PYNQ™

IOP Architecture

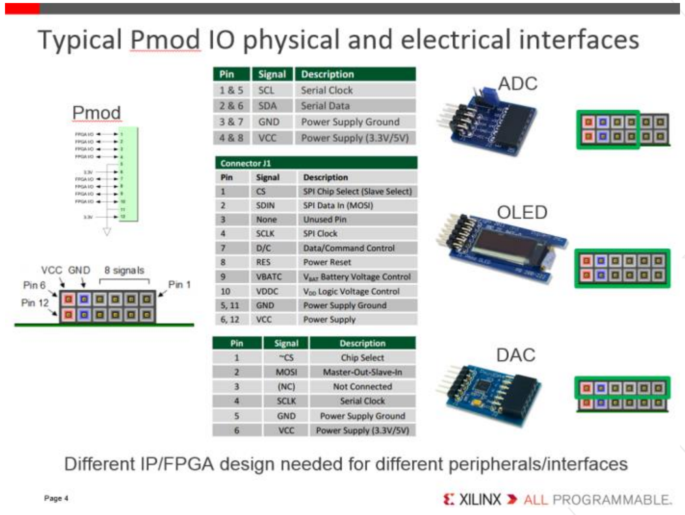
XILINX®

Outline

- IOP & supported interfaces
- IOP architecture
- Software build flow
- Managing projects
- Existing software projects
- Creating your own project

IOPs

- Introduction to IOPs in previous section
- ***base* overlay contains**
  - 2x Pmod IOPs
  - 1x Arduino IOP
  - 1x RPi IOP(PYNQ-Z2)
- **Supports Pmods, Arduino shields, Raspberry Pi and *Grove* peripherals**



Typical Pmod IO physical and electrical interfaces

Pin	Signal	Description
1 & 5	SCL	Serial Clock
2 & 6	SDA	Serial Data
3 & 7	GND	Power Supply Ground
4 & 8	VCC	Power Supply (3.3V/5V)

Connector J1		
Pin	Signal	Description
1	CS	SPI Chip Select (Slave Select)
2	SDIN	SPI Data In (MOSI)
3	None	Unused Pin
4	SCLK	SPI Clock
7	D/C	Data/Command Control
8	RES	Power Reset
9	VBATC	V_{BAT} Battery Voltage Control
10	VDDC	V_{DD} Logic Voltage Control
5, 11	GND	Power Supply Ground
6, 12	VCC	Power Supply

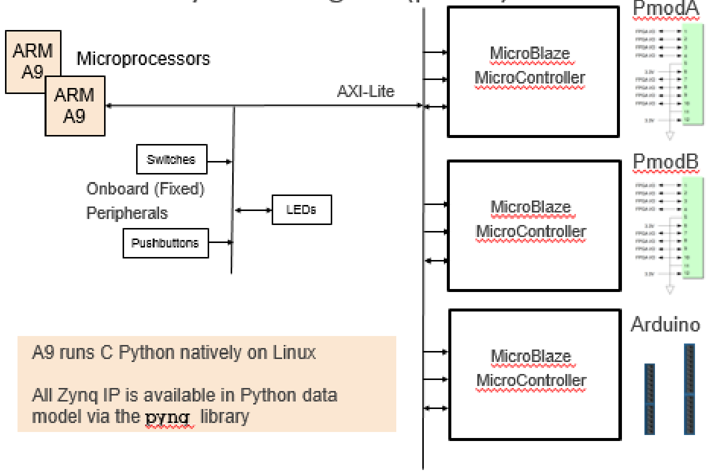
Pin	Signal	Description
1	~CS	Chip Select
2	MOSI	Master-Out-Slave-In
3	(NC)	Not Connected
4	SCLK	Serial Clock
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)

Different IP/FPGA design needed for different peripherals/interfaces

Page 4

XILINX ALL PROGRAMMABLE.

`base` overlay block diagram (partial)

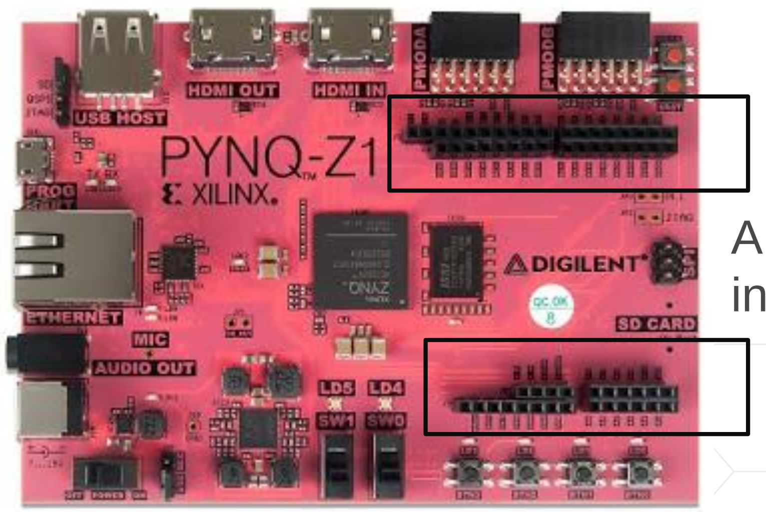

ARM A9
ARM A9
Microprocessors
AXI-Lite
Switches
Onboard (Fixed) Peripherals
LEDs
Pushbuttons
MicroBlaze MicroController
MicroBlaze MicroController
MicroBlaze MicroController
PmodA
PmodB
Arduino

