XILINX
ALL PROGRAMMABLE™

PYNQ™

IOP Architecture

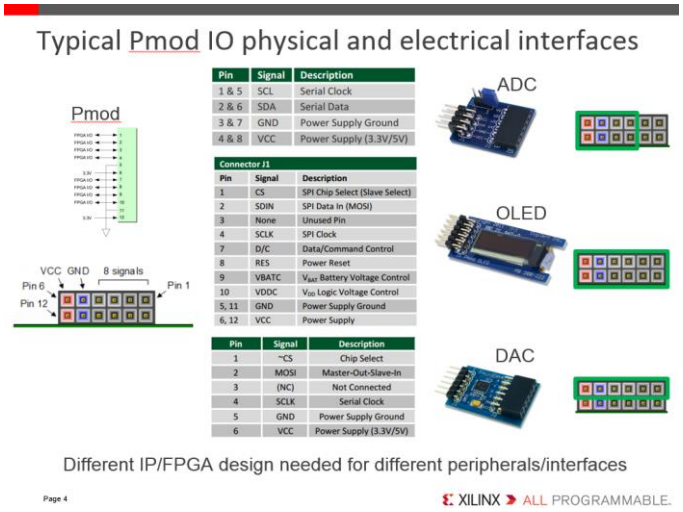
XILINX
ALL PROGRAMMABLE™

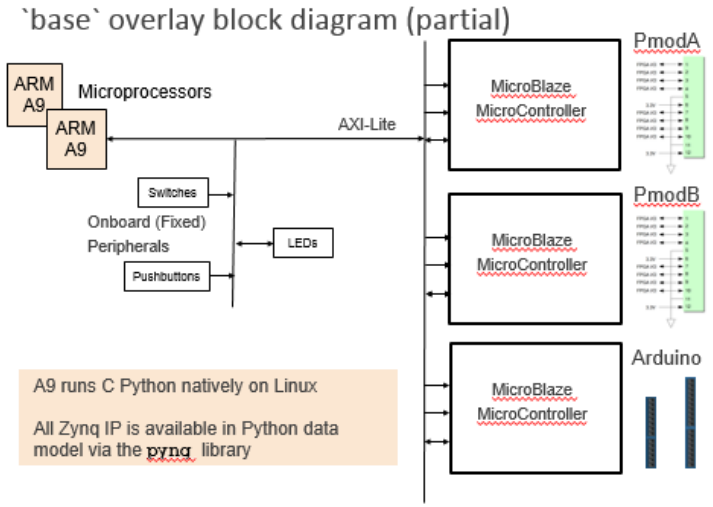
Agenda

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

IOPs

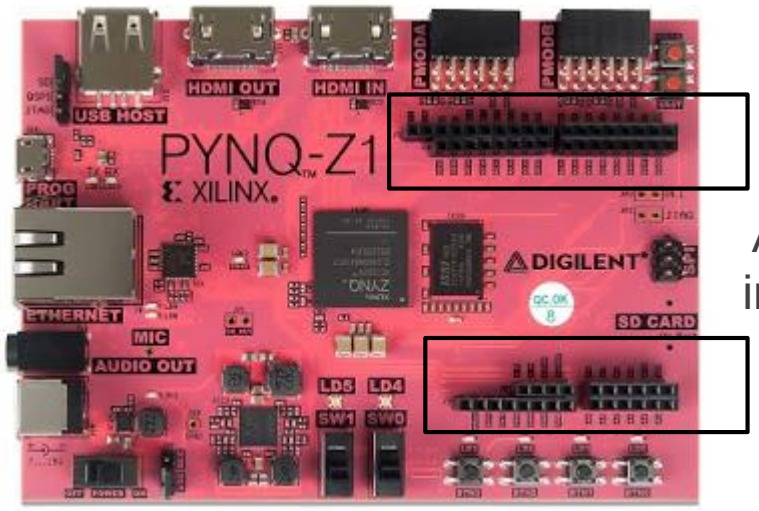
- Introduction to IOPs in Previous section
- *base* overlay contains
  - 2x Pmod IOPs
  - 1x Arduino IOPs
- Supports Pmods, Arduino shields, and *Grove* peripherals





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 header connected to FPGA pins**
  - Interface is built in Overlay
  - Can breadboard to these pins



Arduino interface

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 | 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

