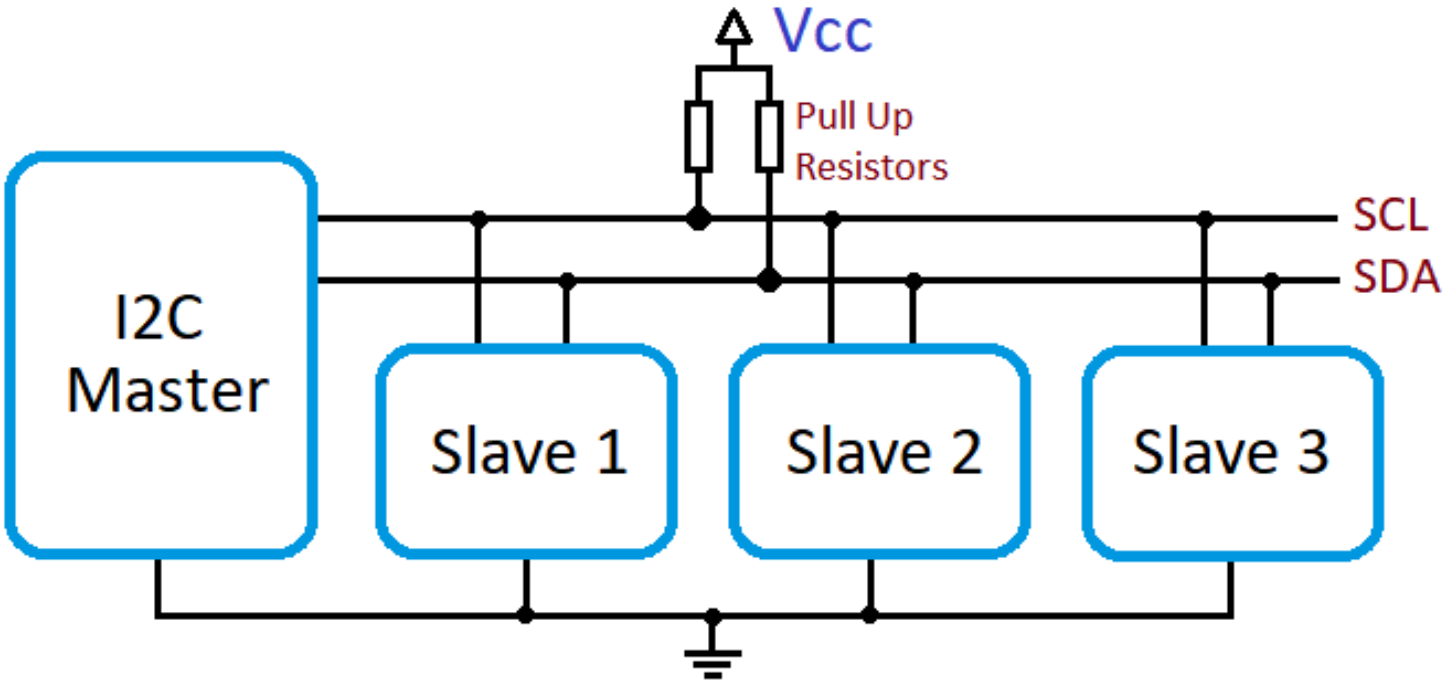
Module 10

I2C interface

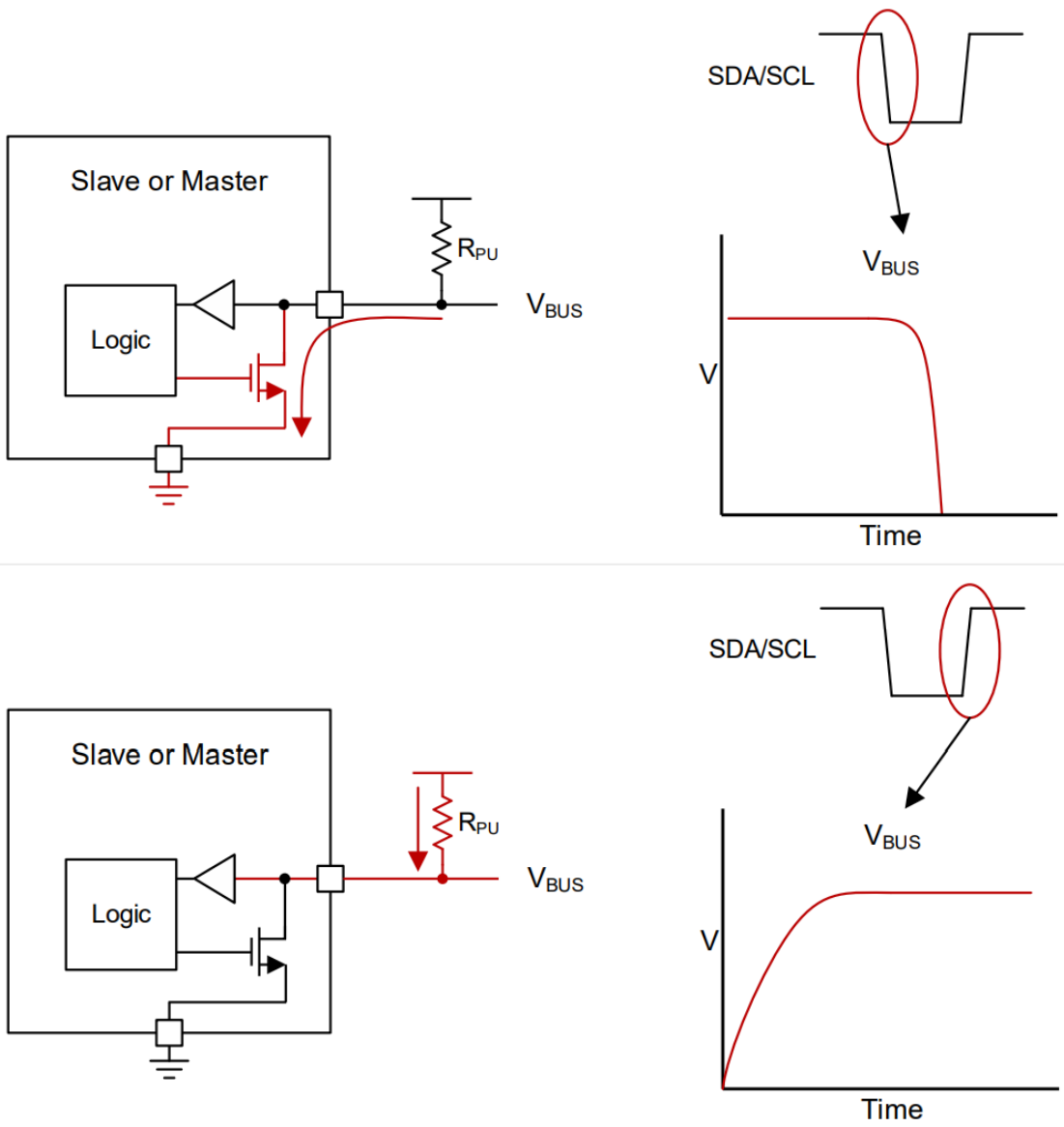
I2C features

- I2C (*Inter Integrated Circuit*) is two wire interface for connecting a microcontroller with peripherals
- The interface uses two bidirectional open-drain lines:
  - Serial Data Line (SDA) and
  - Serial Clock Line (SCL)
- Multi-master and multi-slave interface
- As it is open-drain structure, the use of pull-up resistors is mandatory
- Up to 5 Mbps bitrates (typically: 100 kbps, 400 kbps, 1 Mbps, 3.4 Mbps) depending on number od devices
- Up to few meters cable length
- Logic high voltage level depends on power supply