A9 runs C Python natively on Linux

All Zynq IP is available in Python data model via the `pynq` library

Arduino Interface

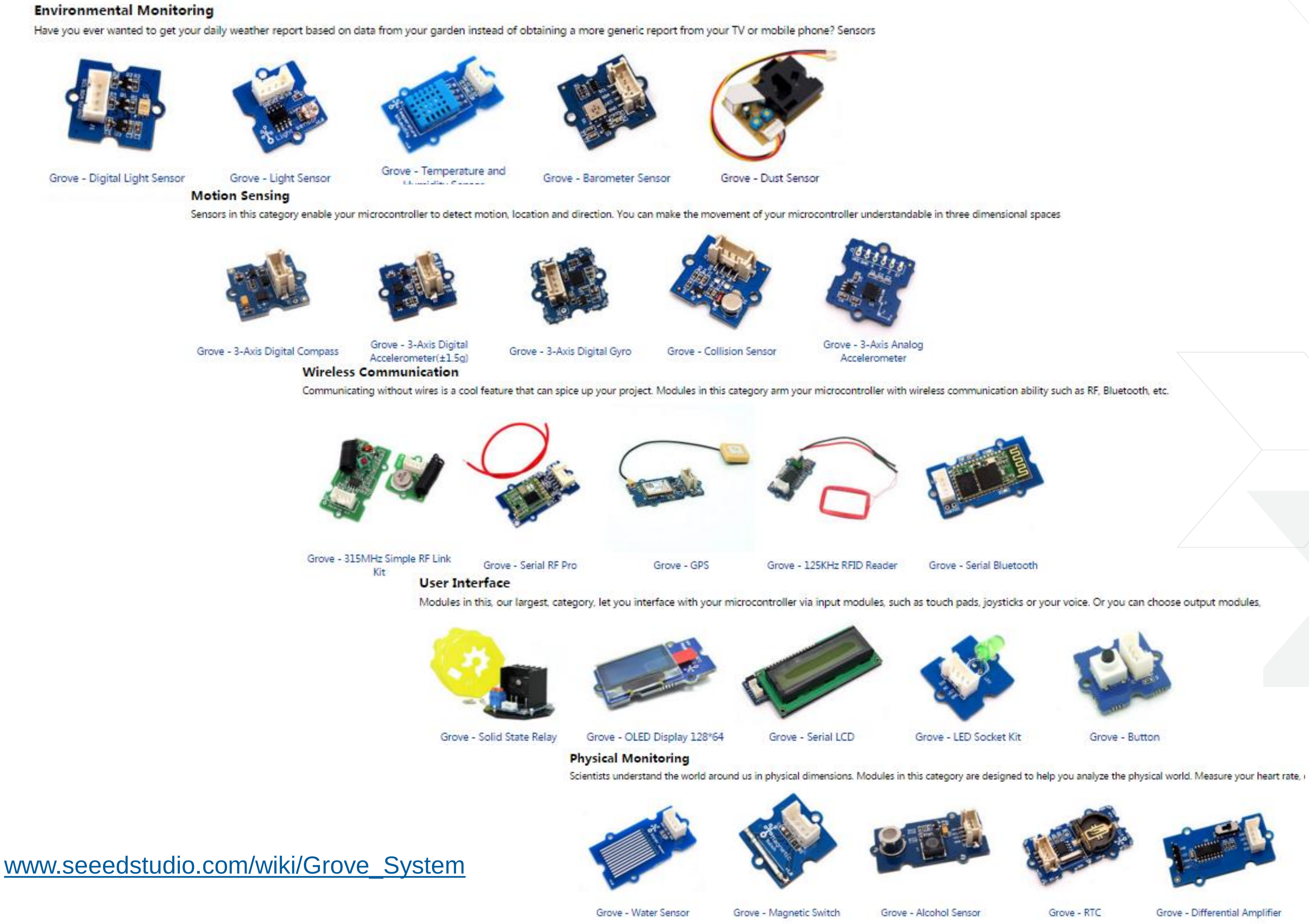
- **Wide range of off-the-shelf Arduino shields**
- **Arduino interface specification**
  - 6 Analog Inputs
  - 14 Digital pins
    - UART, PWM, Timer, SPI, interrupts
  - Dedicated SPI, I2C
- **On PYNQ-Z1/Z2 header connected to FPGA pins**
  - Interface is built in Overlay
  - Can breadboard to these pins



Arduino interface

RANDOMNERDTUTORIALS.COM

Grove: Wide range low-cost sensors, actuators, etc



Environmental Monitoring

Have you ever wanted to get your daily weather report based on data from your garden instead of obtaining a more generic report from your TV or mobile phone? Sensors

Grove - Digital Light Sensor

Grove - Light Sensor

Grove - Temperature and Humidity Sensor

Grove - Barometer Sensor

Grove - Dust Sensor

Motion Sensing

Sensors in this category enable your microcontroller to detect motion, location and direction. You can make the movement of your microcontroller understandable in three dimensional spaces

Grove - 3-Axis Digital Compass

Grove - 3-Axis Digital Accelerometer(±1.5g)

Grove - 3-Axis Digital Gyro

Grove - Collision Sensor

Grove - 3-Axis Analog Accelerometer

Wireless Communication

Communicating without wires is a cool feature that can spice up your project. Modules in this category arm your microcontroller with wireless communication ability such as RF, Bluetooth, etc.

Grove - 315MHz Simple RF Link Kit

Grove - Serial RF Pro

Grove - GPS

Grove - 125KHz RFID Reader

Grove - Serial Bluetooth

User Interface

Modules in this, our largest, category, let you interface with your microcontroller via input modules, such as touch pads, joysticks or your voice. Or you can choose output modules,

Grove - Solid State Relay

Grove - OLED Display 128*64

Grove - Serial LCD

Grove - LED Socket Kit

Grove - Button

Physical Monitoring

Scientists understand the world around us in physical dimensions. Modules in this category are designed to help you analyze the physical world. Measure your heart rate,

Grove - Water Sensor

Grove - Magnetic Switch

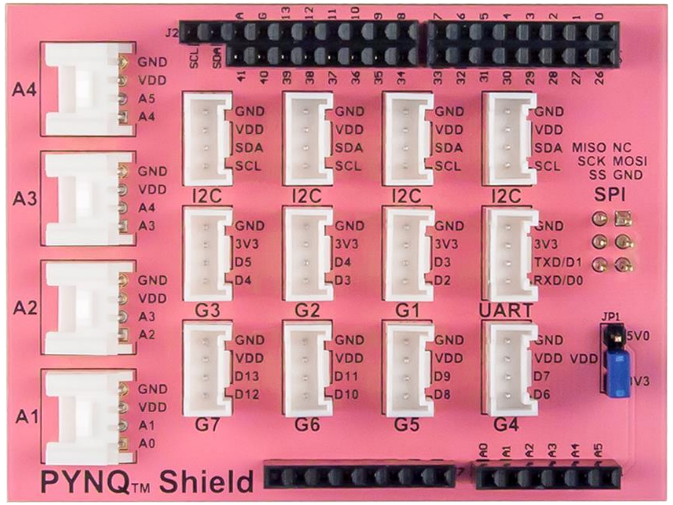
Grove - Alcohol Sensor

Grove - RTC

Grove - Differential Amplifier

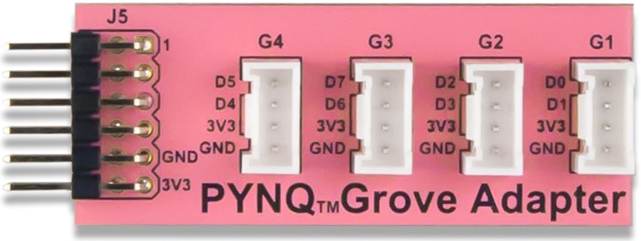
www.seeedstudio.com/wiki/Grove_System

Low-cost PYNQ Shield & Pmod Grove Adapter



PYNQ Shield:

- 4 x Analog ports
- 4 x I2C ports
- 3 x 3.3V GPIO ports
- 1 x UART
- 4 x 3.3/5V switchable GPIO ports
- 1 x SPI header
- 1 x 16-pin GPIO header (inner header)



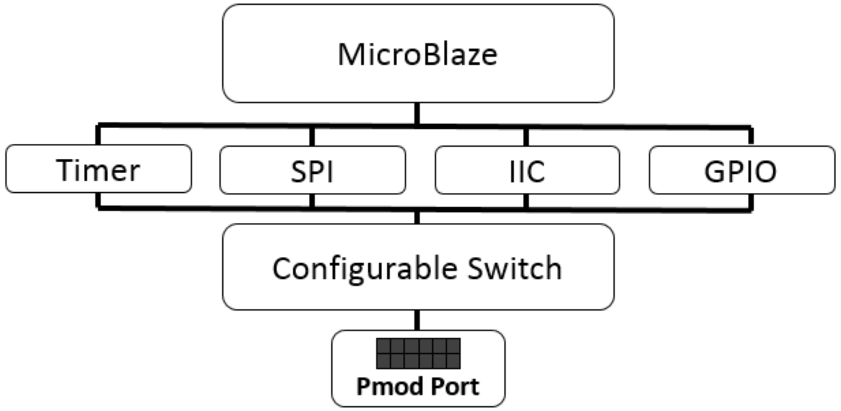
PYNQ Grove Adapter :

- 4 independent sockets for Grove modules
- Pmod compatible
- Solderless breadboard compatible
- Open-source design

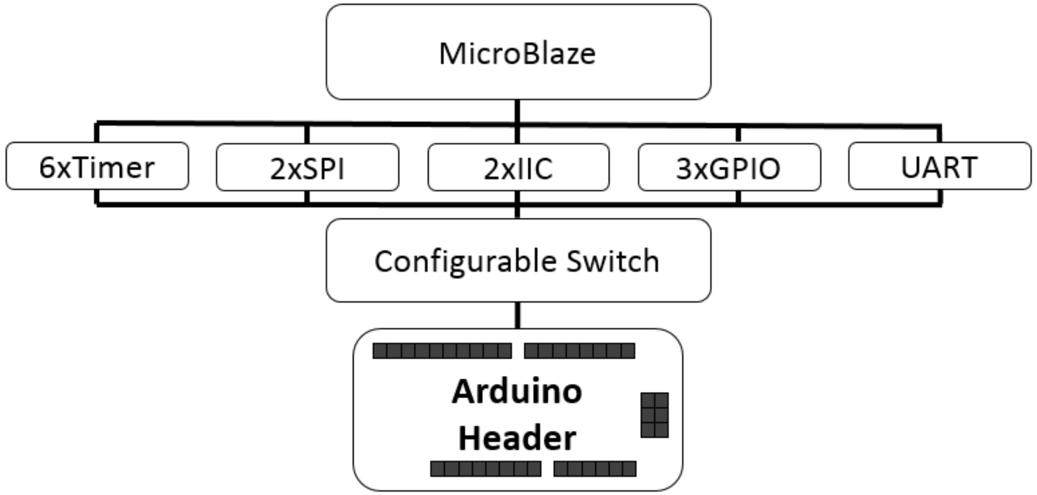
IOP Software

IOP software flow

- Pmod IOP/Arduino IOP/ RPi IOP
  - Same MicroBlaze & instruction/data memory
  - Same configurable switch
  - Supports wide range of peripherals
- The process for building software is the same



