ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# Hands-on Workshop: Introduction to Baremetal Embedded Applications

Larisa Radu – Software Engineer, Software Development

Monica Ignat – Applications Engineering Manager

Dariana Roman – Associate Engineer, Embedded Software

# Who are we?

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

| | |
|---|---|
| Founded | **1965** |
| Headquarters | **Norwood, MA** |
| Employees | **~15,000** |
| Countries | **20+** |
| Products | **~45,000 SKUs** |
| Customers | **125,000** |
| Publicly Listed | **NASDAQ:ADI**<br>Part of S&P 500 and NASDAQ 100 |
| Design Centers | **~45** |
| Global Manufacturing | **U.S. (Massachusetts, California, Washington), Ireland, Philippines, and Malaysia** |

**Over 4700 patents and $4 billion R&D investment in the past 10 years.**

# Technologies for Today's and Tomorrow's Innovations

Autonomous Transportation and Machines

Automotive Electrification

5G and Next-Gen Connectivity

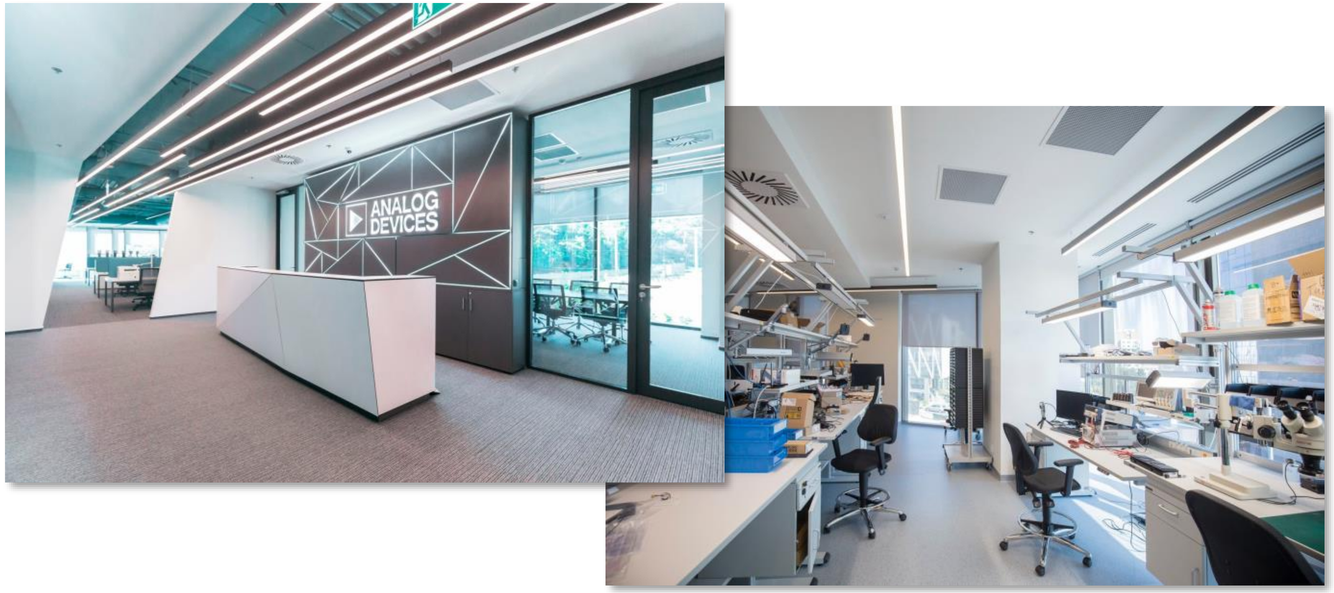
Digital Health

Industry 4.0 and Smart Energy

Immersive Consumer Experiences

# ADI Romania Design Center

- Founded in 2011
- Office 1 – UBC Riviera
  - 1,000 square meters, 100 people capacity
- Office 2 – UBC Tower
  - 1,000 square meters, 120 people capacity
- Multidisciplinary team
  - Hardware design
  - FPGA development (VHDL, Verilog)
  - Embedded software (C/C++, Linux)
  - Applications software (Python, MATLAB, C++)
  - Devops (Jenkins, Microsoft Azure, CI/CD)
  - System architecture
  - UX design
  - Program/Project management
- Project fields
  - RF Communications
  - Precision & High-Speed Instrumentation
  - Depth, Perception and Ranging (ToF, LIDAR)
  - Industrial Automation



# Agenda

- Part 1
  - Background – History and evolution of no-OS
  - What is Baremetal (no-OS)
  - What is a baremetal device driver
  - No-OS API and platforms
  - No-OS projects and examples
  - IIO Concepts
  - Takeaways
- Part 2
  - Instructor-led demo
  - Q&A session

ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

# Hands-on Workshop: Introduction to Embedded Software Applications

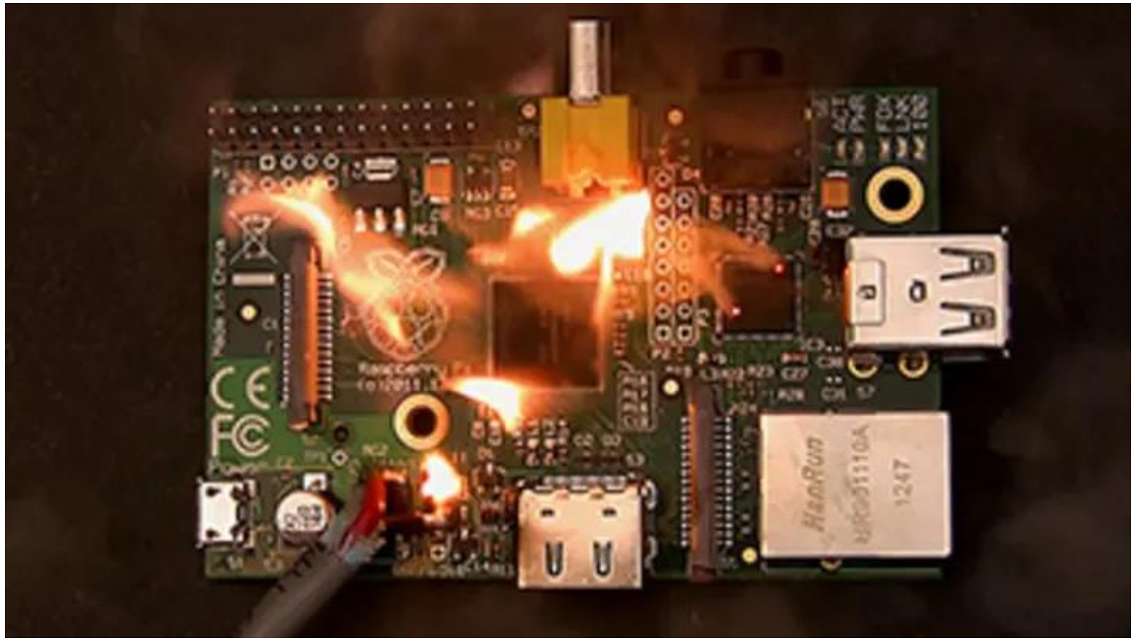
# Baremetal background

Traditionally, industry chip manufacturers would sell ICs without accompanying software:

- Chip
- Datasheet
- Pseudocode with an initialization sequence and/or sequence of data acquisition

Complex parts require complex software

ADI addressed this issue and started providing software for its parts as well.



