USB I/O EXPANDER

PROGRAMMED WITH THE JAL PROGRAMMING LANGUAGE

CONTENTS

- The project
- Features of the USB I/O Expander
- Protocol
- Example using a terminal emulator
- Basic schematic diagram
- PIC Software
- Support tools
- Demonstration video
- References

THE PROJECT

- Created to control pins and devices from the PC using commands entered as ASCII characters
- No need for programming so easy to use
- Python example files available to make life easier
- Complete documentation on how to use the device
- Open source, see the link in the reference section

FEATURES OF THE USB I/O EXPANDER

- The device behaves like a COM port to which commands and data is transmitted as ASCII characters
- Functions:
  - Reading and writing eight I/O pins
  - IIC interface for controlling IIC devices
  - SPI interface for controlling SPI devices
  - Digital to Analog Conversion (DAC) with 2 selectable analog outputs
  - Analog to Digital Conversion (ADC) with 4 selectable analog inputs
  - Two channel Pulse Width Modulation (PWM) signal generator
- Reset and Ping commands available

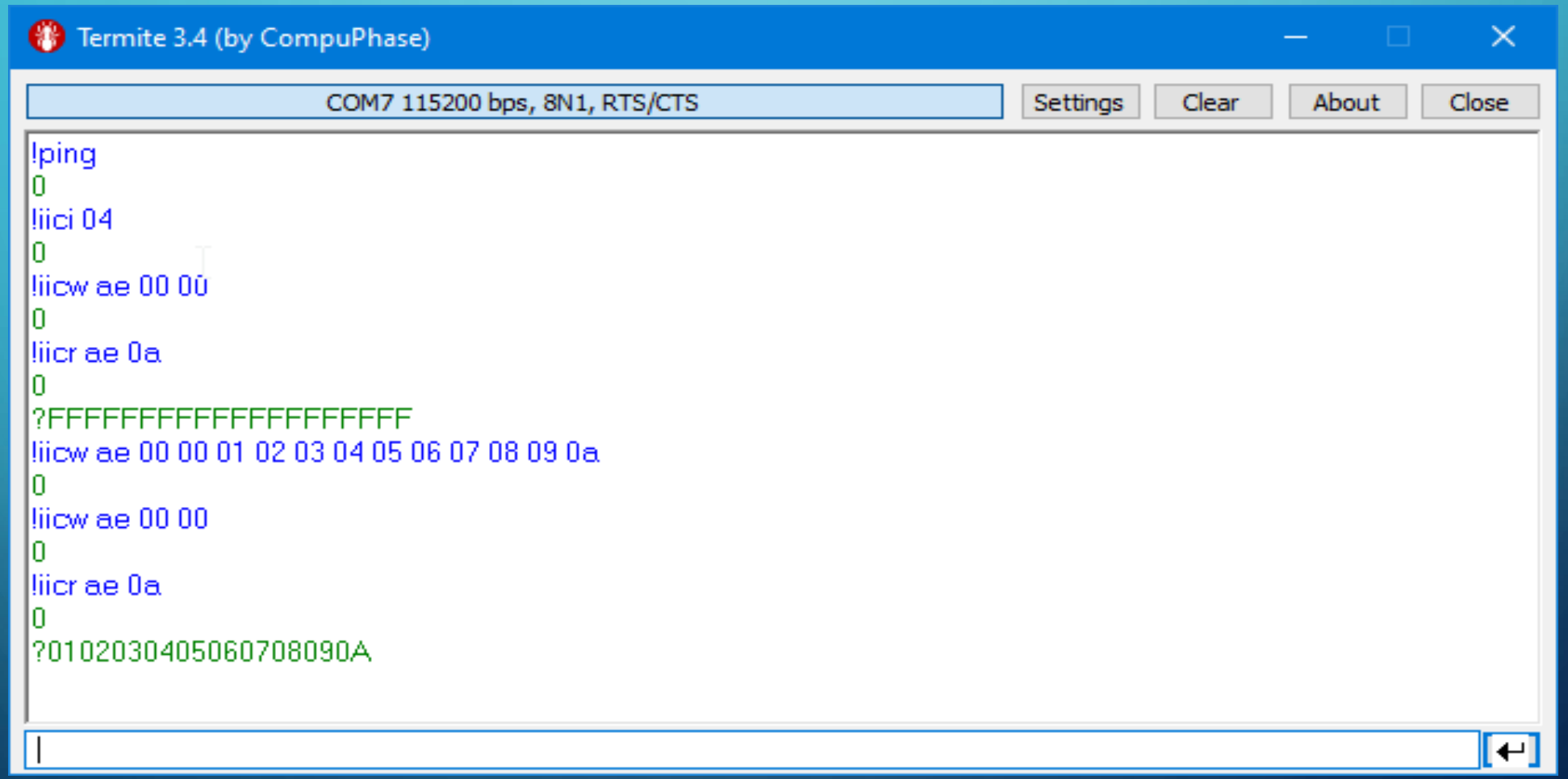
PROTOCOL (1 OF 2)

