = register_chrdev_region(MKDEV(I2C_MAJOR, 0), I2C_MINORS, "i2c");
        ...
        i2c_dev_class = class_create(THIS_MODULE, "i2c-dev");
        ...
        i2c_for_each_dev(NULL, i2cdev_attach_adapter);
```

- register major number(89) and minor numbers
- create class structure
- call i2c_attach_adapter for every adapter

```
static int i2cdev_attach_adapter(struct device *dev, void *dummy)
{
        ...
        cdev_init(&i2c_dev->cdev, &i2cdev_fops);
        i2c_dev->cdev.owner = THIS_MODULE;
        res = cdev_add(&i2c_dev->cdev, MKDEV(I2C_MAJOR, adap->nr), 1);
        ...
        i2c_dev->dev = device_create(i2c_dev_class, &adap->dev,
                                     MKDEV(I2C_MAJOR, adap->nr), NULL,
                                     "i2c-%d", adap->nr);
```

- create cdev structure
- add character device to system
- create device (/dev/i2c-%d)

Linux I2C – I2C-Dev Driver

- i2c_dev_exit()

```
static void __exit i2c_dev_exit(void)
{
        bus_unregister_notifier(&i2c_bus_type, &i2cdev_notifier);
        i2c_for_each_dev(NULL, i2cdev_detach_adapter);
        class_destroy(i2c_dev_class);
        unregister_chrdev_region(MKDEV(I2C_MAJOR, 0), I2C_MINORS);
}
```

- call i2c_detach_adapter() for every adapter
- destroy class structure
- unregister major number(89) and minor numbers

```
static int i2cdev_detach_adapter(struct device *dev, void *dummy)
{
    ...
    cdev_del(&i2c_dev->cdev);
    put_i2c_dev(i2c_dev);
    device_destroy(i2c_dev_class, MKDEV(I2C_MAJOR, adap->nr));
```

- create cdev structure
- destroy device

Linux I2C – I2C-Dev Driver

Terminal - kaiden@debian: /

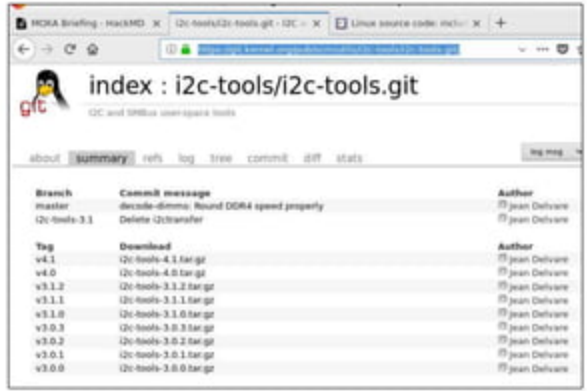
File Edit View Terminal Tabs Help

```
kaiden@debian:/$ sudo modprobe i2c-dev
kaiden@debian:/$ ls -al /dev | grep i2c
crw-rw----    1 root i2c      89,   0 Sep 30 10:08 i2c-0
crw-rw----    1 root i2c      89,   1 Sep 30 10:08 i2c-1
crw-rw----    1 root i2c      89,  10 Sep 30 10:08 i2c-10
crw-rw----    1 root i2c      89,   2 Sep 30 10:08 i2c-2
crw-rw----    1 root i2c      89,   3 Sep 30 10:08 i2c-3
crw-rw----    1 root i2c      89,   4 Sep 30 10:08 i2c-4
crw-rw----    1 root i2c      89,   5 Sep 30 10:08 i2c-5
crw-rw----    1 root i2c      89,   6 Sep 30 10:08 i2c-6
crw-rw----    1 root i2c      89,   7 Sep 30 10:08 i2c-7
crw-rw----    1 root i2c      89,   8 Sep 30 10:08 i2c-8
crw-rw----    1 root i2c      89,   9 Sep 30 10:08 i2c-9
kaiden@debian:/$
```

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – I2C-Tools

- I2C-tools
  - i2cdetect
  - i2cdump
  - i2cset
  - i2cget
  - i2ctransfer
    - By “apt-get install i2c-tools” this executable won’t be included



Linux I2C – I2C-Tools

- i2cdetect：show all the adapter(bus) info

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdetect -l
i2c-3   i2c         i915 gmbus dpd                          I2C adapter
i2c-1   i2c         i915 gmbus dpc                          I2C adapter
i2c-8   i2c         nvkm-0000:01:00.0-bus-0005              I2C adapter
i2c-6   i2c         nvkm-0000:01:00.0-bus-0001              I2C adapter
i2c-4   i2c         DPDDC-A                                 I2C adapter
i2c-2   i2c         i915 gmbus dpb                          I2C adapter
i2c-0   smbus       SMBus I801 adapter at f040              SMBus adapter
i2c-9   i2c         Synopsys DesignWare I2C adapter         I2C adapter
i2c-10  i2c         Synopsys DesignWare I2C adapter         I2C adapter
i2c-7   i2c         nvkm-0000:01:00.0-bus-0002              I2C adapter
i2c-5   i2c         DPDDC-B                                 I2C adapter
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