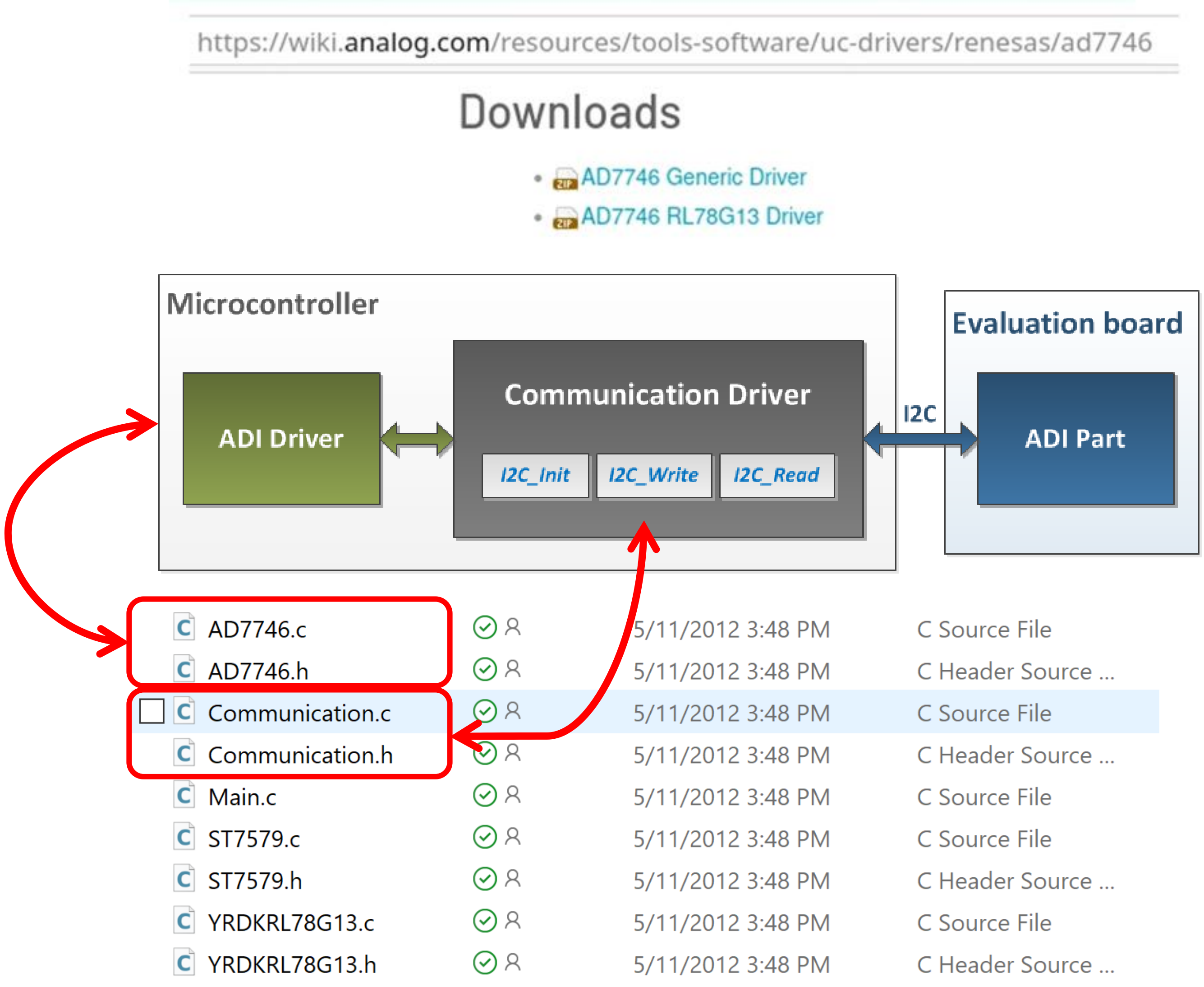
https://res.cloudinary.com/

https://www.pcbtrain.co.uk/

# Baremetal background

## Baremetal 10 years ago looked like this:

- A .zip file containing the driver files
- A .zip file containing a project



https://wiki.analog.com/resources/tools-software/uc-drivers/renesas/ad7746

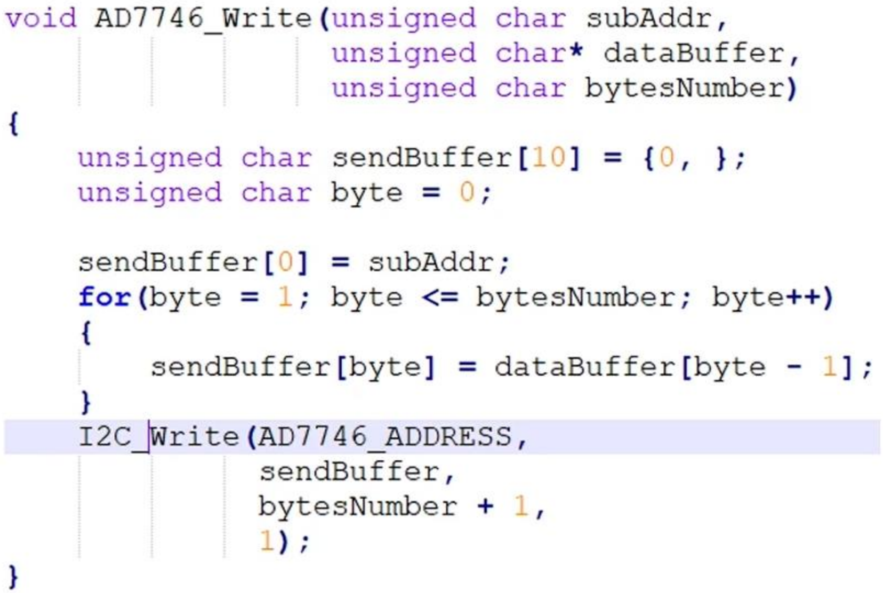
Downloads

- AD7746 Generic Driver
- AD7746 RL78G13 Driver

| | | | |
|---|---|---|---|
| AD7746.c | ✓ | 5/11/2012 3:48 PM | C Source File |
| AD7746.h | ✓ | 5/11/2012 3:48 PM | C Header Source ... |
| □ Communication.c | ✓ | 5/11/2012 3:48 PM | C Source File |
| Communication.h | ✓ | 5/11/2012 3:48 PM | C Header Source ... |
| Main.c | ✓ | 5/11/2012 3:48 PM | C Source File |
| ST7579.c | ✓ | 5/11/2012 3:48 PM | C Source File |
| ST7579.h | ✓ | 5/11/2012 3:48 PM | C Header Source ... |
| YRDKRL78G13.c | ✓ | 5/11/2012 3:48 PM | C Source File |
| YRDKRL78G13.h | ✓ | 5/11/2012 3:48 PM | C Header Source ... |