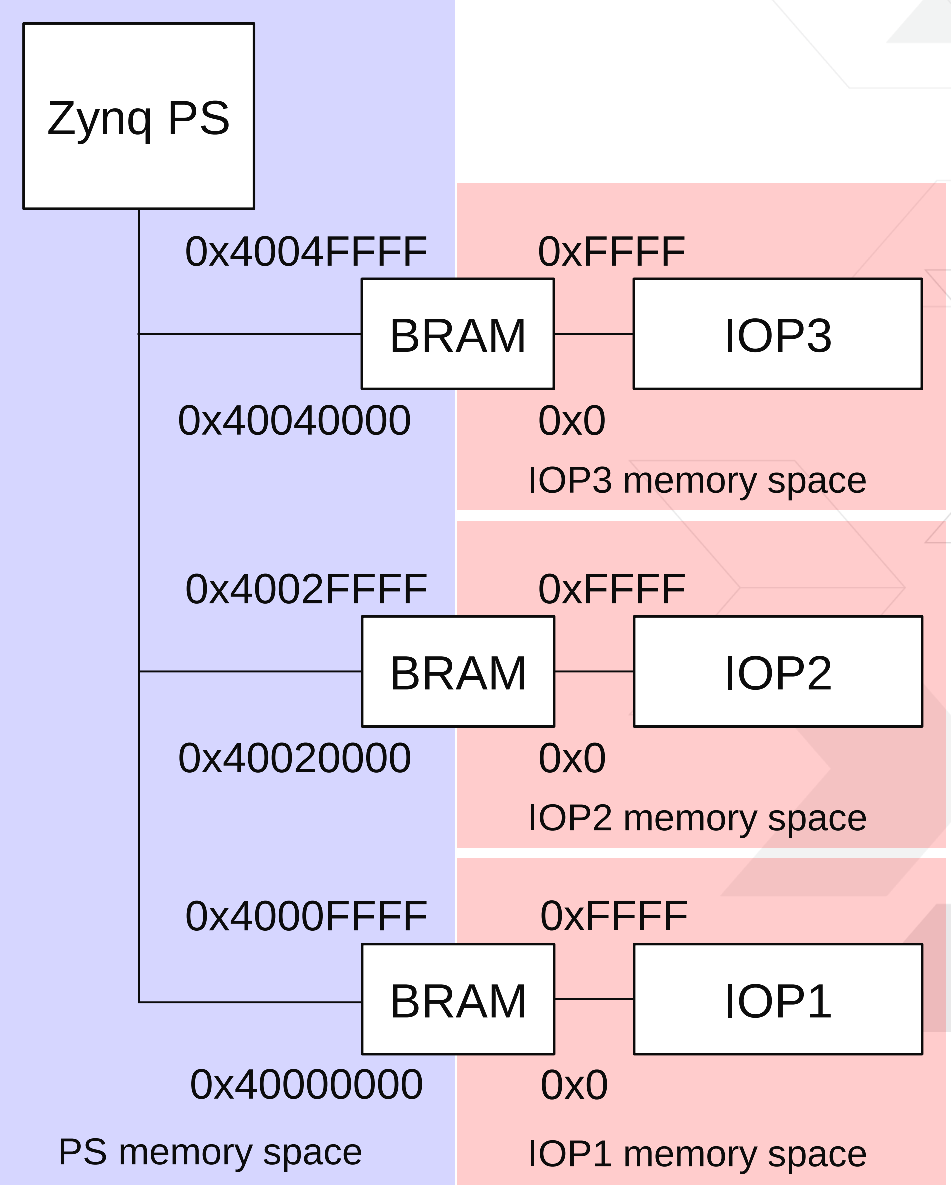
Pmod IOP



Arduino IOP

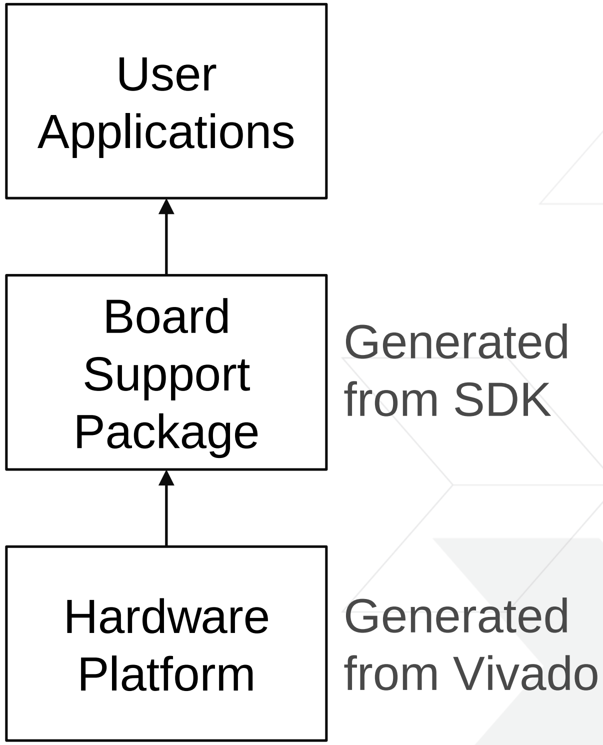
Building an IOP executable

- IOP instruction/data memory accessible from IOP and PS
- From the PS perspective:
  - Each IOP memory has different location in PS memory map
- From the IOP perspective:
  - Each IOP has a consistent memory map
  - Code for an IOP can be compiled for any IOP (of the same type)
    - E.g. Pmod IOP executable will run on other Pmod IOPs, not on an Arduino IOP
  - The same executable can be run on any IOP (of the same type)
- PS/Python can load program, and share data with IOP



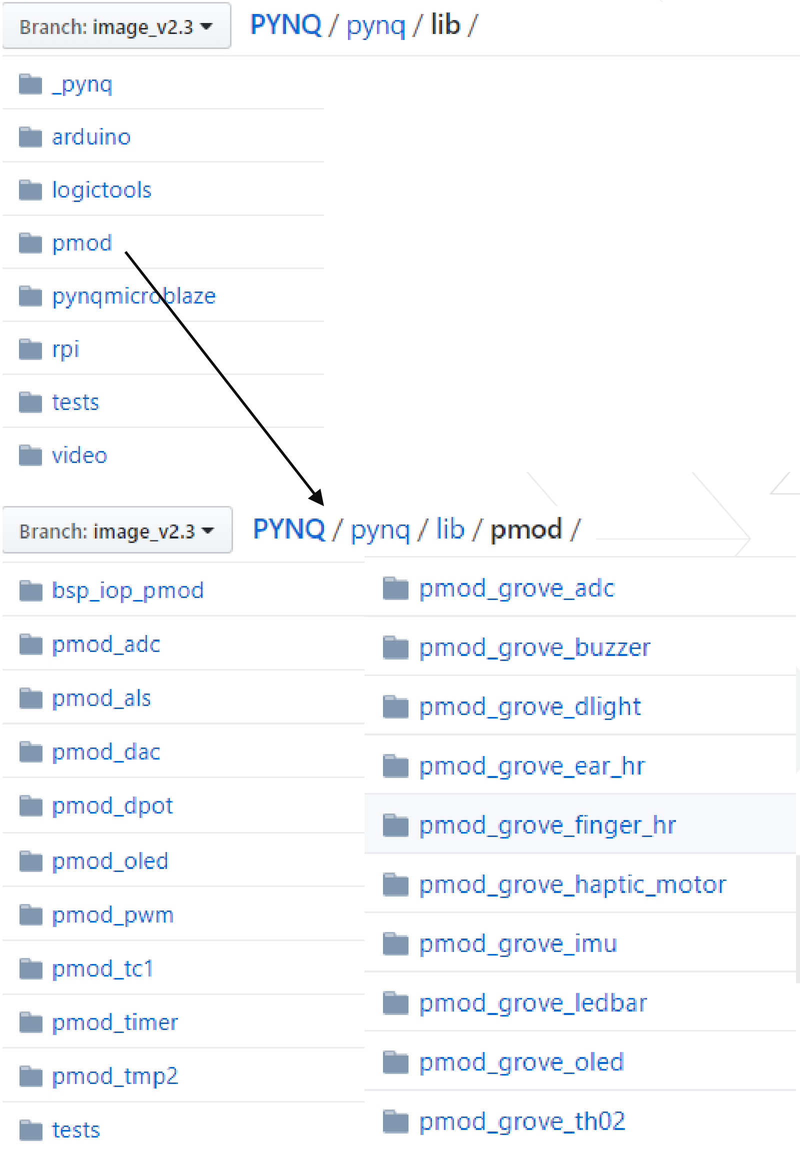
Writing software

- Standard MicroBlaze software design
  - Xilinx SDK
  - gcc/make flow
- "Hardware Platform" required
  - Generated by Vivado
  - Available pre-compiled in Pynq repository
- "Board Support Package" required
  - Requires Hardware Platform
  - Generated by SDK



Example projects (GitHub)

- **Source code and projects available on GitHub for a range of peripherals**
  - Grove and Pmod
  - Some Arduino shield examples
  - Can be used as starting point for a new project
- **API available**
  - IIC, SPI, GPIO, Configurable switch
    - Simple low level API's; Read( ), Write( )
- ***Make* flow to build IOP projects available**