```
Terminal - kaiden@debian: /sys/class/i2c-dev/i2c-0
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:/sys/class/i2c-dev/i2c-0$ cat name
SMBus I801 adapter at f040
kaiden@debian:/sys/class/i2c-dev/i2c-0$
```

Linux I2C – I2C-Tools

- i2cdetect：check funtionality

Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools

File Edit View Terminal Tabs Help

```
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdetect -F 0
Functionalities implemented by /dev/i2c-0:
I2C                              no
SMBus Quick Command              yes
SMBus Send Byte                  yes
SMBus Receive Byte               yes
SMBus Write Byte                 yes
SMBus Read Byte                  yes
SMBus Write Word                 yes
SMBus Read Word                  yes
SMBus Process Call               no
SMBus Block Write                yes
SMBus Block Read                 yes
SMBus Block Process Call         no
SMBus PEC                        yes
I2C Block Write                  yes
I2C Block Read                   yes
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Tools

- i2cdetect：check slave devices attach to a certain adapter

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdetect -a 0
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-0.
I will probe address range 0x00-0x7f.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00 -- -- -- -- -- -- -- 08 -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: 30 31 32 -- 34 35 -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- -- -- --
50: -- -- 52 -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Tools

- i2cdump：show the values of all the registers within a slave device

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdump -a 0 0x52
No size specified (using byte-data access)
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-0, address 0x52, mode byte
Continue? [Y/n] y
     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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
40: 01 98 05 16 30 1f 39 ac e0 39 39 30 35 36 32 34    ????0?9?9905624
50: 2d 30 31 30 2e 41 30 30 47 20 20 20 20 00 80 2c    -010.A00G    .?,
60: 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 88    B...............?
70: 07 37 34 33 39 30 34 34 00 00 01 01 00 00 00 00    ?7439044..??....
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Tools

- i2cget/i2cset：SMBus transaction
- i2ctransfer：I2C transaction

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo
No size specified (using byte-data access)
WARNING! This program can confuse your I2C bus, cause
I will probe file /dev/i2c-0, address 0x52, mode byte
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40: 01 98 05 16 30 1f 39 ac e0 39 39 30 35 36 32 34
50: 2d 30 31 30 2e 41 30 30 47 20 20 20 20 00 80 2c
60: 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 88
70: 07 37 34 33 39 30 34 34 00 00 01 01 00 00 00 00
```

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cget 0 0x52 0x41
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will read from device file /dev/i2c-0, chip address 0x52, data address
0x41, using read byte data.
Continue? [Y/n] y
0x98
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
Error: Adapter does not have I2C transfers capability
```