I2C topology



I2C internals

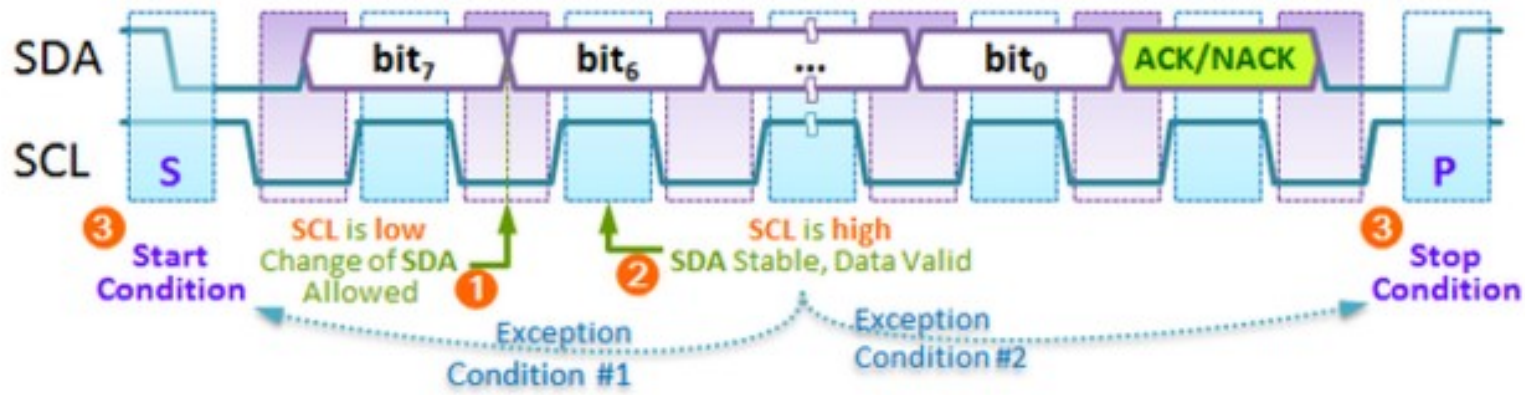


I2C nodes

- *Master node*: generates the clock and initiate and control all communication (typically microcontroller)
- *Slave node*: receives the clock and responds when addressed by master (typically peripherals)
- Each slave has a unique address (7-bit or 10-bits)
- Master and slave can switch roles
- Multi-master interface $\rightarrow$ bus contention and arbitration mechanism must exists

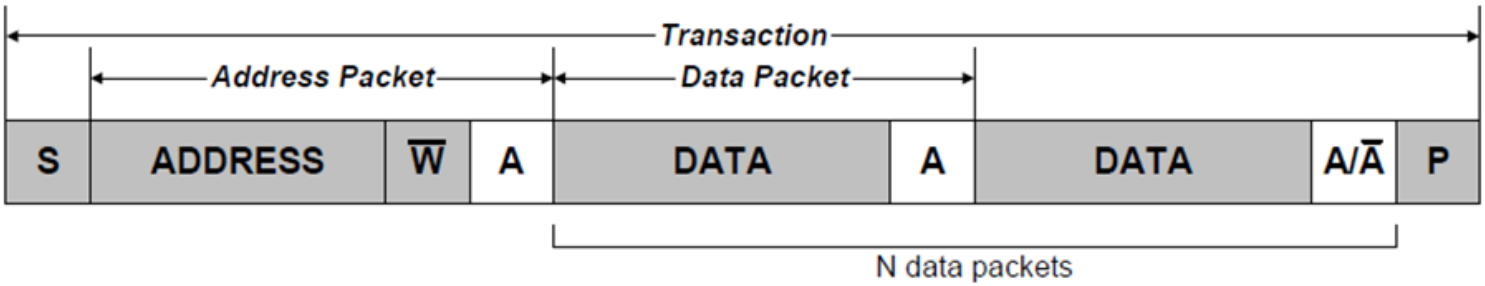
I2C transation

- The transaction is initiated by sending *start sequence*
- The transaction ends with the *stop sequence*
- Each word is acknowledged with either positive (ACK or low) or negative (NACK or high) signal
- Optionally, master regenerate the start sequence (aka *repeated start*) in the middle of the transaction to change the access direction