Branch: image_v2.3 ▾ PYNQ / pynq / lib /

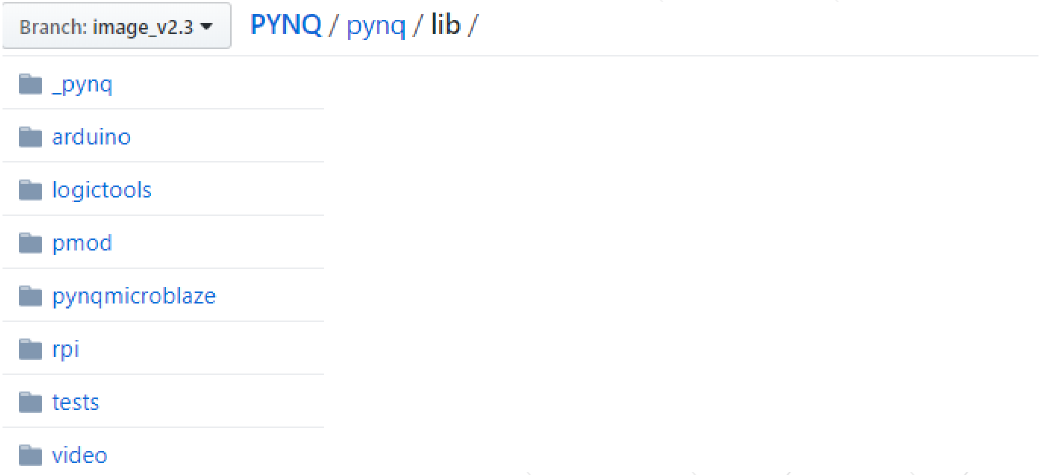
- _pynq
- arduino
- logictools
- pmod
- pynqmicroblaze
- rpi
- tests
- video

Branch: image_v2.3 ▾ PYNQ / pynq / lib / pmod /

- bsp_iop_pmod
- pmod_adc
- pmod_als
- pmod_dac
- pmod_dpot
- pmod_oled
- pmod_pwm
- pmod_tc1
- pmod_timer
- pmod_tmp2
- tests
- pmod_grove_adc
- pmod_grove_buzzer
- pmod_grove_dlight
- pmod_grove_ear_hr
- pmod_grove_finger_hr
- pmod_grove_haptic_motor
- pmod_grove_imu
- pmod_grove_ledbar
- pmod_grove_oled
- pmod_grove_th02

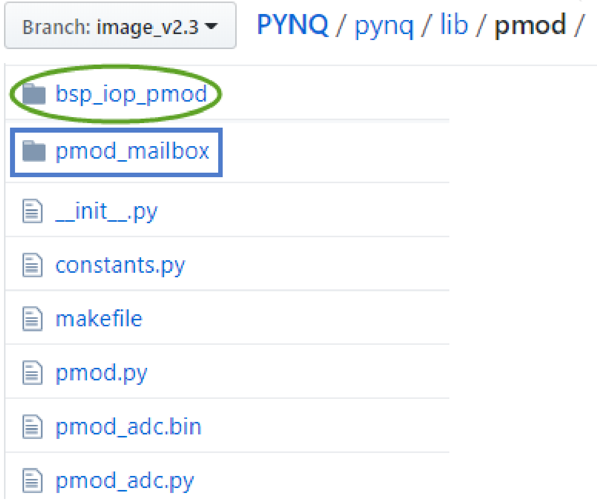
Software directory (GitHub)

- > **Various software projects grouped according to interface and overlay related reside under ./pynq/lib/**
  - >> Arduino, logictools, Pmod, pynqmicroblaze, rpi, video
- > **Under each group reside related software projects, bsp, makefile, bin (binary executable files), and Python class file**
- > **mailbox**
  - >> Enables data and command/status exchanges between AP and IOP



Branch: image_v2.3 ▾ PYNQ / pynq / lib /

- _pynq
- arduino
- logictools
- pmod
- pynqmicroblaze
- rpi
- tests
- video



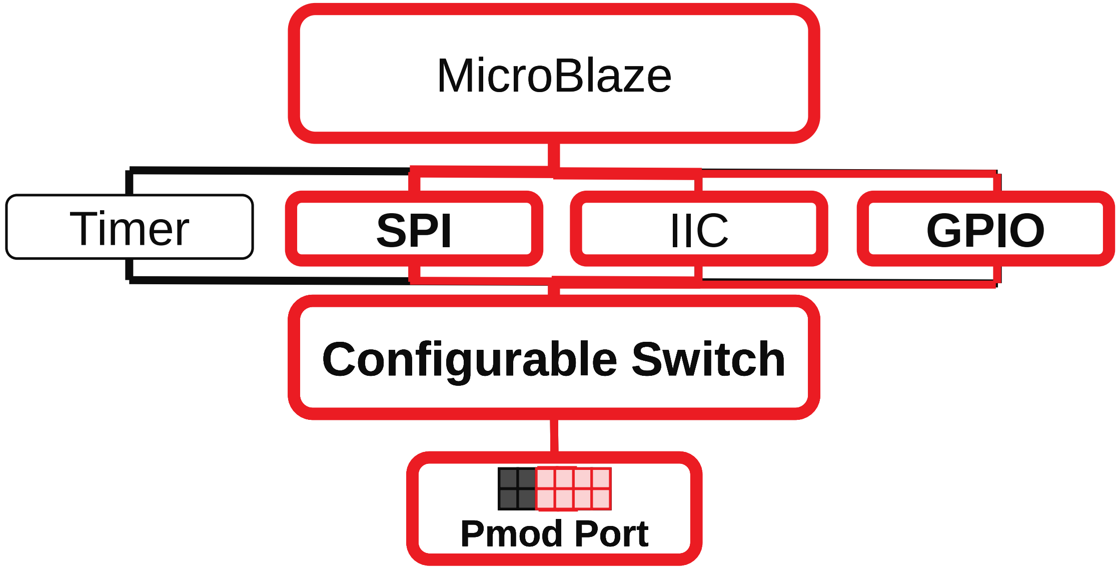
Branch: image_v2.3 ▾ PYNQ / pynq / lib / pmod /

- bsp_iop_pmod
- pmod_mailbox
- __init__.py
- constants.py
- makefile
- pmod.py
- pmod_adc.bin
- pmod_adc.py