I2C transaction not supported by a SMBus adapter

Linux I2C – I2C-Tools

- i2cget/i2cset：SMBus transaction
- i2ctransfer：I2C transaction

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sud
[sudo] password for kaiden:
No size specified (using byte-data access)
WARNING! This program can confuse your I2C bus, caus
I will probe file /dev/i2c-4, address 0x50, mode byt
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00 ff ff ff ff ff ff ff 00 06 af ed 38 00 00 00 00
10: 2b 18 01 04 95 22 13 78 02 51 25 93 58 57 8f 28
20: 1f 50 54 00 00 00 01 01 01 01 01 01 01 01 01 01
30: 01 01 01 01 01 01 78 37 80 b4 70 38 2e 40 6c 30
40: aa 00 58 c1 10 00 00 18 00 00 00 0f 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 20 00 00 00 fe 00 41
60: 55 4f 0a 20 20 20 20 20 20 20 20 20 00 00 00 fe
70: 00 42 31 35 36 48 54 4e 30 33 2e 38 20 0a 00 e9
```

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cget 4 0x50 0x60
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will read from device file /dev/i2c-4, chip address 0x50, data address
0x60, using read byte data.
Continue? [Y/n] y
0x55
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2ctransfer 4 w1@0x50 0x60 r1
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will send the following messages to device file /dev/i2c-4:
msg 0: addr 0x50, write, len 1, buf 0x60
msg 1: addr 0x50, read, len 1
Continue? [y/N] y
0x55
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

both kinds of transaction work well on a I2C adatper

Linux I2C – I2C-Tools

- I2C-tools source code：tools/i2cbusses.c

```c
int open_i2c_dev(int i2cbus, char *filename, size_t size, int quiet)
{
    int file;

    snprintf(filename, size, "/dev/i2c/%d", i2cbus);
    filename[size - 1] = '\0';
    file = open(filename, O_RDWR);

    if (file < 0 && (errno == ENOENT || errno == ENOTDIR)) {
        sprintf(filename, "/dev/i2c-%d", i2cbus);
        file = open(filename, O_RDWR);
    }
```

```c
static enum adt i2c_get_funcs(int i2cbus)
{
    unsigned long funcs;
    int file;
    char filename[20];
    enum adt ret;

    file = open_i2c_dev(i2cbus, filename, sizeof(filename), 1);
    if (file < 0)
        return adt_unknown;

    if (ioctl(file, I2C_FUNCS, &funcs) < 0)
        ret = adt_unknown;
    else if (funcs & I2C_FUNC_I2C)
        ret = adt_i2c;
```

```c
int set_slave_addr(int file, int address, int force)
{
    /* with force, let the user read from/write to the registers
    /* even when a driver is also running */
    if (ioctl(file, force ? I2C_SLAVE_FORCE : I2C_SLAVE, address) < 0) {
        fprintf(stderr,
                "Error: Could not set address to 0x%02x: %s\n",
                address, strerror(errno));
        return -errno;
    }

    return 0;
}
```

Linux I2C – I2C-Tools

- I2C-tools source code：lib/smbus.c

```
__s32 i2c_smbus_read_byte(int file)
{
    union i2c_smbus_data data;
    int err;

    err = i2c_smbus_access(file, I2C_SMBUS_READ, 0, I2C_SMBUS_BYTE, &data);
    if (err < 0)
        return err;

    return 0x0FF & data.byte;
}
```

```
__s32 i2c_smbus_access(int file, char read_write, __u8 command,
                       int size, union i2c_smbus_data *data)
{
    struct i2c_smbus_ioctl_data args;
    __s32 err;

    args.read_write = read_write;
    args.command = command;
    args.size = size;
    args.data = data;

    err = ioctl(file, I2C_SMBUS, &args);
    if (err == -1)
        err = -errno;
    return err;
}
```

Linux I2C – I2C-Tools

- i2c-tools source code：tools/i2ctransfer.c

```
msgs[nmsgs].addr = address;
msgs[nmsgs].flags = flags;
msgs[nmsgs].len = len;
```

```
struct i2c_rdwr_ioctl_data rdwr;

rdwr.msgs = msgs;
rdwr.nmsgs = nmsgs;
nmsgs_sent = ioctl(file, I2C_RDWR, &rdwr);
if (nmsgs_sent < 0) {
    fprintf(stderr, "Error: Sending messages failed: %s\n", strerror(errno));
    goto err_out;
} else if (nmsgs_sent < nmsgs) {
    fprintf(stderr, "Warning: only %d/%d messages were sent\n", nmsgs_sent, nmsgs);
}
```

- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – I2C-Stub Driver

- I2C-stub driver： a fake I2C/SMBus driver