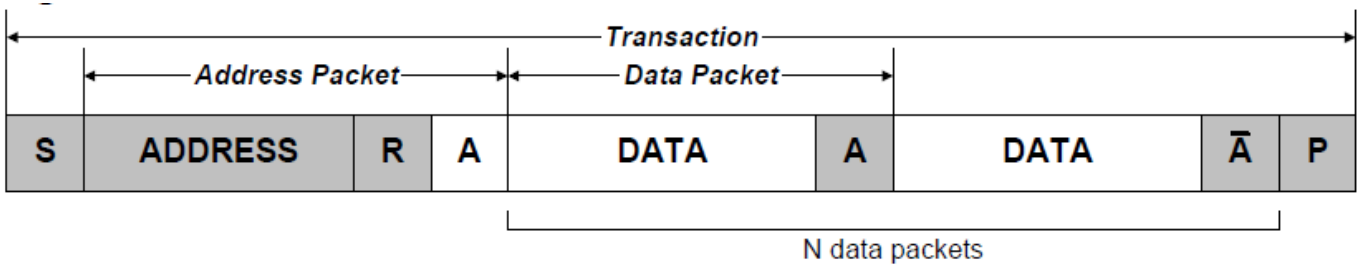
I2C operations

Write operation



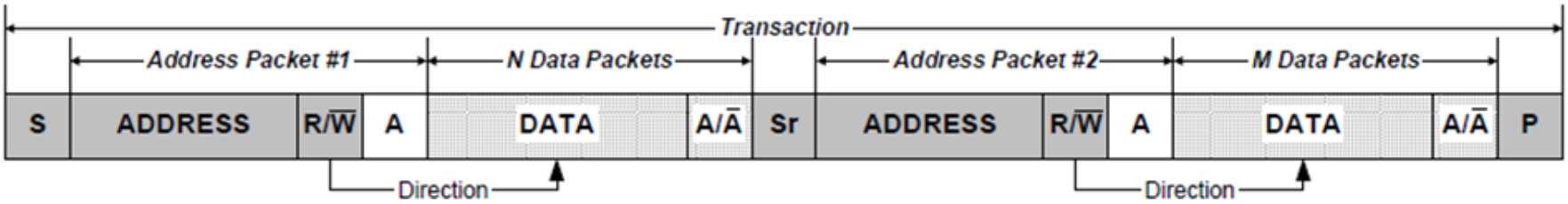
I2C operations

Read operation



I2C operations

Combined operation



Linux i2cdev API

I2C device in Linux can be accessed from userspace via `/dev/i2c-N`:

- `open()` to open an I2C device
- Using `read()` and `write()` access, the peripheral is accessed using two I2C transactions (with stop sequence between them)
- Combined read/write transaction can be achieved using `ioctl()` requests and `struct i2c_rdwr_ioctl_data` structure with the following fields:
  - `msgs`: pointer to an array of structures `struct i2c_msg` describing I2C messages
  - `nmsgs`: number of messages to exchange
- `struct i2c_msg` has the following fields:
  - `addr`: slave address
  - `flags`: various flags that define how transaction will be handled
  - `len`: message length
  - `buf`: pointer to message data
- `close()` to close the I2C device when done

Opening and closing an I2C device

- Opening:

```
fd = open("/dev/i2c-0", O_RDWR);
if (fd == -1)
     printf("Failed to open I2C device.\n");
```

- Closing:

```
close(fd);
```

More information can be found on
https://www.kernel.org/doc/Documentation/i2c/dev-interface

Example reading/writing

```
int addr = 0x40;
unsigned char reg = 0x10;
unsigned char rx_buffer[10];
unsigned char tx_buffer[10];
...
// configure the slave address
ioctl(fd, I2C_SLAVE, addr)

for (;;)
{
    tx_buffer[0] = reg;
    write(fd, tx_buffer, 1);
    read(fd, rx_buffer, 1);
}
```

Combined reading/writing (ioctl way)

```
struct i2c_msg iomsgs[] = {
    [0] = {
        .addr = addr,
        .flags = 0,
        .buf = &reg,
        .len = 1
    },
    [1] = {
        .addr = addr,
        .flags = I2C_M_RD,
        .buf = rx_buffer,
        .len = 1
    },
};
```

```
struct i2c_rdwr_ioctl_data
  msgset = {
    .msgs = iomsgs,
    .nmsgs = 2
};
...
for (;;)
{
    ioctl(fd, I2C_RDWR,
      &msgset)
}
```