- A command uses the following syntax: `!<command><data><cr><lf>`
  - All `<data>` must be given in ASCII in hexadecimal format (`00..FF`). Spaces in the data are ignored. Command and data are case insensitive.
- When data is returned, e.g. when reading data via the IIC interface the answer format is: `?<data><cr><lf>`
  - All data is returned in hexadecimal format. No spaces are given between the bytes in the data.

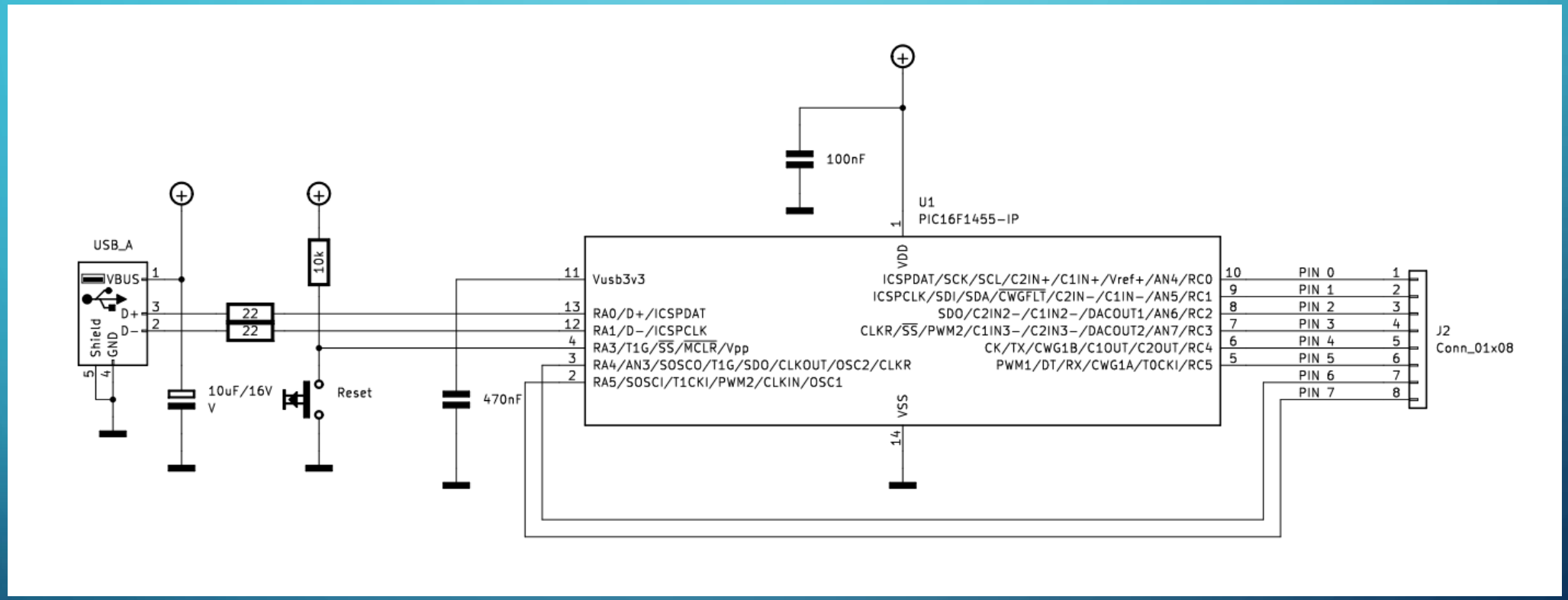
PROTOCOL (2 OF 2)