```
Terminal - kaiden@debian: /sys/bus/i2c/devices
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:/sys/bus/i2c/devices$ ls
i2c-0  i2c-10  i2c-3  i2c-5  i2c-7  i2c-9
i2c-1  i2c-2   i2c-4  i2c-6  i2c-8  i2c-FTE1001:00
kaiden@debian:/sys/bus/i2c/devices$ sudo modprobe i2c-stub chip_addr=0x50
kaiden@debian:/sys/bus/i2c/devices$ ls
i2c-0  i2c-10  i2c-2  i2c-4  i2c-6  i2c-8  i2c-FTE1001:00
i2c-1  i2c-11  i2c-3  i2c-5  i2c-7  i2c-9
kaiden@debian:/sys/bus/i2c/devices$
```

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdetect -l
i2c-3   i2c     i915 gmbus dpd          I2C adapter
i2c-1   i2c     i915 gmbus dpc          I2C adapter
i2c-11  smbus   SMBus stub driver       SMBus adapter
i2c-8   i2c     nvkm-0000:01:00.0-bus-0005  I2C adapter
i2c-6   i2c     nvkm-0000:01:00.0-bus-0001  I2C adapter
i2c-4   i2c     DPDDC-A                 I2C adapter
i2c-2   i2c     i915 gmbus dpb          I2C adapter
i2c-0   smbus   SMBus I801 adapter at f040  SMBus adapter
i2c-9   i2c     Synopsys DesignWare I2C adapter  I2C adapter
i2c-10  i2c     Synopsys DesignWare I2C adapter  I2C adapter
i2c-7   i2c     nvkm-0000:01:00.0-bus-0002  I2C adapter
i2c-5   i2c     DPDDC-B                 I2C adapter
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Stub Driver

- I2C-stub adapter funtionality

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdetect -F 11
Functionalities implemented by /dev/i2c-11:
I2C                              no
SMBus Quick Command              yes
SMBus Send Byte                  yes
SMBus Receive Byte               yes
SMBus Write Byte                 yes
SMBus Read Byte                  yes
SMBus Write Word                 yes
SMBus Read Word                  yes
SMBus Process Call               no
SMBus Block Write                no
SMBus Block Read                 no
SMBus Block Process Call         no
SMBus PEC                        no
I2C Block Write                  yes
I2C Block Read                   yes
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Stub Driver

- chip_addr：slave address of virtual client

```
Terminal - kaiden@debian: /sys/bus/i2c/devices
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:/sys/bus/i2c/devices$ ls
i2c-0  i2c-10  i2c-3  i2c-5  i2c-7  i2c-9
i2c-1  i2c-2   i2c-4  i2c-6  i2c-8  i2c-FTE1001:00
kaiden@debian:/sys/bus/i2c/devices$ sudo modprobe i2c-stub chip_addr=0x50
```

```
Terminal - kaiden@debian: ~/Kaidenspace/i2c-tools-4.1/tools
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdetect -a 11
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-11.
I will probe address range 0x00-0x7f
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Stub Driver

- I2C-stub adapter works as a real adapter (can do SMBus transfer)

```
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdump -a 11 0x50
No size specified (using byte-data access)
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-11, address 0x50, mode byte
Continue? [Y/n] y
     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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

```
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cset -y 11 0x50 0x00 0xde
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cset -y 11 0x50 0x01 0xad
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cset -y 11 0x50 0x02 0xbe
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cset -y 11 0x50 0x03 0xef
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$ sudo ./i2cdump -ya 11 0x50
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: de ad be ef 00 00 00 00 00 00 00 00 00 00 00 00    ????............
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
kaiden@debian:~/Kaidenspace/i2c-tools-4.1/tools$
```

Linux I2C – I2C-Stub Driver

- I2C driver testing with i2c-stub adapter
  - addresses of **eeprom** device are between 0x50~ 0x57
  - with chip_addr=0x50, device will be bound after modprobe eeprom driver

Terminal - kaiden@debian: /sys/bus/i2c/devices/11-0050

File Edit View Terminal Tabs Help

```
kaiden@debian:/sys/bus/i2c/devices$ ls
i2c-0  i2c-10  i2c-2  i2c-4  i2c-6  i2c-8  i2c-FTE1001:00
i2c-1  i2c-11  i2c-3  i2c-5  i2c-7  i2c-9
kaiden@debian:/sys/bus/i2c/devices$ sudo modprobe eeprom
kaiden@debian:/sys/bus/i2c/devices$ ls
0-0052  4-0050  i2c-1   i2c-11  i2c-3  i2c-5  i2c-7  i2c-9
11-0050 i2c-0   i2c-10  i2c-2   i2c-4  i2c-6  i2c-8  i2c-FTE1001:00
```