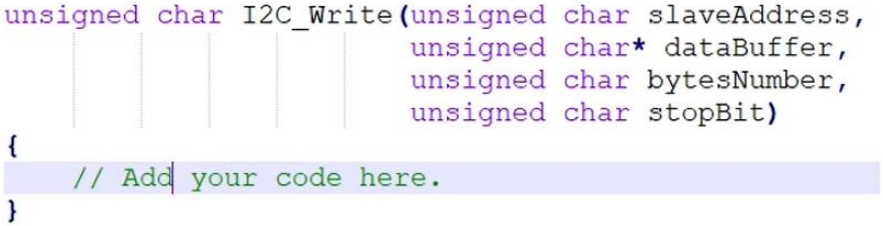
# Baremetal background

## ADI part driver:



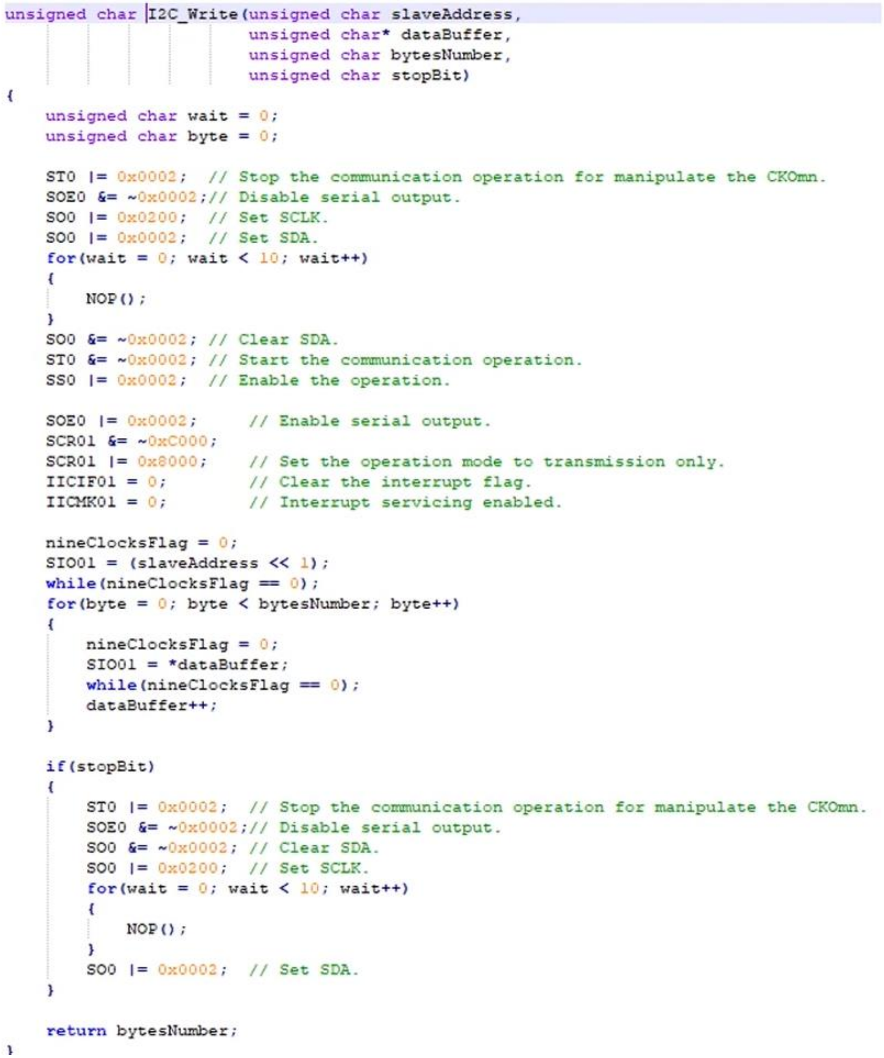
```
void AD7746_Write(unsigned char subAddr,
                  unsigned char* dataBuffer,
                  unsigned char bytesNumber)
{
    unsigned char sendBuffer[10] = {0, };
    unsigned char byte = 0;

    sendBuffer[0] = subAddr;
    for(byte = 1; byte <= bytesNumber; byte++)
    {
        sendBuffer[byte] = dataBuffer[byte - 1];
    }
    I2C_Write(AD7746_ADDRESS,
              sendBuffer,
              bytesNumber + 1,
              1);
}
```

