MUNI FI

PV198 - SPI II

One-chip Controllers

Daniel Dlhopolček, Marek Vrbka, Jan Koniarik, Oldřich Pecák, Tomáš Rohlínek, Ján Labuda, Jan Horáček, Matúš Škvarla, Ondřej Bleha, Martin Klimeš, Adam Valt

Faculty of Informatics, Masaryk University

9/2024

Intro

- Switch the branch to *Week_09*!
- Discussion of HW8

Context

- In some cases it is necessary to serve not to rule.
- We will be implementing a simple SPI slave device.
- When request is received from the master, send data back:
  - Reading the data from sensor is slow $\rightarrow$ they have to be prepared in advance.
- We will be using **non-blocking** SPI API.

Requirements

Master can send two types of messages (identified by the initial byte of the message):

1. `send_ADC` (`0x55`): report latest measured ADC value.
2. `receive_UART` (`0xCC`): print 16 subsequent bytes to UART.

Template

Template provides an example of the API to use and basic solution structure.

1. `DSPI_SlaveUserCallback`
   This function is called by the driver after a transfer is finished (similar to interrupt).
2. `DSPI_SlaveTransferNonBlocking`
   This function passes data to the driver, which then transfers it to the master when required.

Raspberry PI pico

We will be using RPI pico as the SPI master.
RPI pico runs *MicroPython* with functions preloaded on the board:

- `print_on_UART()`
- `read_ADC()`
- `read_count()`

Source code in the IS: `main.py`.

Wiring

Make connection in this exact order:

1. Connect pins between K66 and RPI.
2. Connect K66 to PC using USB.
3. Connect RPI to PC using USB.

Wrong order can damage the board!

RPI pico	FRDM-K66P
GND	GND
GP2	PTD1
GP3	PTD3
GP4	PTD2
GP5	PTD0

Counter

Every **response** from slave to master will start with a byte containing *the counter*.

1. Every time you finish either `receive_UART` or `send_ADC` operation, increment the counter.
2. Count is represented by an 8bit unsigned number. It can overflow.
3. Repeated comunication with neither `send_ADC` nor `receive_UART` does not affect the counter.

send_ADC

Read ADC value:

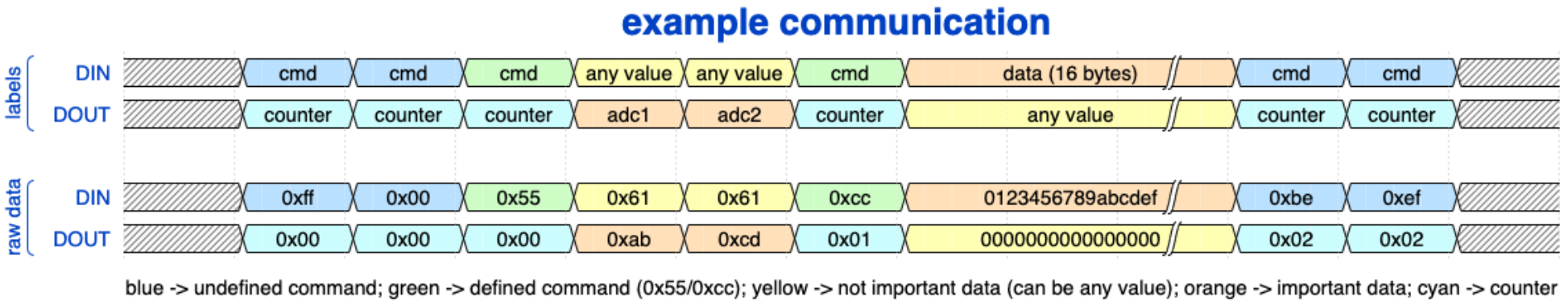
1. Respond with values from ADC1_SE_13.
   - (for inital testing use a constant value)
2. Read command is identified by value `0x55`.
3. Expected ADC value is Big Endian, 12bit long.
4. Prepare ADC value in advance.

receive_UART

Receive 16 bytes over SPI, then send them over UART.

1. Configure PTB11 as `UART3_TX`.
2. Write command is identified by value `0xCC`.
3. You will recieve a chunk of 16 bytes. After you get them all, print your data to UART.
4. Beware: blocking transmit on UART might give you problems in callback.

Example


example communication
labels
raw data
DIN
DOUT
DIN
DOUT
cmd cmd cmd any value any value cmd data (16 bytes) cmd cmd
counter counter counter adc1 adc2 counter any value counter counter
0xff 0x00 0x55 0x61 0x61 0xcc 0123456789abcdef 0xbe 0xef
0x00 0x00 0x00 0xab 0xcd 0x01 0000000000000000 0x02 0x02
blue -> undefined command; green -> defined command (0x55/0xcc); yellow -> not important data (can be any value); orange -> important data; cyan -> counter

Homework

Finish assignment for the lesson.

Scope of lesson is deliberately larger.

Bonus

Recieve chunks of 3 bytes.

1. Configure PTB10 as UART3_RX.
2. Write command is initialized by value `0x37`.
3. You will be asked to return 3 bytes stored, that you recieved by UART3_RX.
4. If there were no data stored, return random data but do not increment counter.
5. Make internal memory 64messages long. If more massages recieved, overwrite the oldest one.

MUNI
FACULTY
OF INFORMATICS