```
kaiden@debian:/sys/bus/i2c/drivers/eeprom$ ls
0-0052  11-0070  4-0050  bind  module  uevent  unbind
kaiden@debian:/sys/bus/i2c/drivers/eeprom$
```

Linux I2C – I2C-Stub Driver

- I2C driver testing with i2c-stub adapter
  - with chip_addr=0x70, device won't be detected after modprobe eeprom driver -> file "new_device" comes to the rescue

```
Terminal - kaiden@debian: /sys/bus/i2c/drivers/eeprom
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:/sys/bus/i2c/devices$ sudo modprobe i2c-stub chip_addr=0x70
kaiden@debian:/sys/bus/i2c/devices$ sudo modprobe eeprom
kaiden@debian:/sys/bus/i2c/devices$ ls
0-0052  i2c-0  i2c-10  i2c-2  i2c-4  i2c-6  i2c-8  i2c-FTE1001:00
4-0050  i2c-1  i2c-11  i2c-3  i2c-5  i2c-7  i2c-9

kaiden@debian:/sys/bus/i2c/drivers/eeprom$ ls
0-0052  4-0050  bind  module  uevent  unbind

kaiden@debian:/sys/bus/i2c/devices/i2c-11$ echo eeprom 0x70 > new_device
kaiden@debian:/sys/bus/i2c/devices/i2c-11$ cd ..
kaiden@debian:/sys/bus/i2c/devices$ cd ..
kaiden@debian:/sys/bus/i2c$ ls
devices  drivers  drivers_autoprobe  drivers_probe  uevent
kaiden@debian:/sys/bus/i2c$ cd devices/
kaiden@debian:/sys/bus/i2c/devices$ ls
0-0052  4-0050  i2c-1  i2c-11  i2c-3  i2c-5  i2c-7  i2c-9
11-0070  i2c-0  i2c-10  i2c-2  i2c-4  i2c-6  i2c-8  i2c-FTE1001:00

kaiden@debian:/sys/bus/i2c/drivers/eeprom$ ls
0-0052  11-0070  4-0050  bind  module  uevent  unbind
kaiden@debian:/sys/bus/i2c/drivers/eeprom$
```

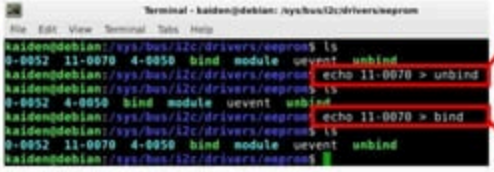
Linux I2C – I2C-Stub Driver

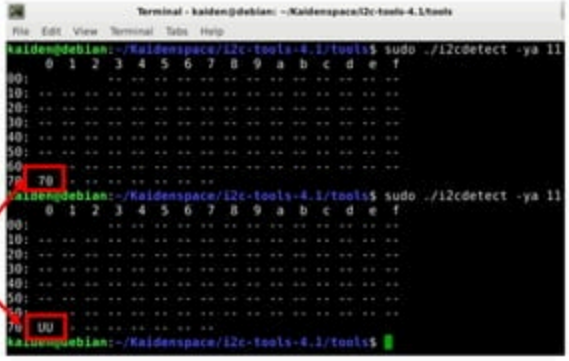
- I2C driver testing with i2c-stub adapter
  - devices：new_device/delete_device

```
Terminal - kaiden@debian: /sys/bus/i2c/devices/i2c-11
File  Edit  View  Terminal  Tabs  Help
kaiden@debian:/sys/bus/i2c/devices/i2c-11$ ls
11-0070  delete_device  i2c-dev  name  new_device  power  subsystem  uevent
kaiden@debian:/sys/bus/i2c/devices/i2c-11$ echo 0x70 > delete_device
kaiden@debian:/sys/bus/i2c/devices/i2c-11$ ls
delete_device  i2c-dev  name  new_device  power  subsystem  uevent
kaiden@debian:/sys/bus/i2c/devices/i2c-11$ echo eeprom 0x70 > new_device
kaiden@debian:/sys/bus/i2c/devices/i2c-11$ ls
11-0070  delete_device  i2c-dev  name  new_device  power  subsystem  uevent
kaiden@debian:/sys/bus/i2c/devices/i2c-11$
```

Linux I2C – I2C-Stub Driver

- I2C driver testing with i2c-stub adapter
  - drivers：bind/unbind





- Introduction to I2C
  - Basics
  - I2C/SMBus Protocol
- Linux I2C
  - Linux I2C Subsystem
  - I2C Bus Driver
  - I2C Adapter Driver
  - I2C-Dev Driver
  - I2C-Tools
  - I2C-Stub Driver
  - I2C Client Driver

Linux I2C – I2C Client Driver

- Client driver (/drivers/mfd/tps65086.c)
  - tps65086_probe()
  - tps65086_remove()
  - struct i2c_driver tps65086_driver
  - module_i2c_driver(tps65086_driver)

Linux I2C – I2C Client Driver

- Probe / remove