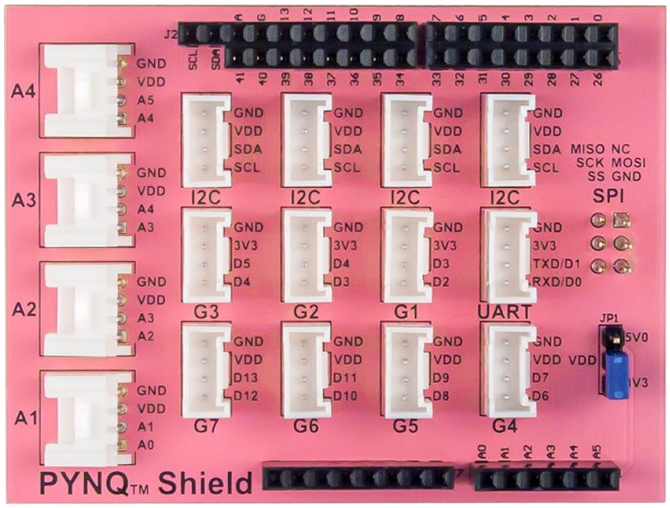
Physical Monitoring

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

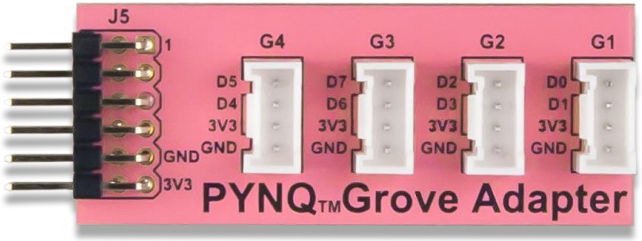
www.seeedstudio.com/wiki/Grove_System

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)



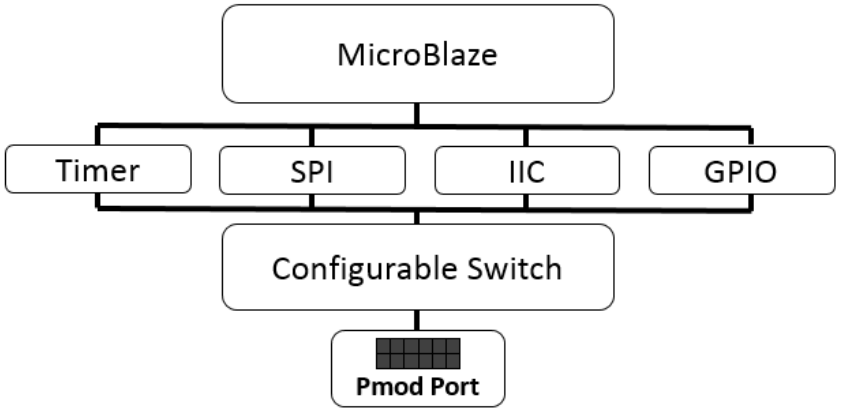
PYNQ Grove Adapter :

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

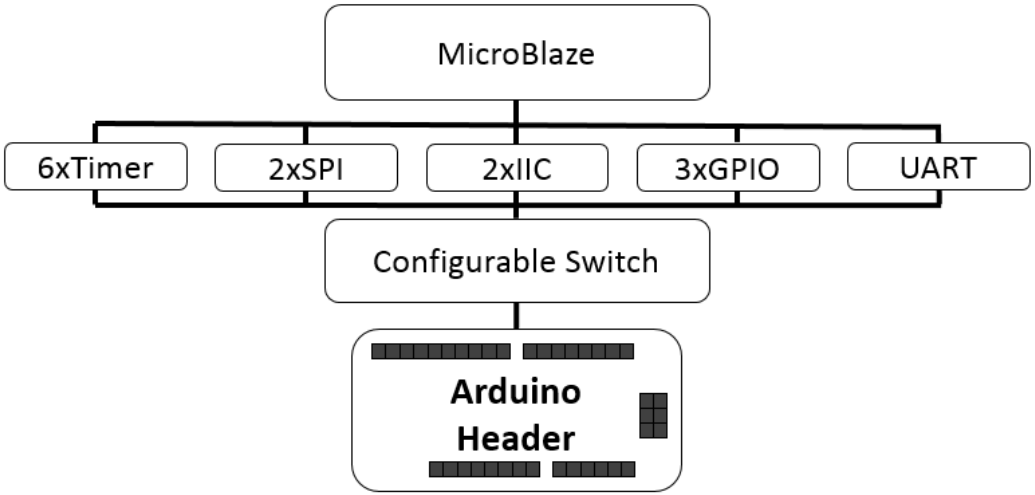
IOP Software

IOP software flow

- Pmod IOP/Arduino IOP
  - Contain the same MicroBlaze & instruction/data memory
  - Configurable switch and peripherals
- The process for building software is the same