- After executing a command the following response is given as ASCII character:
  - `0` = Command OK
  - `1` = Error in command or its parameters
  - `2` = Unknown command
- Example:
  - Command: `!IICR AE 0A` (IIC read command reading 10 bytes from IIC address AE)
  - Response: `0` (Command OK)
  - Answer `?0102030405060708090A` (Answer with the 10 bytes read, no spaces)

EXAMPLE USING A TERMINAL EMULATOR



BASIC SCHEMATIC DIAGRAM



PIC SOFTWARE

- The whole program for the PIC16F1455 is written in JAL and takes about 7 Kbytes of ROM and about 400 bytes of RAM
  - The PIC16F1455 has 8192 bytes of ROM and 1024 bytes of RAM
- The main program handles all commands and sends the response. All other functionality is available in separate include files
- JAL libraries are used for most of the provided functionality like IIC, SPI, DAC, ADC and PWM

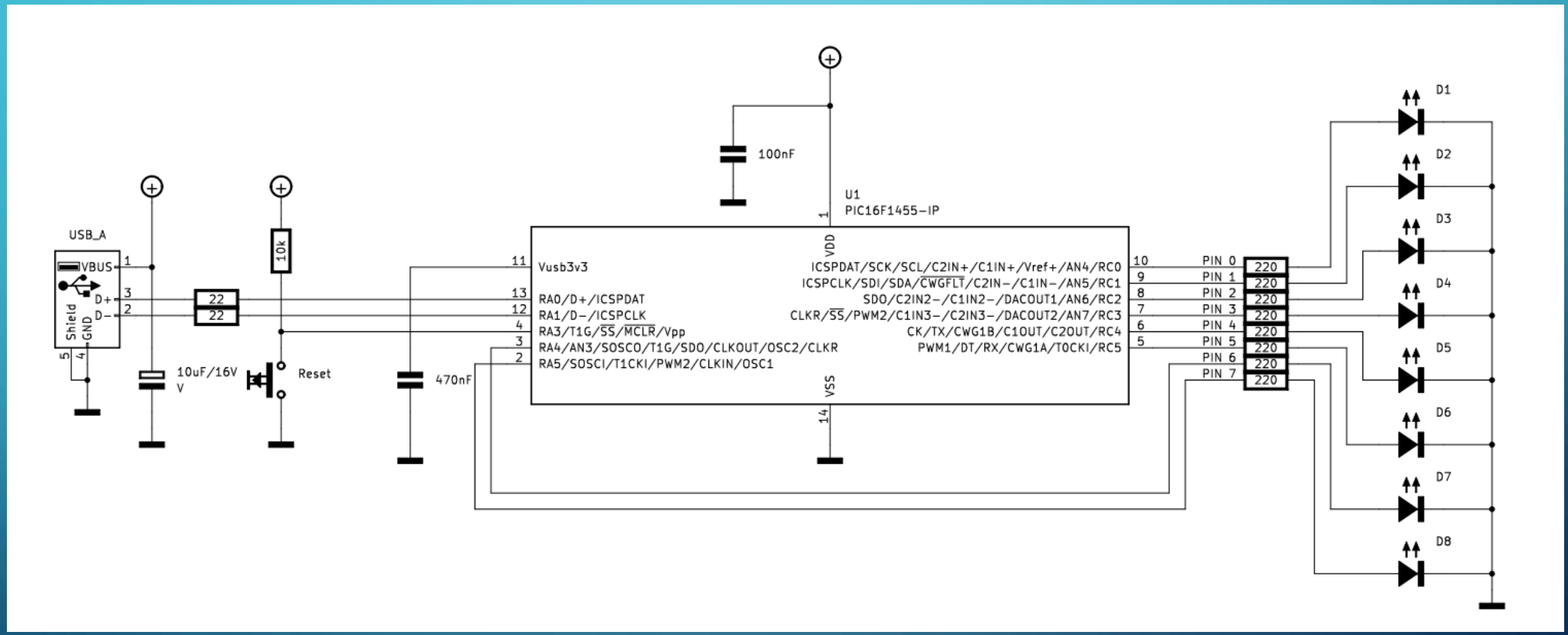
SUPPORT TOOLS

- The device can easily be operated without the need for special tools.
  - A terminal emulation program is sufficient to control the device using ASCII characters
- All commands and responses are described in a separate manual
- A Python library ‘usb_io_expander.py’ is available for ease of use
- Python example files are available for all features