```
static int tps65086_probe(struct i2c_client *client,
                          const struct i2c_device_id *ids)
{
        struct tps65086 *tps;
        unsigned int version;
        int ret;

        tps = devm_kzalloc(&client->dev, sizeof(*tps), GFP_KERNEL);
        if (!tps)
                return -ENOMEM;

        i2c_set_clientdata(client, tps);
        tps->dev = &client->dev;
        tps->irq = client->irq;

        tps->regmap = devm_regmap_init_i2c(client, &tps65086_regmap_config);
        ...
        ret = regmap_read(tps->regmap, TPS65086_DEVICEID, &version);
        ...
        ret = regmap_add_irq_chip(tps->regmap, tps->irq, IRQF_ONESHOT, 0,
                                  &tps65086_irq_chip, &tps->irq_data);
        ...
        ret = mfd_add_devices(tps->dev, PLATFORM_DEVID_AUTO, tps65086_cells,
                              ARRAY_SIZE(tps65086_cells), NULL, 0,
                              regmap_irq_get_domain(tps->irq_data));
```

```
static int tps65086_remove(struct i2c_client *client)
{
        struct tps65086 *tps = i2c_get_clientdata(client);

        regmap_del_irq_chip(tps->irq, tps->irq_data);

        return 0;
}
```

Linux I2C – I2C Client Driver

- i2c_get_clientdata() / i2c_set_clientdata()

```
static inline void *i2c_get_clientdata(const struct i2c_client *dev)
{
        return dev_get_drvdata(&dev->dev);
}
```

```
static inline void *dev_get_drvdata(const struct device *dev)
{
        return dev->driver_data;
}
```

```
static inline void i2c_set_clientdata(struct i2c_client *dev, void *data)
{
        dev_set_drvdata(&dev->dev, data);
}
```

```
static inline void dev_set_drvdata(struct device *dev, void *data)
{
        dev->driver_data = data;
}
```

Linux I2C – I2C Client Driver

- devmem_regmap_init_i2c()
  - __regmap_lockdep_wrapper()
    - __regmap_init_i2c()
      - regmap_get_i2c_bus()
        - i2c_check_functionality()
        - return
      - __regmap_init()

&regmap_i2c
&regmap_i2c_smbus_i2c_block
&regmap_smbus_word_swapped
&regmap_smbus_word
&regmap_smbus_byte

```
static const struct regmap_bus *regmap_get_i2c_bus(struct i2c_client *i2c,
                                    const struct regmap_config *config)
{
        if (i2c_check_functionality(i2c->adapter, I2C_FUNC_I2C))
                return &regmap_i2c;
        else if (config->val_bits == 8 && config->reg_bits == 8 &&
                 i2c_check_functionality(i2c->adapter,
                                         I2C_FUNC_SMBUS_I2C_BLOCK))
                return &regmap_i2c_smbus_i2c_block;
        else if (config->val_bits == 16 && config->reg_bits == 8 &&
                 i2c_check_functionality(i2c->adapter,
                                         I2C_FUNC_SMBUS_WORD_DATA))
                switch (regmap_get_val_endian(&i2c->dev, NULL, config)) {
                case REGMAP_ENDIAN_LITTLE:
                        return &regmap_smbus_word;
                case REGMAP_ENDIAN_BIG:
                        return &regmap_smbus_word_swapped;
                default:                /* everything else is not supported */
                        break;
                }
        else if (config->val_bits == 8 && config->reg_bits == 8 &&
                 i2c_check_functionality(i2c->adapter,
                                         I2C_FUNC_SMBUS_BYTE_DATA))
                return &regmap_smbus_byte;

        return ERR_PTR(-ENOTSUPP);
}
```

Linux I2C – I2C Client Driver

- regmap_i2c

```c
static struct regmap_bus regmap_i2c = {
        .write = regmap_i2c_write,
        .gather_write = regmap_i2c_gather_write,
        .read = regmap_i2c_read,
        .reg_format_endian_default = REGMAP_ENDIAN_BIG,
        .val_format_endian_default = REGMAP_ENDIAN_BIG,
};
```