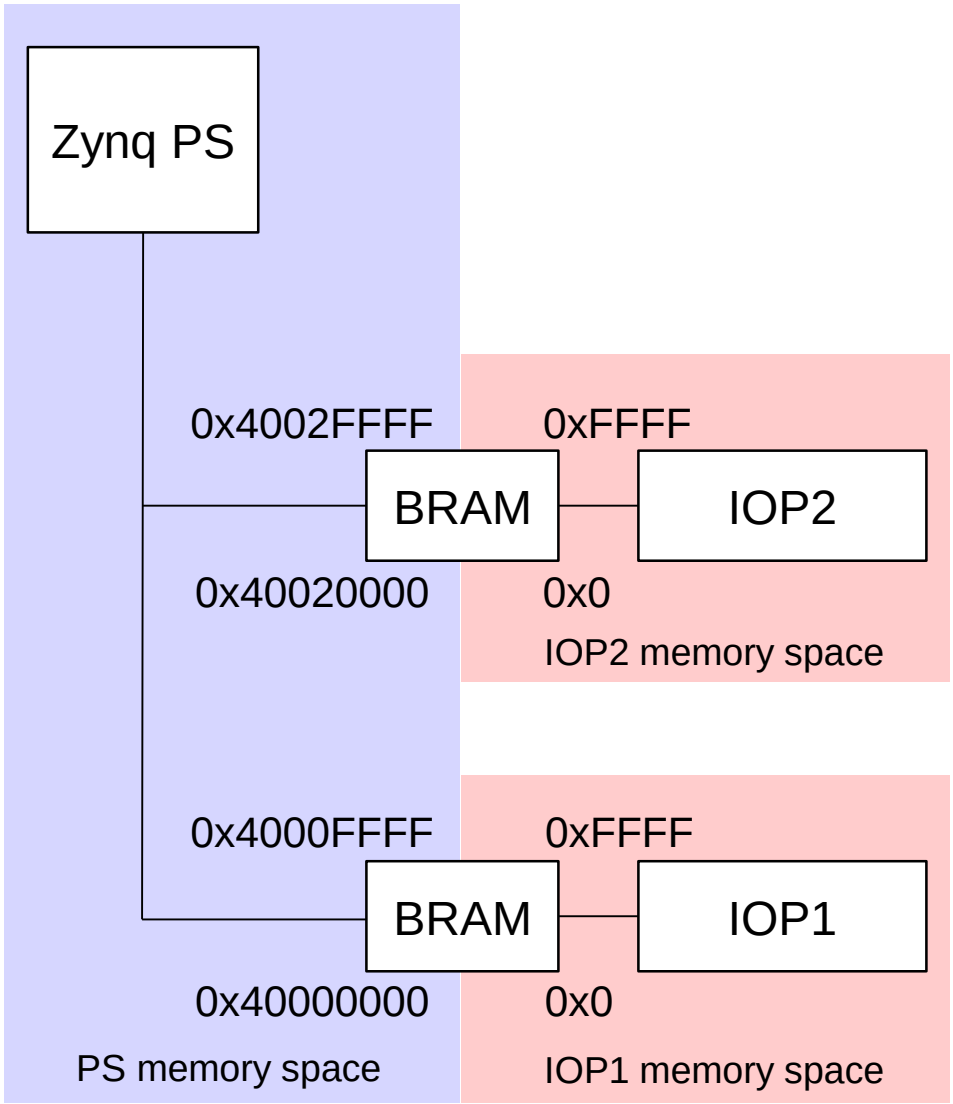
Pmod IOP



Arduino IOP

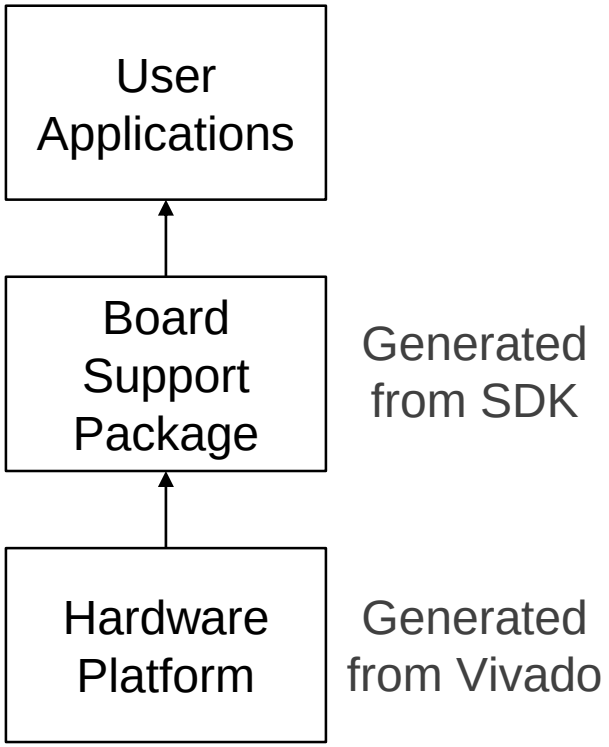
Building an IOP executable

- 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