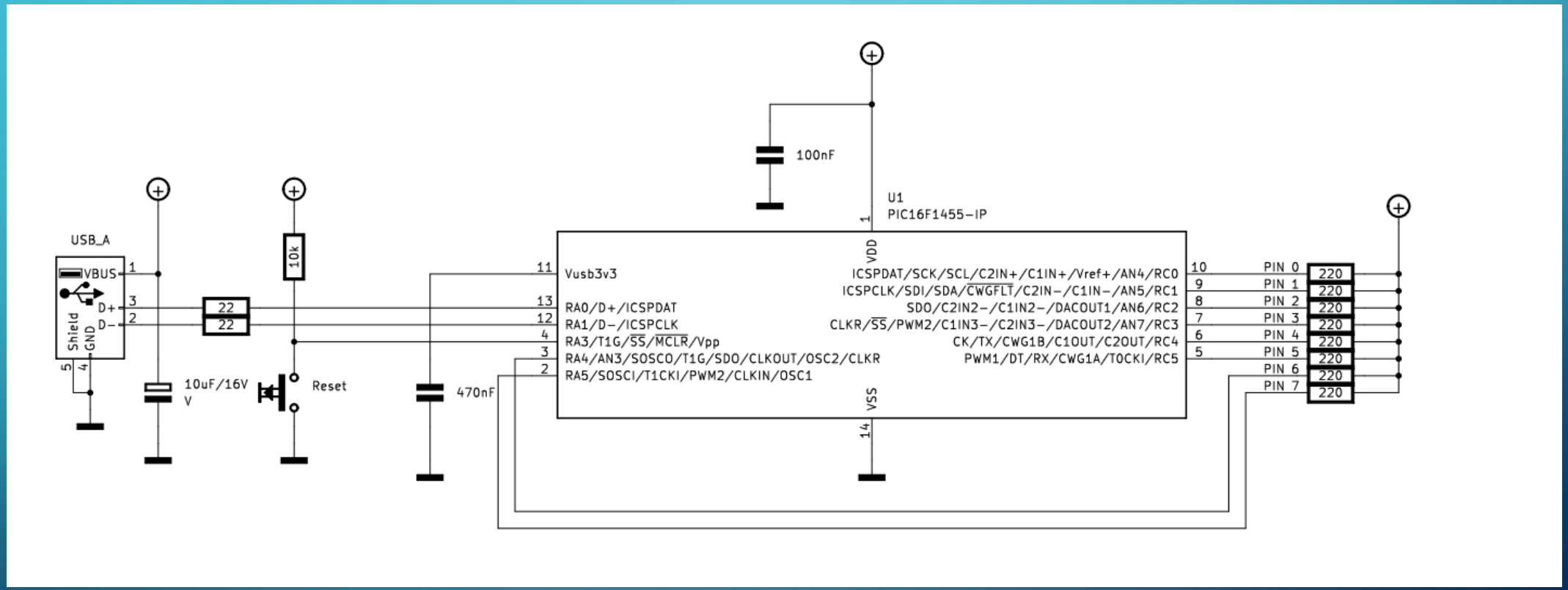
DEMONSTRATION VIDEO

- Each feature is demonstrated using different hardware connected to the device
- The schematic diagram used for the demo is shown before the demo starts
- The Python example files are used for this demonstration