## Communication template:



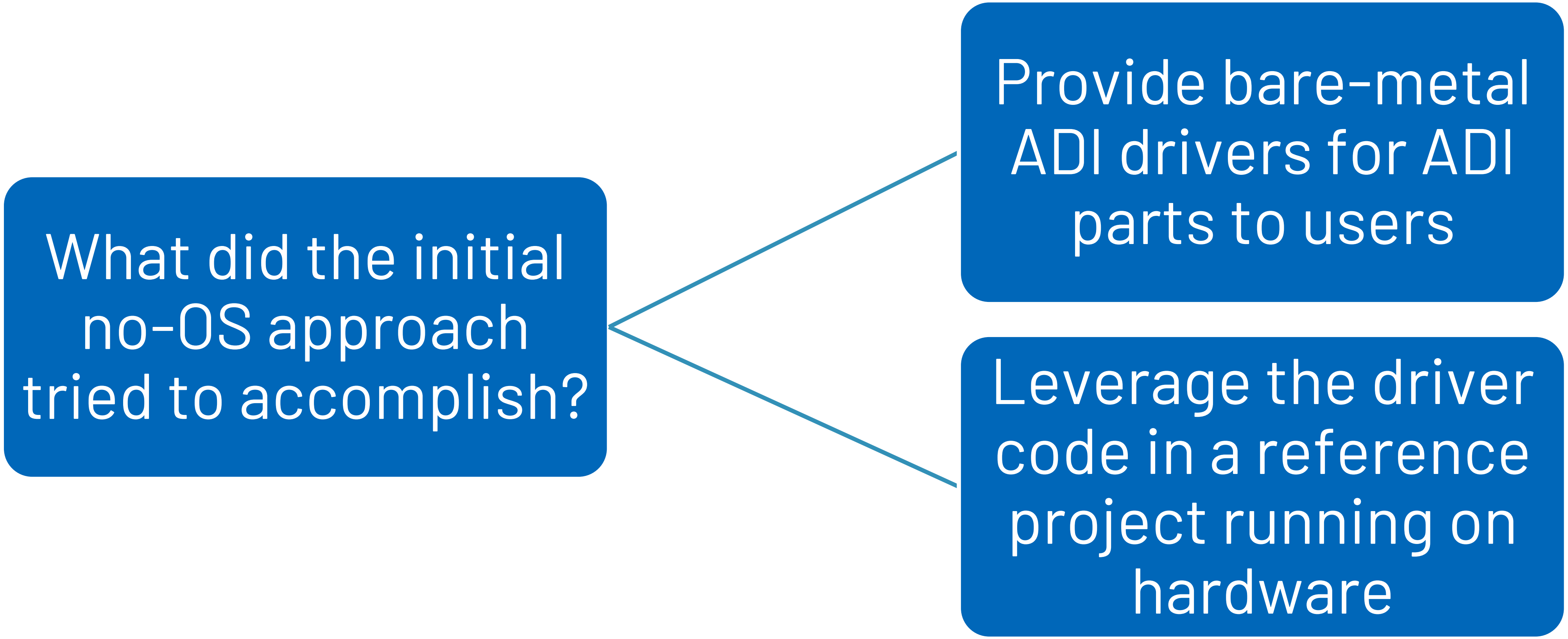
```
unsigned char I2C_Write(unsigned char slaveAddress,
                        unsigned char* dataBuffer,
                        unsigned char bytesNumber,
                        unsigned char stopBit)
{
    // Add your code here.
}
```

## Communication implementation for Renesas MCU:



```
unsigned char I2C_Write(unsigned char slaveAddress,
                        unsigned char* dataBuffer,
                        unsigned char bytesNumber,
                        unsigned char stopBit)
{
    unsigned char wait = 0;
    unsigned char byte = 0;

    ST0 |= 0x0002;  // Stop the communication operation for manipulate the CKOmn.
    SOE0 &= ~0x0002;// Disable serial output.
    SO0 |= 0x0200;  // Set SCLK.
    SO0 |= 0x0002;  // Set SDA.
    for(wait = 0; wait < 10; wait++)
    {
        NOP();
    }
    SO0 &= ~0x0002; // Clear SDA.
    ST0 &= ~0x0002; // Start the communication operation.
    SS0 |= 0x0002;  // Enable the operation.

    SOE0 |= 0x0002;     // Enable serial output.
    SCR01 &= ~0xC000;
    SCR01 |= 0x8000;    // Set the operation mode to transmission only.
    IICIF01 = 0;        // Clear the interrupt flag.
    IICMK01 = 0;        // Interrupt servicing enabled.

    nineClocksFlag = 0;
    SIO01 = (slaveAddress << 1);
    while(nineClocksFlag == 0);
    for(byte = 0; byte < bytesNumber; byte++)
    {
        nineClocksFlag = 0;
        SIO01 = *dataBuffer;
        while(nineClocksFlag == 0);
        dataBuffer++;
    }

    if(stopBit)
    {
        ST0 |= 0x0002;  // Stop the communication operation for manipulate the CKOmn.
        SOE0 &= ~0x0002;// Disable serial output.
        SO0 &= ~0x0002; // Clear SDA.
        SO0 |= 0x0200;  // Set SCLK.
        for(wait = 0; wait < 10; wait++)
        {
            NOP();
        }
        SO0 |= 0x0002;  // Set SDA.
    }

    return bytesNumber;
}
```

# Baremetal background



What did the initial no-OS approach tried to accomplish?