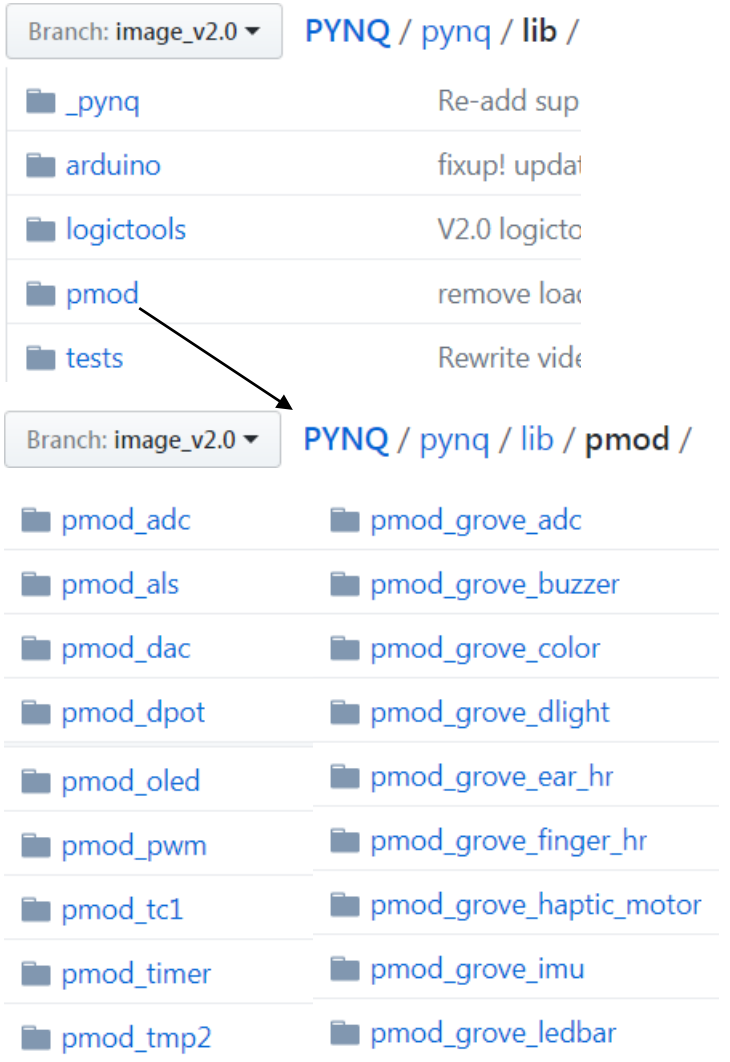
Writing software

- 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



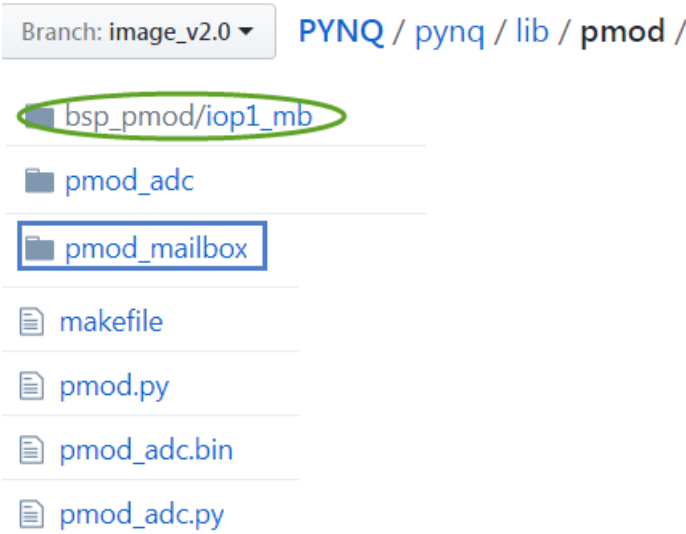
Example projects (GitHub)