```c
static int regmap_i2c_read(void *context,
                           const void *reg, size_t reg_size,
                           void *val, size_t val_size)
{
        struct device *dev = context;
        struct i2c_client *i2c = to_i2c_client(dev);
        struct i2c_msg xfer[2];
        int ret;

        xfer[0].addr = i2c->addr;
        xfer[0].flags = 0;
        xfer[0].len = reg_size;
        xfer[0].buf = (void *)reg;

        xfer[1].addr = i2c->addr;
        xfer[1].flags = I2C_M_RD;
        xfer[1].len = val_size;
        xfer[1].buf = val;

        ret = i2c_transfer(i2c->adapter, xfer, 2);
        if (ret == 2)
                return 0;
        else if (ret < 0)
                return ret;
        else
                return -EIO;
}
```

Linux I2C – I2C Client Driver

- mfd_add_devices()
  - mfd_add_device()
    - platform_device_add()
      - device_add()

Linux I2C – I2C Client Driver

- Matching

/arch/arm64/boot/dts/xilinx/zynqmp-zcu100-recvC.dts

```
static const struct of_device_id tps65086_of_match_table[] = {
        { .compatible = "ti,tps65086", },
        { /* sentinel */ }
};
MODULE_DEVICE_TABLE(of, tps65086_of_match_table);
```

```
i2csw_4: i2c@4 {
        #address-cells = <1>;
        #size-cells = <0>;
        reg = <0x4>;

        pmic: pmic@5e { /* Custom TI PMIC u33 */
                compatible = "ti,tps65086";
                reg = <0x5e>;
                interrupt-parent = <&gpio>;
                interrupts = <77 GPIO_ACTIVE_LOW>;
                #gpio-cells = <2>;
                gpio-controller;
        };
};
```

Linux I2C – I2C Client Driver

- Device driver structure

```
static struct i2c_driver tps65086_driver = {
        .driver         = {
                .name   = "tps65086",
                .of_match_table = tps65086_of_match_table,
        },
        .probe          = tps65086_probe,
        .remove         = tps65086_remove,
        .id_table       = tps65086_id_table,
};
module_i2c_driver(tps65086_driver);
```

Linux I2C – I2C Client Driver

- Driver registration/unregistration

```
#define module_i2c_driver(__i2c_driver) \
        module_driver(__i2c_driver, i2c_add_driver, \
                      i2c_del_driver)
```

```
#define module_driver(__driver, __register, __unregister, ...) \
static int __init __driver##_init(void) \
{ \
        return __register(&(__driver) , ##__VA_ARGS__); \
} \
module_init(__driver##_init); \
static void __exit __driver##_exit(void) \
{ \
        __unregister(&(__driver) , ##__VA_ARGS__); \
} \
module_exit(__driver##_exit);
```

```
static int __init tps65086_driver_init(void)
{
        return i2c_add_driver(&(tps65086_driver));
}
module_init(tps65086_driver_init);
static void __exit tps65086_driver_exit(void)
{
        i2c_del_driver(&(tps65086_driver));
}
module_exit(tps65086_driver_exit);;
```

Linux I2C – I2C Client Driver

- i2c_add_driver() / i2c_del_driver()

```
#define i2c_add_driver(driver) \
        i2c_register_driver(THIS_MODULE, driver)
```

```
int i2c_register_driver(struct module *owner, struct i2c_driver *driver)
{
        int res;

        /* Can't register until after driver model init */
        if (WARN_ON(!is_registered))
                return -EAGAIN;

        /* add the driver to the list of i2c drivers in the driver core */
        driver->driver.owner = owner;
        driver->driver.bus = &i2c_bus_type;
        INIT_LIST_HEAD(&driver->clients);