Provide bare-metal ADI drivers for ADI parts to users

Leverage the driver code in a reference project running on hardware

# Baremetal background - usage

## Advantages

- Driver code was MCU independent

## Disadvantages

- Customer responsibility to port reference project on a different MCU
- .zip file distribution led to no version control and code duplication

## Evolution provided:

- Provide a way for reference projects to run on multiple hardware combinations
- Provide a build system that generates binaries and run them on hardware
- Expose parts as IIO devices to PC applications
- Improve code quality

# What is no-OS?

Baremetal(No-OS) = system that does not run on a specific operating system

## No-OS

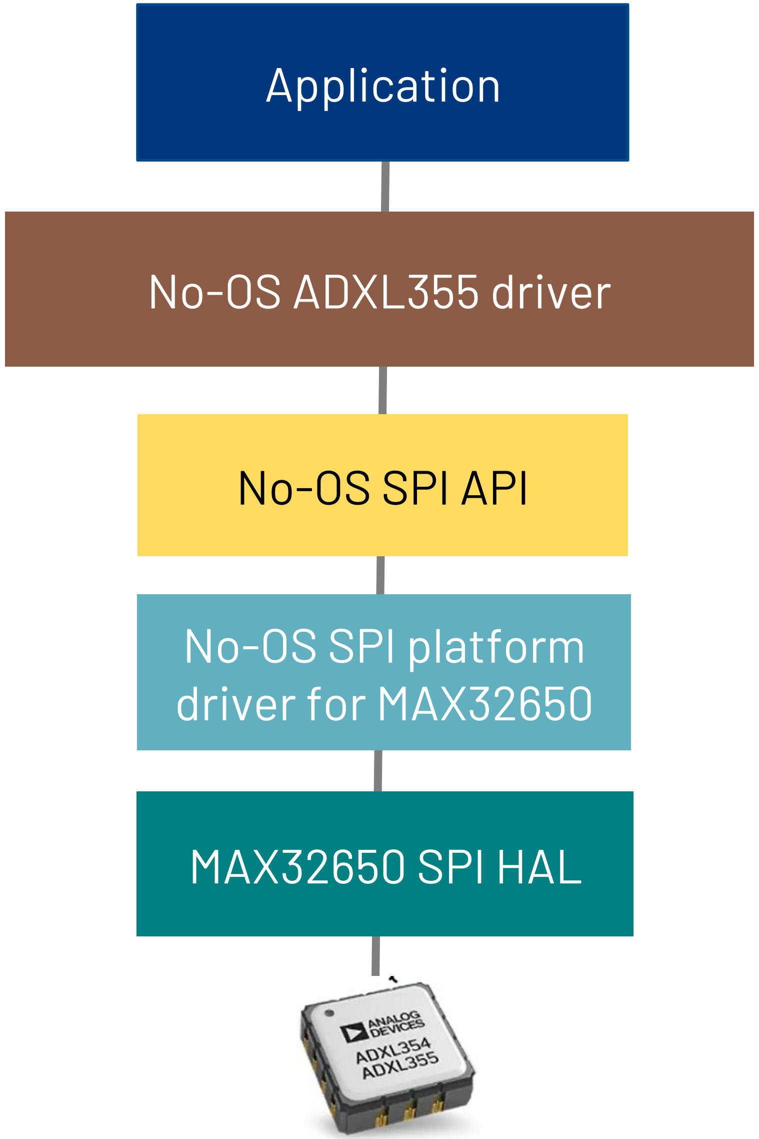
- Software framework for embedded bare-metal development
- Open-source
- ADI-BSD license
- Free
- Large collection of platform agnostic device drivers for ADI parts
- Significant collection of reference projects leveraging ADI evaluation boards
- Reference projects can run on a wide range of hardware
- Provides IIO enabled devices, making them accessible to PC applications that use libiio

Resources
https://wiki.analog.com/resources/no-os
https://github.com/analogdevicesinc/no-OS