Programming the IO Switch

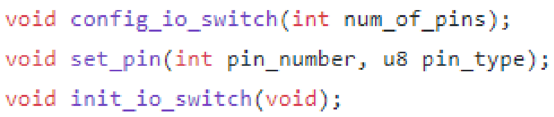
Configurable Switch

> **Allows peripherals with different interfaces to be used in the same overlay without needing a new FPGA design**

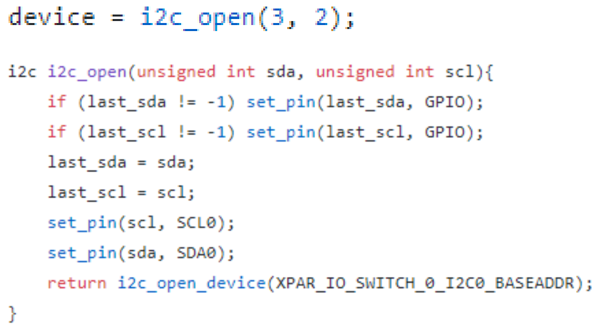


Configurable Switch

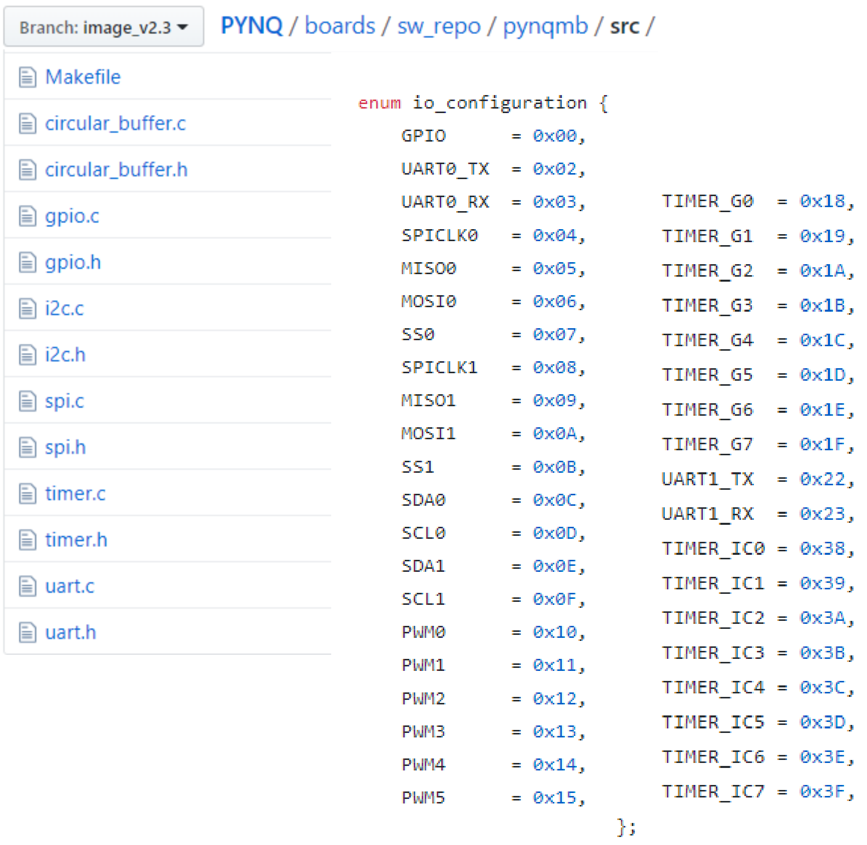
- **Common API for all types of interfaces**
  - Pmod, Arduino, Raspberry Pi
  - xio_switch.h, xio_switch.c
    - config_io_switch(), set_pin(), init_io_switch()
- **Pin types can be**
  - GPIO, I2C, SPI, Timer, UART
- **Call open_device()**
  - Calls set_pin()
    - Pin 3 is configured as SCL
    - Pin 2 is configured as SDA



```
void config_io_switch(int num_of_pins);
void set_pin(int pin_number, u8 pin_type);
void init_io_switch(void);
```



```
device = i2c_open(3, 2);

i2c i2c_open(unsigned int sda, unsigned int scl){
    if (last_sda != -1) set_pin(last_sda, GPIO);
    if (last_scl != -1) set_pin(last_scl, GPIO);
    last_sda = sda;
    last_scl = scl;
    set_pin(scl, SCL0);
    set_pin(sda, SDA0);
    return i2c_open_device(XPAR_IO_SWITCH_0_I2C0_BASEADDR);
}
```



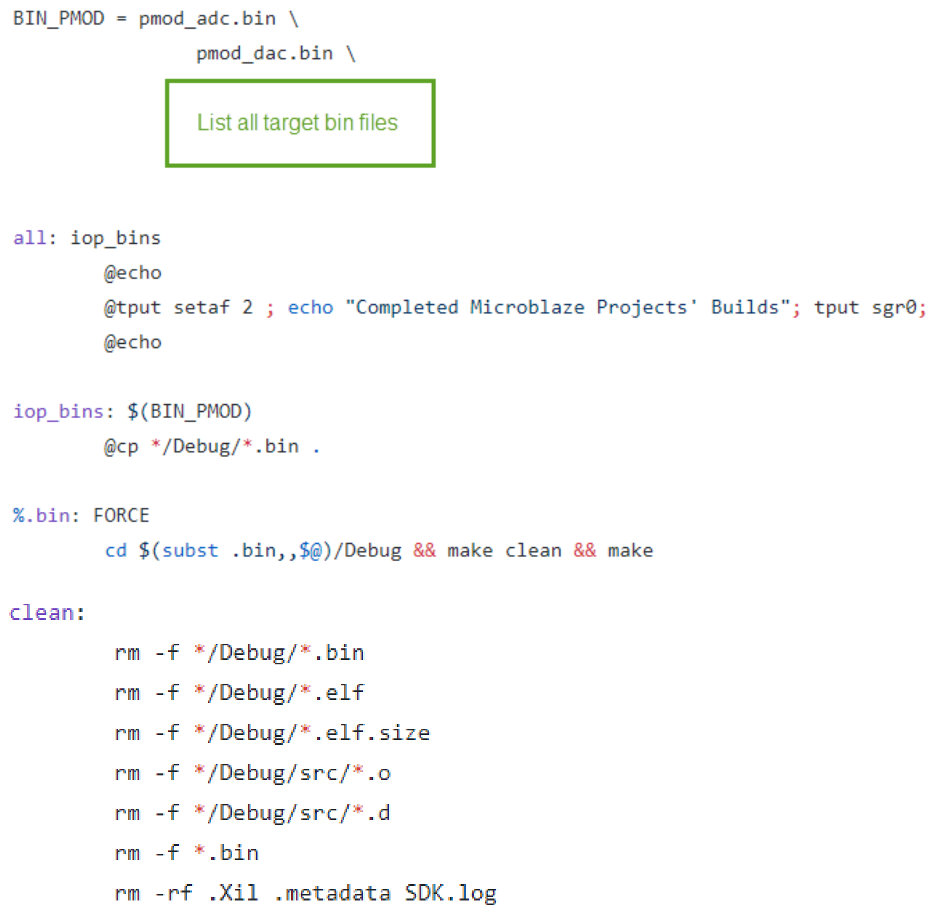
Makefile
circular_buffer.c
circular_buffer.h
gpio.c
gpio.h
i2c.c
i2c.h
spi.c
spi.h
timer.c
timer.h
uart.c
uart.h

```
enum io_configuration {
    GPIO      = 0x00,
    UART0_TX  = 0x02,
    UART0_RX  = 0x03,      TIMER_G0   = 0x18,
    SPICLK0   = 0x04,      TIMER_G1   = 0x19,
    MISO0     = 0x05,      TIMER_G2   = 0x1A,
    MOSI0     = 0x06,      TIMER_G3   = 0x1B,
    SS0       = 0x07,      TIMER_G4   = 0x1C,
    SPICLK1   = 0x08,      TIMER_G5   = 0x1D,
    MISO1     = 0x09,      TIMER_G6   = 0x1E,
    MOSI1     = 0x0A,      TIMER_G7   = 0x1F,
    SS1       = 0x0B,      UART1_TX   = 0x22,
    SDA0      = 0x0C,      UART1_RX   = 0x23,
    SCL0      = 0x0D,      TIMER_IC0 = 0x38,
    SDA1      = 0x0E,      TIMER_IC1 = 0x39,
    SCL1      = 0x0F,      TIMER_IC2 = 0x3A,
    PWM0      = 0x10,      TIMER_IC3 = 0x3B,
    PWM1      = 0x11,      TIMER_IC4 = 0x3C,
    PWM2      = 0x12,      TIMER_IC5 = 0x3D,
    PWM3      = 0x13,      TIMER_IC6 = 0x3E,
    PWM4      = 0x14,      TIMER_IC7 = 0x3F,
    PWM5      = 0x15,
                       };
```