- 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( )
  - pmod.c, pmod.h; arduino.c, arduino.h
- *Make* flow to build IOP projects available



Software directory (GitHub)

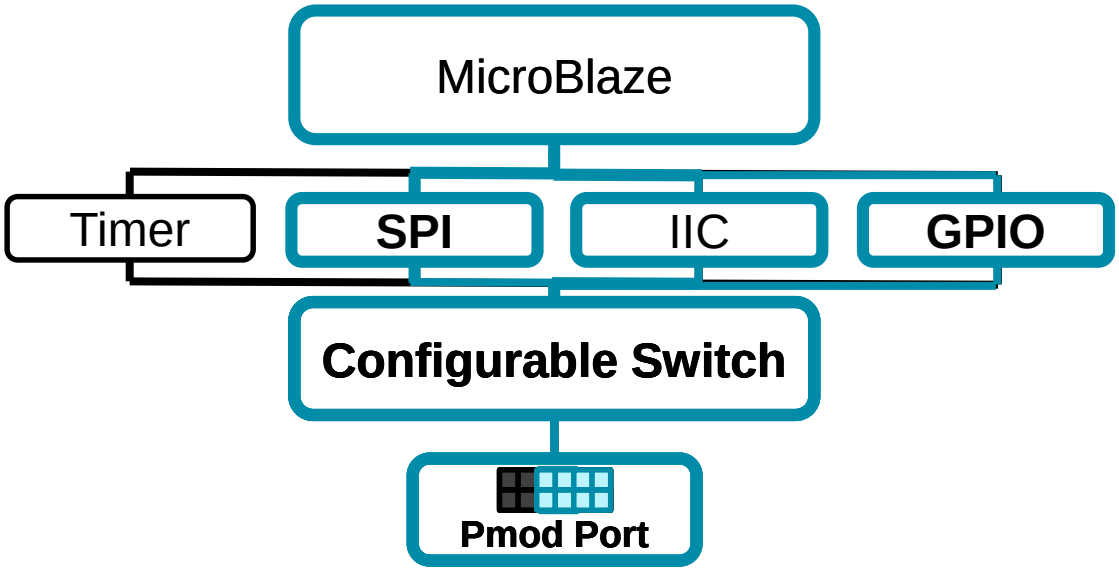
- Various software projects grouped according to interface and overlay related reside under ./pynq/lib/
  - Arduino, logictools, Pmod
- Under each group reside related software projects, bsp, makefile, bin (binary executable files), and Python class file
- mailbox
  - Enables data and command/status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



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 (Pmod)

- 8 pins can be connected to:
  - GPIO, I2C, SPI, Timer
- config_pmod_switch( )
  - Write a value for each pin
  - Pin map defined in pmod.h
  - e.g. Connect SPI to first 4 pins, and GPIO 5-8

```
config_pmod_switch(SPICLK, MISO, MOSI, SS,
                   GPIO_5, GPIO_6, GPIO_7, GPIO_8);
```

```
/*
 * Switch Configuration
 */
void config_pmod_switch(char pin0, char pin1, char pin2, char pin3,
                        char pin4, char pin5, char pin6, char pin7);

// Switch mappings used for IOP Switch configuration
#define GPIO_0 0x0
#define GPIO_1 0x1
#define GPIO_2 0x2
#define GPIO_3 0x3
#define GPIO_4 0x4
#define GPIO_5 0x5
#define GPIO_6 0x6
#define GPIO_7 0x7
#define SCL    0x8
#define SDA    0x9
#define SPICLK 0xa
#define MISO   0xb
#define MOSI   0xc
#define SS     0xd
#define PWM    0xe
#define TIMER  0xf
```

Configurable Switch (Arduino)