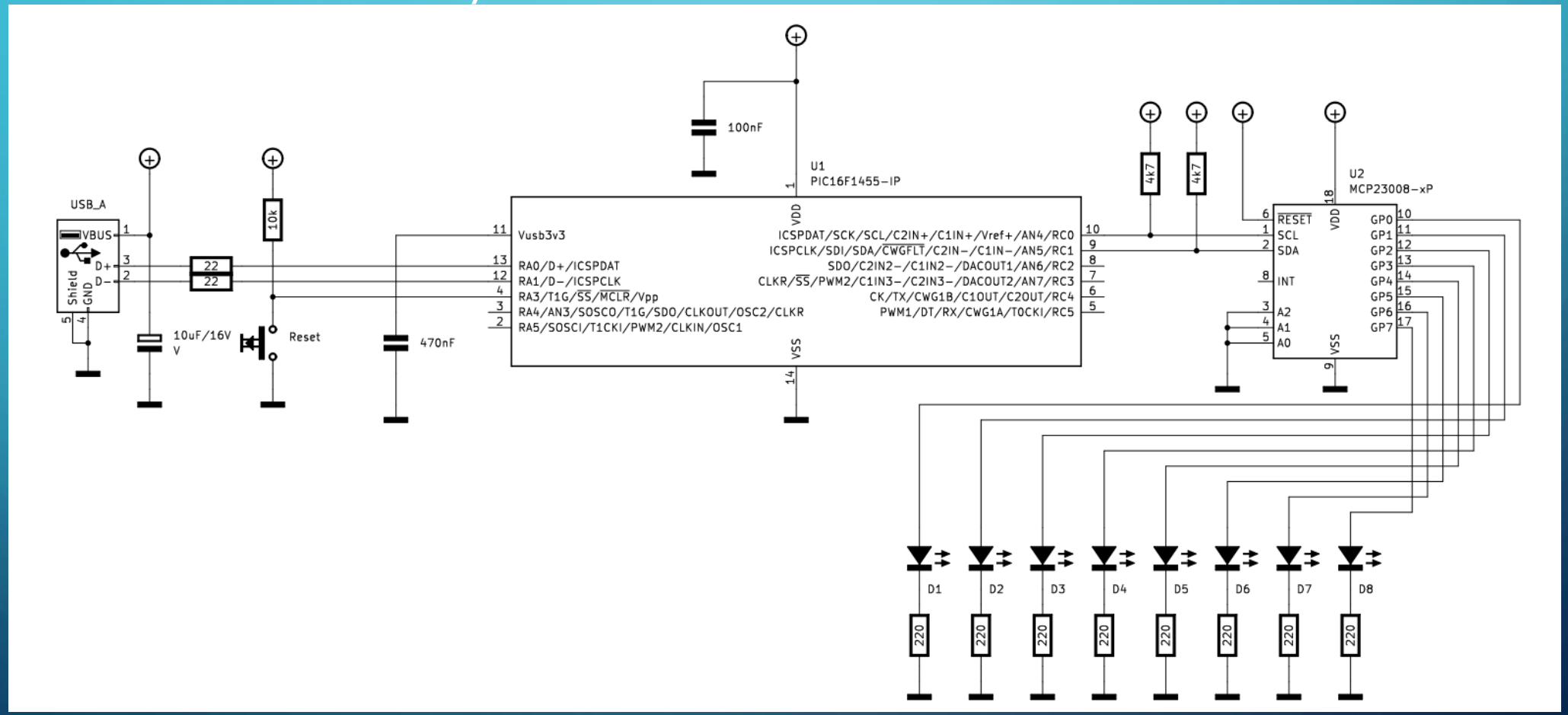
CONTROL THE PINS AS OUTPUT



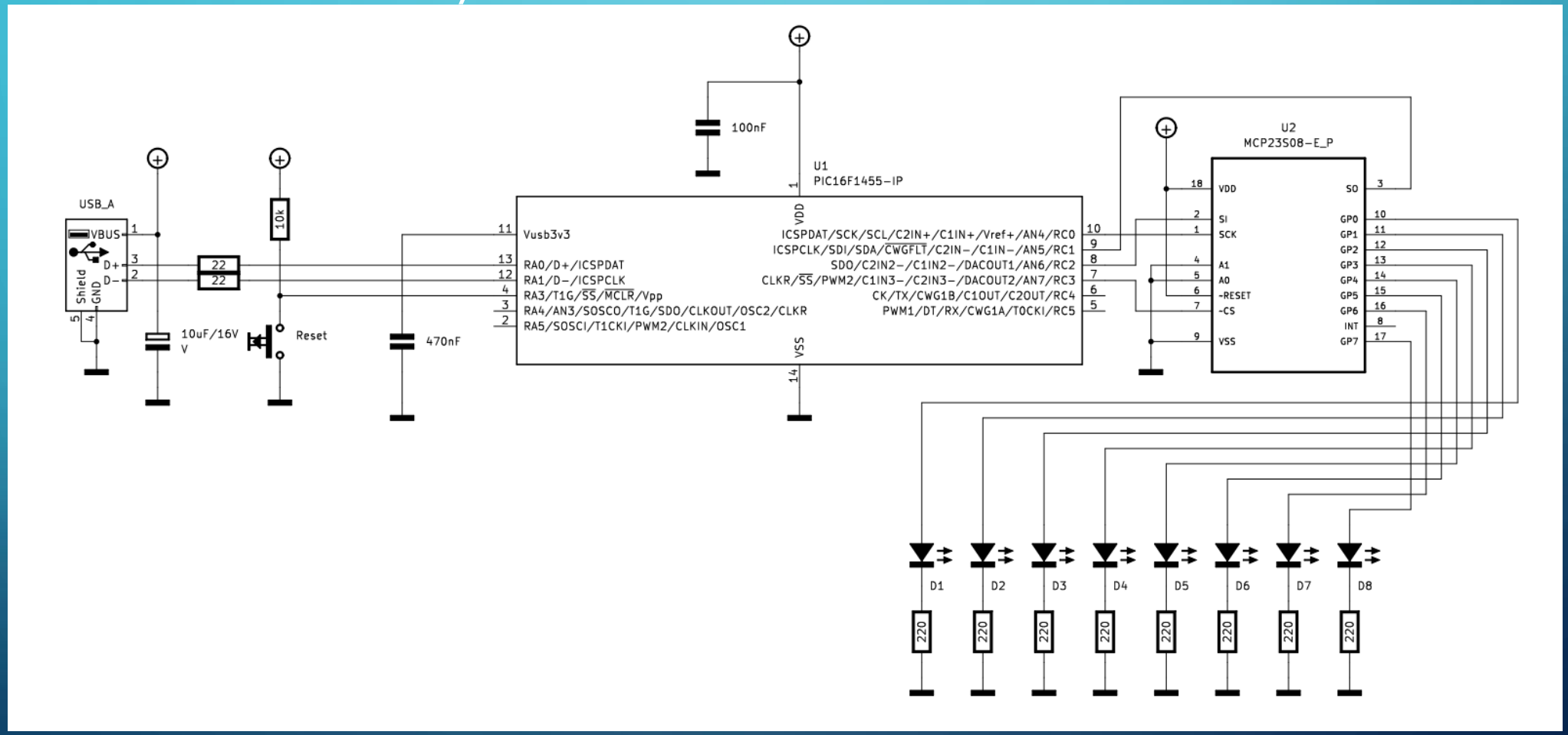
CONTROL THE PINS AS INPUT



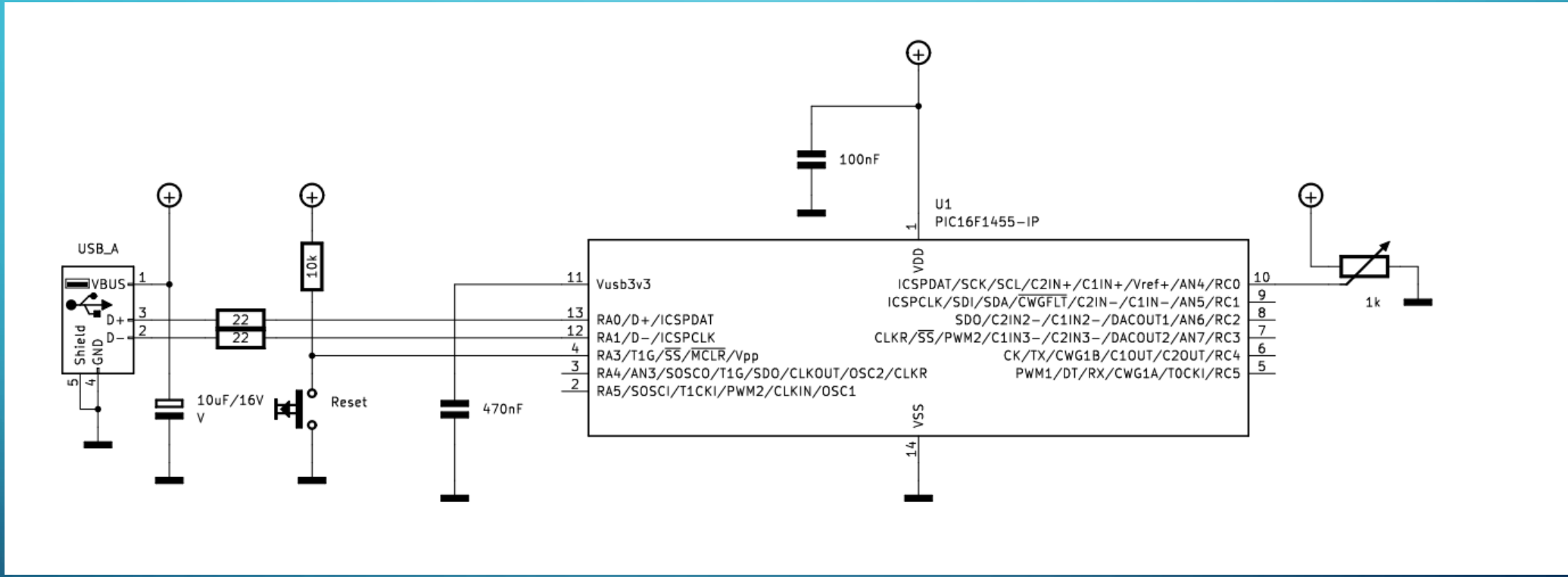
CONTROL AN I/O EXPANDER VIA IIC



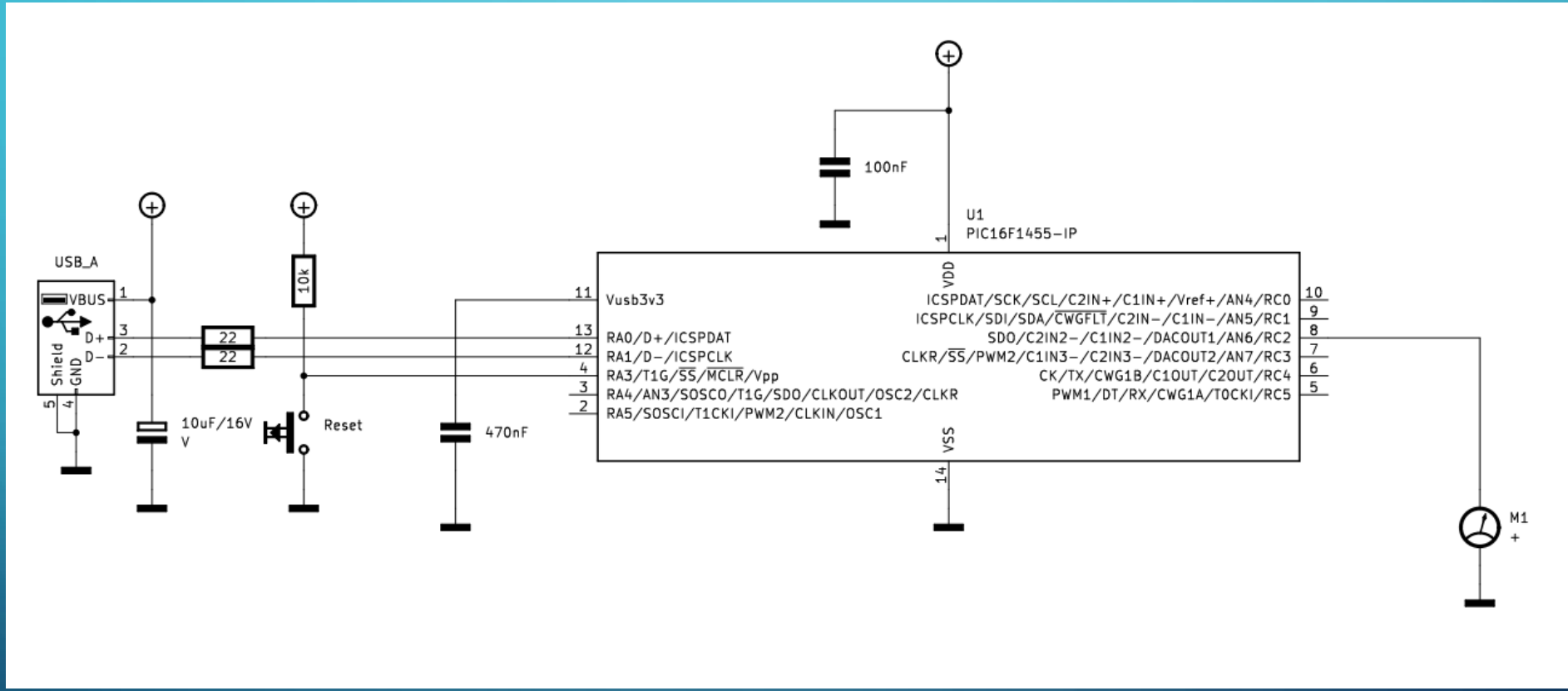
CONTROL AN I/O EXPANDER VIA SPI