Building software

Makefile flow

- Xilinx SDK installation on host PC
- Creates SDK Workspace
- Traverses & builds each project directory
  - Generate binary executable (.bin) for each project
  - Copy executables to bin/



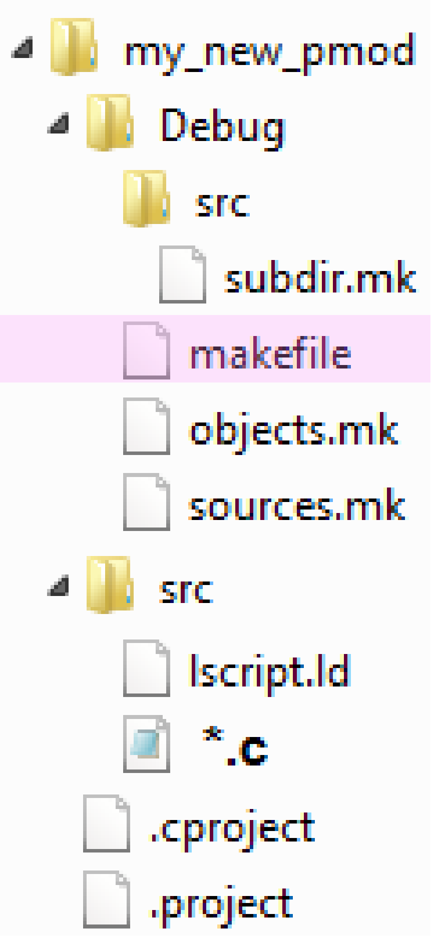
```
BIN_PMOD = pmod_adc.bin \
                pmod_dac.bin \

all: iop_bins
        @echo
        @tput setaf 2 ; echo "Completed Microblaze Projects' Builds"; tput sgr0;
        @echo

iop_bins: $(BIN_PMOD)
        @cp */Debug/*.bin .

%.bin: FORCE
        cd $(subst .bin,,$@)/Debug && make clean && make

clean:
         rm -f */Debug/*.bin
         rm -f */Debug/*.elf
         rm -f */Debug/*.elf.size
         rm -f */Debug/src/*.o
         rm -f */Debug/src/*.d
         rm -f *.bin
         rm -rf .Xil .metadata SDK.log
```

List all target bin files

.\pynq\lib\pmod\makefile

Project makefile

- **Each software project has a makefile**
  - E.g. pynq\lib\pmod\pmod_als\Debug\makefile
  - Called by top level make
  - Builds software project, generates executable (.elf)
- **Binary executable file (.bin)**
  - Project *make* converts from .elf to binary format
  - Loaded to MicroBlaze instruction memory
- **BIN_* defined in top level makefile**
  - \pynq\lib\*\makefile
  - Includes each project in the build flow
  - Add your own project name + “.bin”



```
my_new_pmod
  Debug
    src
      subdir.mk
    makefile
    objects.mk
    sources.mk
  src
    lscript.ld
    *.c
  .cproject
  .project
```