- 6 external analog pins can be connected to analog inputs (XADC)
  - Can also be used as Digital I/O
- 14 digital pins can be connected to:
  - GPIO, I2C, SPI, Timer
- config_arduino_switch( )
  - Write a value for each pin
  - Pin map defined in arduino.h

```
/*
 * Switch Configuration
 */
void config_arduino_switch(char A_pin0, char A_pin1, char A_pin2, char A_pin3,
                           char A_pin4, char A_pin5, char D_pin0_1,
                      char D_pin2, char D_pin3, char D_pin4, char D_pin5,
                      char D_pin6, char D_pin7, char D_pin8, char D_pin9,
                      char D_pin10, char D_pin11, char D_pin12, char D_pin13);

// Switch mappings used for Arduino Switch configuration
#define A_GPIO 0x0
#define A_INT 0x0
#define A_SDA 0x2
#define A_SCL 0x3

#define D_GPIO 0x0
#define D_UART 0x1
#define D_INT 0x1
#define D_PWM 0x2
#define D_TIMER_G 0x3    // to drive Timer Generate on the
#define D_SPICLK 0x4
#define D_MISO 0x5
#define D_MOSI 0x6
#define D_SS 0x7
#define D_TIMER_IC 0xb // to sink to Timer Input Capture
```

Building software

Makefile flow

- Xilinx SDK install required
- Creates SDK Workspace
- Traverses & builds each project directory
  - Generate binary executable (.bin) for each project
  - Copy executables to bin/

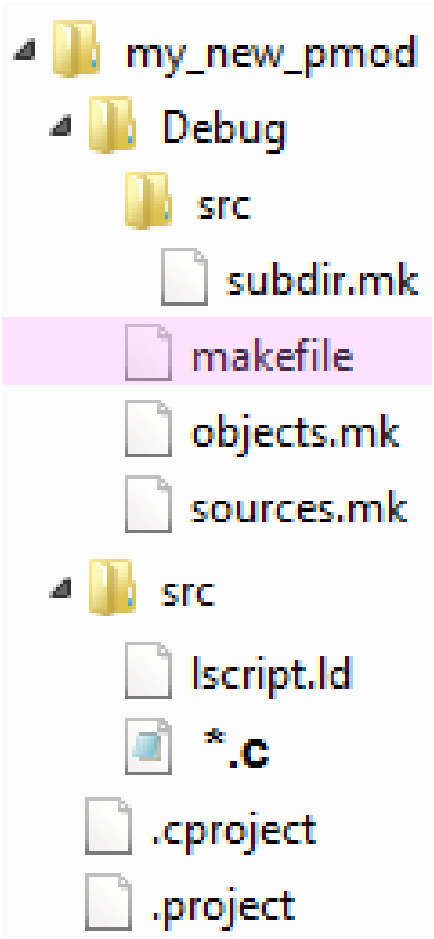
```
BIN_PMOD = pmod_adc.bin \
           pmod_dac.bin \

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

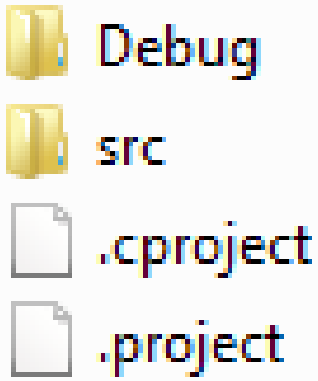
- Each software project has a makefile
  - E.g. pynq\lib\pmod\pmod_als\Debug\makefile
  - Called by top level make
  - Builds software project, generates executable (.elf)
- Binary executable file (.bin)
  - Project *make* converts from .elf to binary format
  - Loaded to MicroBlaze instruction memory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



Managing Projects

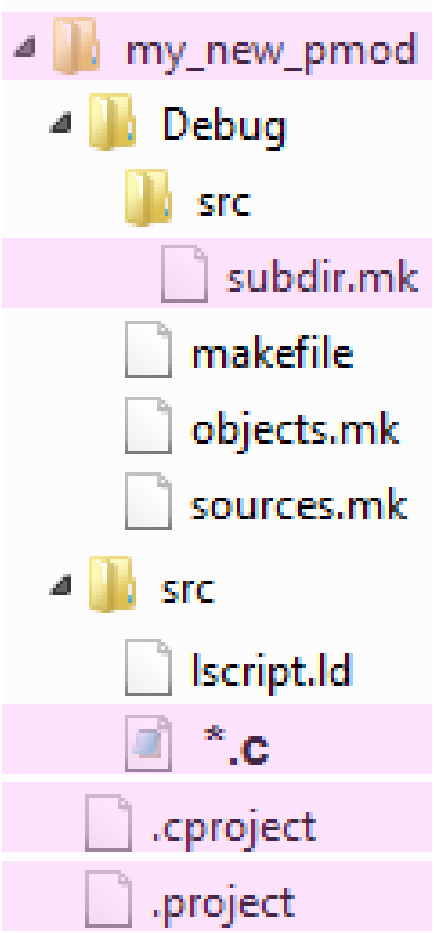
IOP Project

- 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)



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/



Summary

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

Questions?