# No-OS Device Drivers

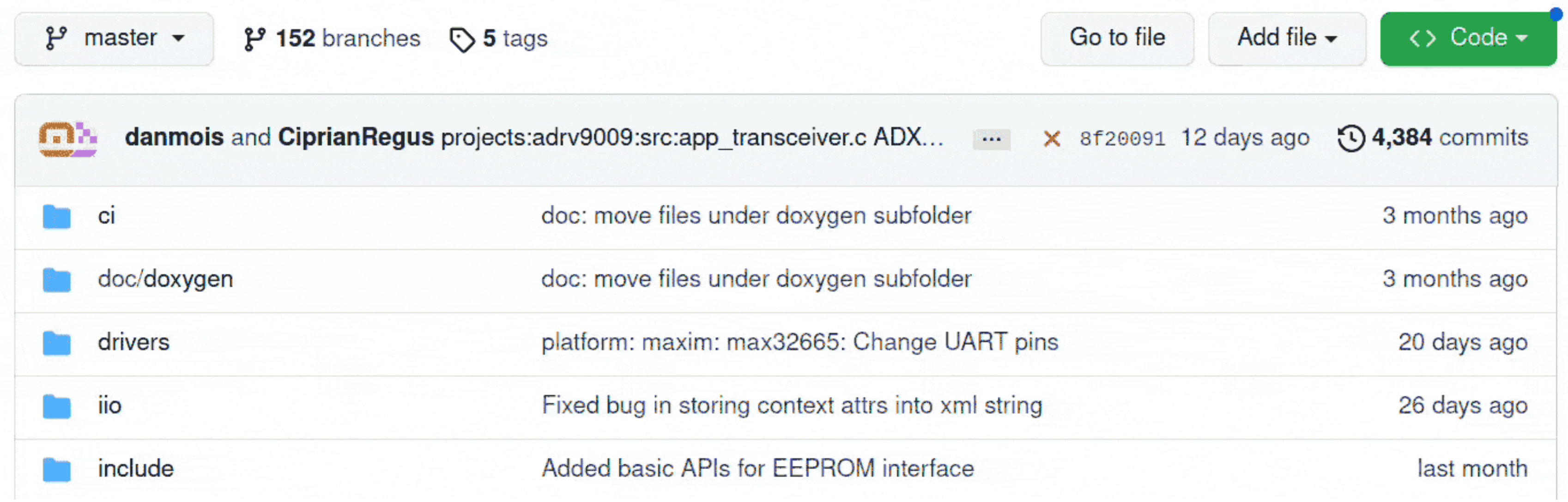
## What is a No-OS device driver?

- Implemented in C, in a .c and .h pair, stored under /drivers
- Its programming interface is directly called by the application code
- Defines its own descriptor structures and init_param
- Contains minimum init() and remove() functions
- Performs no-OS API calls, does not perform platform specific function calls, it's platform agnostic
- Software application can access hardware functionality without knowing in detail how the device operates



# No-OS Device Drivers

- *Implemented in C as a .c and .h pair, stored under drivers/*

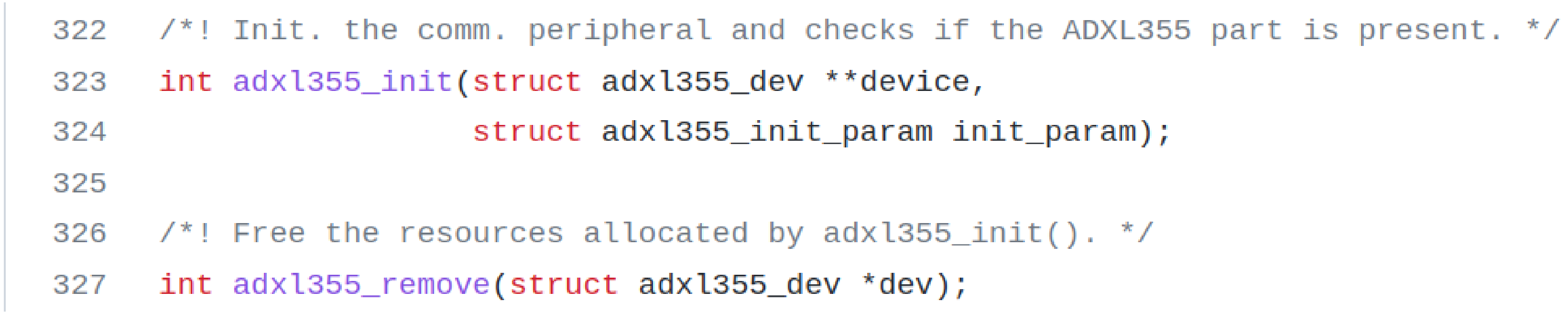


master ▾ 152 branches 5 tags Go to file Add file ▾ <> Code ▾

**danmois** and **CiprianRegus** projects:adrv9009:src:app_transceiver.c ADX... ··· ✕ 8f20091 12 days ago 4,384 commits

| | | |
|---|---|---|
| ci | doc: move files under doxygen subfolder | 3 months ago |
| doc/doxygen | doc: move files under doxygen subfolder | 3 months ago |
| drivers | platform: maxim: max32665: Change UART pins | 20 days ago |
| iio | Fixed bug in storing context attrs into xml string | 26 days ago |
| include | Added basic APIs for EEPROM interface | last month |

