Chapter 14

I2C

I2C

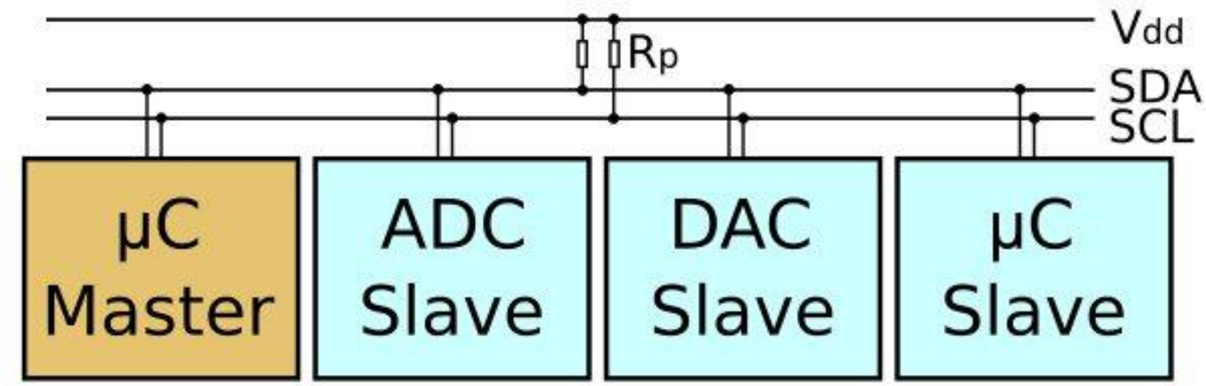
- Previously we were working with serial communication protocol which is a widely used protocol because of its very simple
- This simplicity makes it easy to implement on top of other protocols like bluetooth and USB
- However this simplicity brings some downsides
- data exchanges, like reading a digital sensor, would require the sensor vendor to come up with another protocol on top of it
- Luckily for us, there are plenty of other communication protocols in the embedded space.

I2C

- Our F3 board has **3 motion sensors** (i.e. accelerometer, magnetometer and gyroscope)
- The accelerometer and magnetometer are **packaged in a single component** and can be accessed using an I2C bus.
- I2C stands for Inter-Integrated Circuit and it is a **synchronous serial communication protocol**
- This protocol uses two lines to exchange data
    - SDA : A data line
    - SCL : A clock line

I2C

- Protocol is synchronous because a clock line is used to synchronize the communication



- As you can see this protocol uses **master slave** model

I2C

- Master is the device that starts and derive the communication with the slave
- Several devices, both masters and slaves can be connected to the same bus at the same time
- Master can communicate with specific slave by broadcasting it's address (address can be 7 or 10 bit long)
- Once master started communication, no other device can make use of bus until master stops communication

I2C

- SCL or clock line determines how fast data can be exchanged
- Usually operates at 100 KHz (standard mode) or 400 KHz (fast mode)

General Protocol

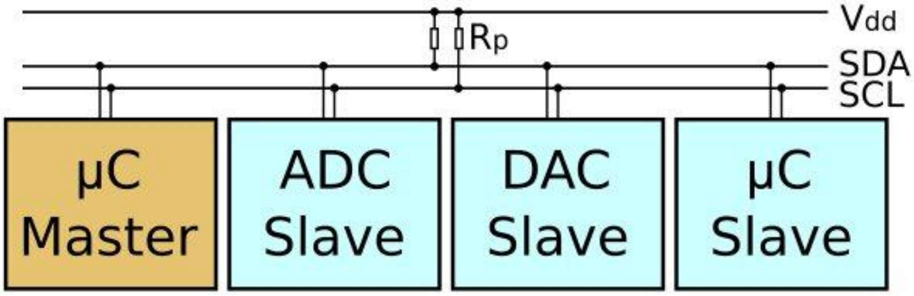
General Protocol

- Using I2C protocol we can communicate with multiple devices unlike serial communication protocol where we cannot.
- Now let see how this communication takes place
- Two way communication is possible in this protocol
- Master -> Slave (Where master writes to slave)
- Master <- Slave (Where master reads from slave)

Master -> Slave

When master wants to send data to slave

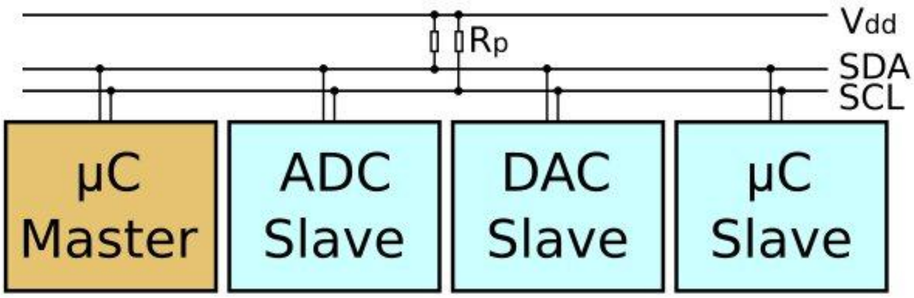
1. Master: Broadcast START
2. M: Broadcast slave address(7 bits)
   + the R/W (8th) bit set to **WRITE**
3. Slave: Responds **ACK**
   (ACKnowledgement)
4. M: Send one byte
5. S: Responds ACK
6. Repeat steps 4 and 5 zero or more times
7. M: Broadcast STOP OR (broadcast RESTART and go back to (2))



Master <- Slave

When master wants to read data from slave

1. M: Broadcast START
2. M: Broadcast slave address(7 bits)
   + the R/W (8th) bit set to **READ**
3. Slave: Responds ACK
   (ACKnowledgement)
4. S: Send one byte
5. M: Responds ACK
6. Repeat steps 4 and 5 zero or more times
7. M: Broadcast STOP OR (broadcast RESTART and go back to (2))



LSM303DLHC

LSM303DLHC

- Two sensors, the magnetometer and the accelerometer, are packaged in this chip
- These sensors can be accessed via an I2C bus
- Each sensor behaves like an I2C slave and has a different address
- Each sensor has its own memory where it stores the results of sensing its environment
- Our interaction with these sensors will mainly involve reading their memory

LSM303DLHC

- Memory is modeled as byte addressable registers of these sensors
- Sensors can be configured by writing to their registers
- This way, these sensors are similar to the peripherals inside the uC
- Difference is that their registers are not mapped into the uCs' memory. Instead, their registers have to be accessed via the I2C bus

LSM303DLHC

- After all this theory let's try reading some data from these sensors
- Following are the registers and fields will be used in reading/writing from/to sensors:
  - CR2: SADD1, RD_WRN, NBYTES, START, AUTOEND
  - ISR: TXIS, RXNE, TC
  - TXDR: TXDATA
  - RXDR: RXDATA

Starter code can be viewed in the source repository

Summary