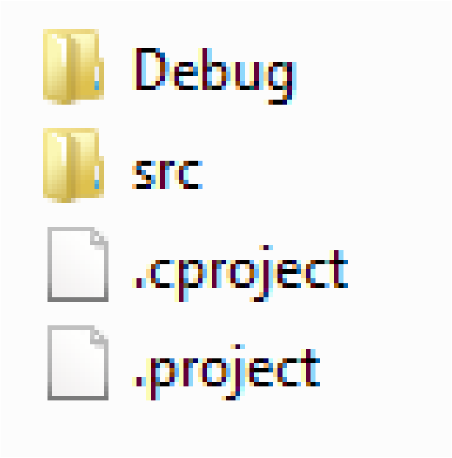
Managing Projects

XILINX

PYNQ

IOP Project

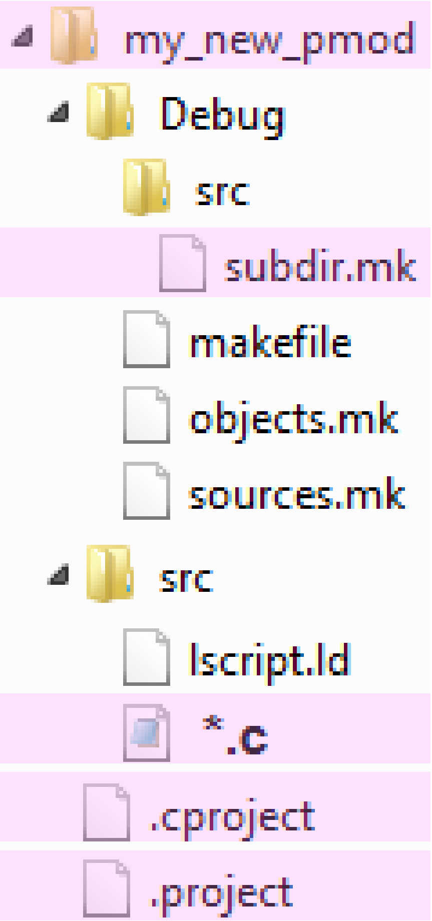
- **Xilinx SDK project files**
  - .cproject, .project
  - Not essential, but allow project to be imported back into SDK
- **src/**
  - Contains C source code, and linker script
- **Debug/**
  - makefile to build IOP project as seen previously
  - Other project files (includes objects, sources, directories, build settings)



Debug
src
.cproject
.project

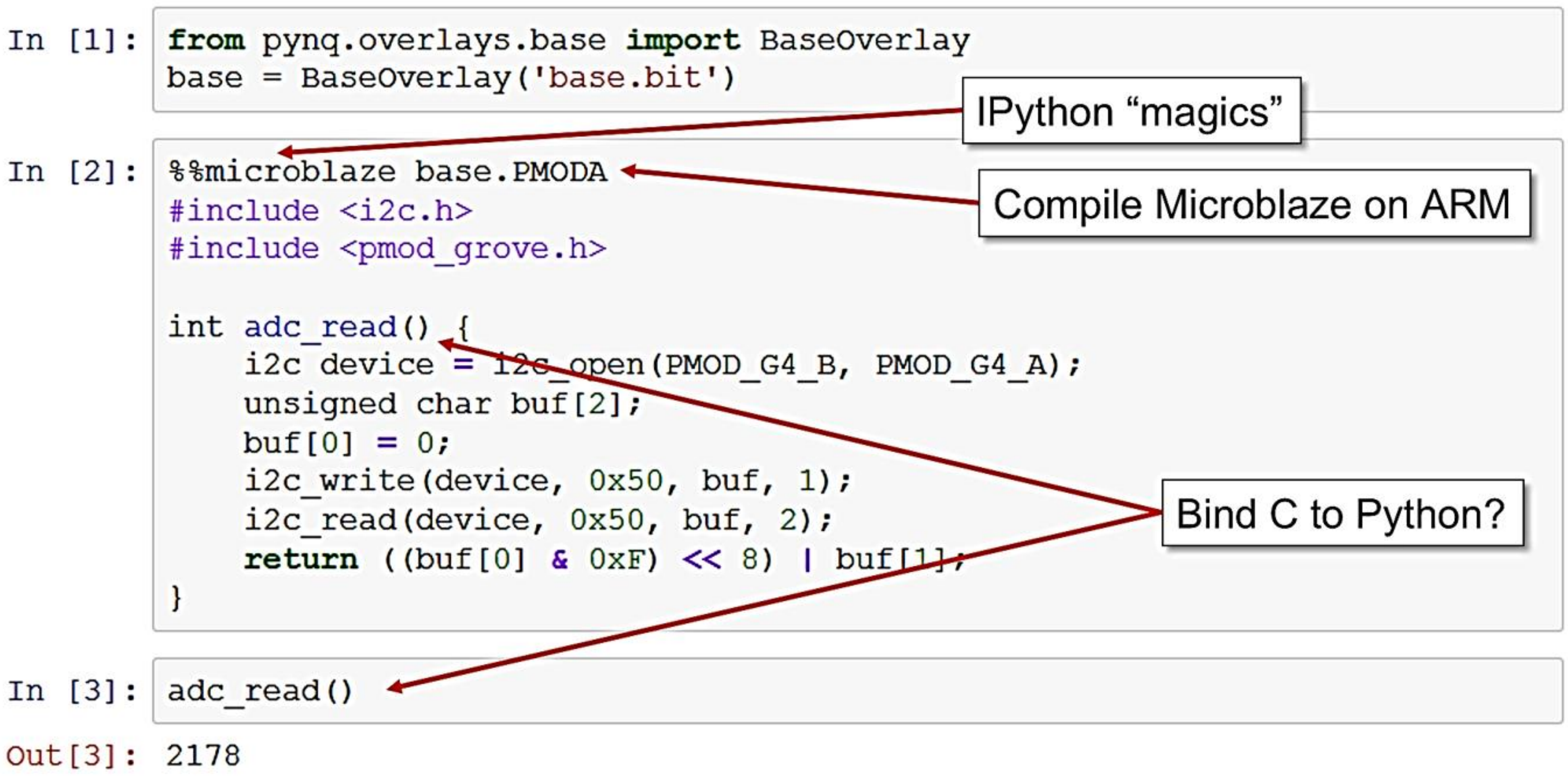
Creating your own IOP program

- Recommended to start with existing project
- Copy project folder and rename
  - E.g. pmod_als -> my_new_pmod
- Find and replace project name in the following files:
  - E.g. pmod_als -> my_new_pmod
    - .project, .cproject
    - Debug/makefile
    - Debug/src/subdir.mk
      - Add any other new source files to this file
- Modify/Replace existing .c/.h source file in src/



```
my_new_pmod
    Debug
        src
            subdir.mk
        makefile
        objects.mk
        sources.mk
    src
        lscript.ld
        *.c
    .cproject
    .project
```

MicroBlaze magic!



```
In [1]: from pynq.overlays.base import BaseOverlay
        base = BaseOverlay('base.bit')

In [2]: %%microblaze base.PMODA
        #include <i2c.h>
        #include <pmod_grove.h>

        int adc_read() {
            i2c device = i2c_open(PMOD_G4_B, PMOD_G4_A);
            unsigned char buf[2];
            buf[0] = 0;
            i2c_write(device, 0x50, buf, 1);
            i2c_read(device, 0x50, buf, 2);
            return ((buf[0] & 0xF) << 8) | buf[1];
        }

In [3]: adc_read()

Out[3]: 2178
```

IPython “magics”

Compile Microblaze on ARM

Bind C to Python?

Summary

- IOP & supported interfaces
- IOP architecture
  - Pmod, Arduino, Raspberry Pi
- Software build flow
  - Makefile
- Managing projects
  - Existing software projects
  - Creating your own projects

Adaptable.
Intelligent.

XILINX.

PYNQ