        /* When registration returns, the driver core
         * will have called probe() for all matching-but-unbound devices.
         */
        res = driver_register(&driver->driver);
        if (res)
                return res;

        pr_debug("driver [%s] registered\n", driver->driver.name);

        /* ask the adapters that may already exist */
        i2c_for_each_dev(driver, __process_new_driver);

        return 0;
}
EXPORT_SYMBOL(i2c_register_driver);
```

```
void i2c_del_driver(struct i2c_driver *driver)
{
        i2c_for_each_dev(driver, __process_removed_driver);

        driver_unregister(&driver->driver);
        pr_debug("driver [%s] unregistered\n", driver->driver.name);
}
EXPORT_SYMBOL(i2c_del_driver);
```

Linux I2C – I2C Client Driver

- driver_register()
  - bus_add_driver()
    - driver_attach()
      - __driver_attach()
        - driver_probe_device()
          - really_probe()

```
static int really_probe(struct device *dev, struct device_driver *drv)
{
    ...
    if (dev->bus->probe) {
            ret = dev->bus->probe(dev);
            if (ret)
                    goto probe_failed;
    } else if (drv->probe) {
            ret = drv->probe(dev);
            if (ret)
                    goto probe_failed;
    }
```

```
struct bus_type i2c_bus_type = {
        .name           = "i2c",
        .match          = i2c_device_match,
        .probe          = i2c_device_probe,
        .remove         = i2c_device_remove,
        .shutdown       = i2c_device_shutdown,
};
EXPORT_SYMBOL_GPL(i2c_bus_type);
```

probe is assigned in i2_bus_type
so bus->probe will be called

Linux I2C – I2C Client Driver

- driver_unregister()
  - Bus_remove_driver()
    - Driver_detach()
      - Device_release_driver_internal()
        - __device_release_driver()

```
static void __device_release_driver(struct device *dev, struct device *parent)
{
        struct device_driver *drv;
        ...
        if (dev->bus && dev->bus->remove)
                dev->bus->remove(dev);
        else if (drv->remove)
                drv->remove(dev);
```

```
struct bus_type i2c_bus_type = {
        .name           = "i2c",
        .match          = i2c_device_match,
        .probe          = i2c_device_probe,
        .remove         = i2c_device_remove,
        .shutdown       = i2c_device_shutdown,
};
EXPORT_SYMBOL_GPL(i2c_bus_type);
```

Linux I2C – I2C Client Driver

- i2c_device_probe() / i2c_device_remove()

```
static int i2c_device_probe(struct device *dev)
{
        struct i2c_client       *client = i2c_verify_client(dev);
        struct i2c_driver       *driver;
        int status;
        ...

        if (!driver->id_table &&
            !i2c_acpi_match_device(dev->driver->acpi_match_table, client) &&
            !i2c_of_match_device(dev->driver->of_match_table, client))
                return -ENODEV;

        ...

        if (driver->probe_new)
                status = driver->probe_new(client);
        else if (driver->probe)
                status = driver->probe(client,
                                       i2c_match_id(driver->id_table, client));
        else
                status = -EINVAL;

        if (status)
                goto err_detach_pm_domain;
```

```
static int i2c_device_remove(struct device *dev)
{
        struct i2c_client       *client = i2c_verify_client(dev);
        struct i2c_driver       *driver;
        int status = 0;

        if (!client || !dev->driver)
                return 0;

        driver = to_i2c_driver(dev->driver);
        if (driver->remove) {
                dev_dbg(dev, "remove\n");
                status = driver->remove(client);
        }

        dev_pm_domain_detach(&client->dev, true);

        dev_pm_clear_wake_irq(&client->dev);
        device_init_wakeup(&client->dev, false);

        return status;
}
```

Linux I2C – I2C Client Driver

- __process_new_driver()
  - i2c_do_add_adapter()
    - i2c_detect()
      - i2c_detect_address()
        - i2c_default_probe()
          - i2c_check_functionality()
          - i2c_smbus_xfer()
        - driver->detect()
        - i2c_new_device()
          - device_register(&client->dev)
            - device_add()
    - driver->attach_adapter()

- __process_removed_driver()
  - i2c_do_del_adapter()
    - i2c_unregister_device()
      - device_unregister(&client->dev)
        - device_del()

Thank you for your patience