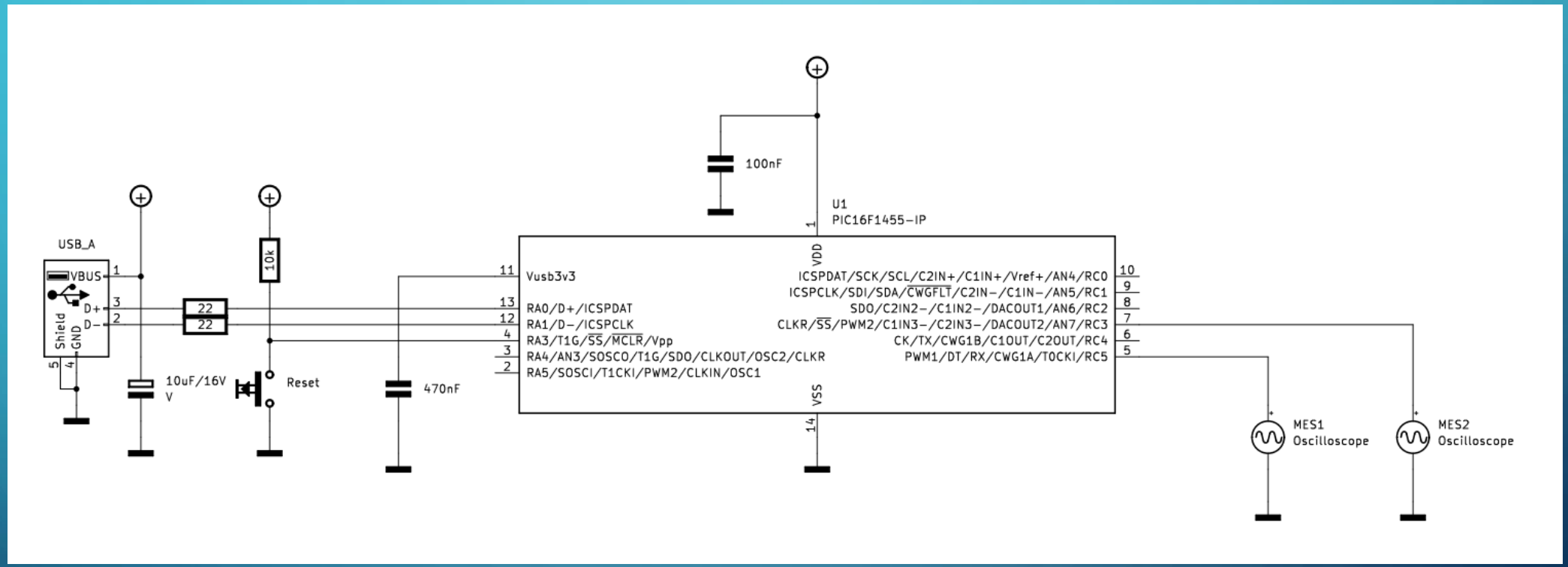
CONTROL THE ADC



CONTROL THE DAC



CONTROL THE PWM



REFERENCES

- The JAL sources, hardware and documentation of this USB I/O Expander can be found at: https://github.com/RobJansen62/USB_IO_Expander
- If you want to know more about JAL and the JAL programming language, visit the JAL website at: www.justanotherlanguage.org
- If you have questions on how to use JAL, there is a very good **Jallib Tutotrial** available on the JAL website or you can visit the **Jallist** Google Group and post your question
- For reporting bugs in JAL libraries or the JAL compiler or for specific requests for libraries visit the **Jallib** Google Group
- All JAL sources including the sources of the JAL compiler can be found on GitHub at: https://github.com/jallib/

SEE YOU NEXT TIME

Thanks for watching