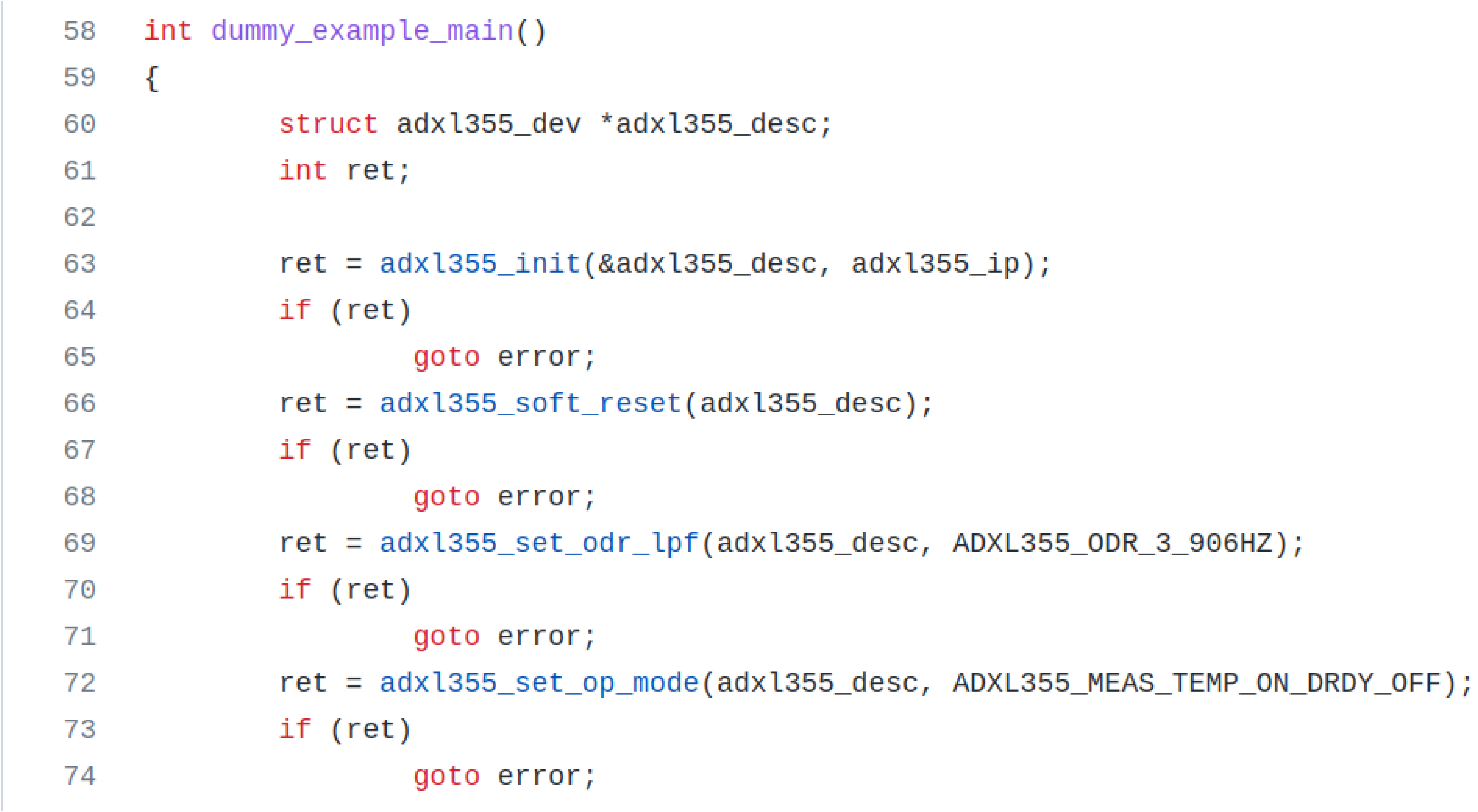
- *Consists of init() and remove() functions at minimum*



```
/*! Init. the comm. peripheral and checks if the ADXL355 part is present. */
int adxl355_init(struct adxl355_dev **device,
                 struct adxl355_init_param init_param);

/*! Free the resources allocated by adxl355_init(). */
int adxl355_remove(struct adxl355_dev *dev);
```

# No-OS Device Drivers

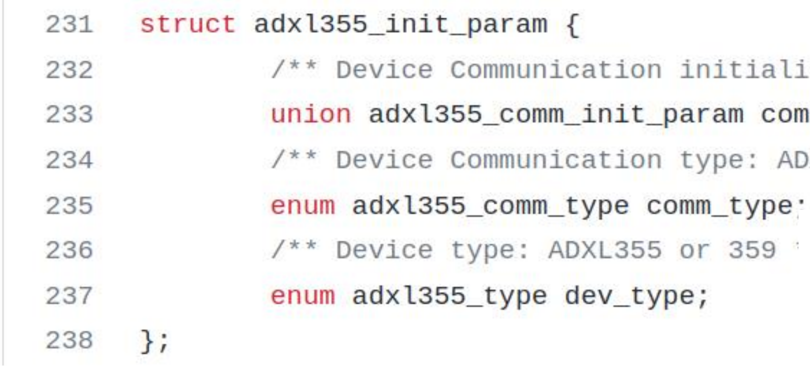
- *Its programming interface is directly called by the application code*



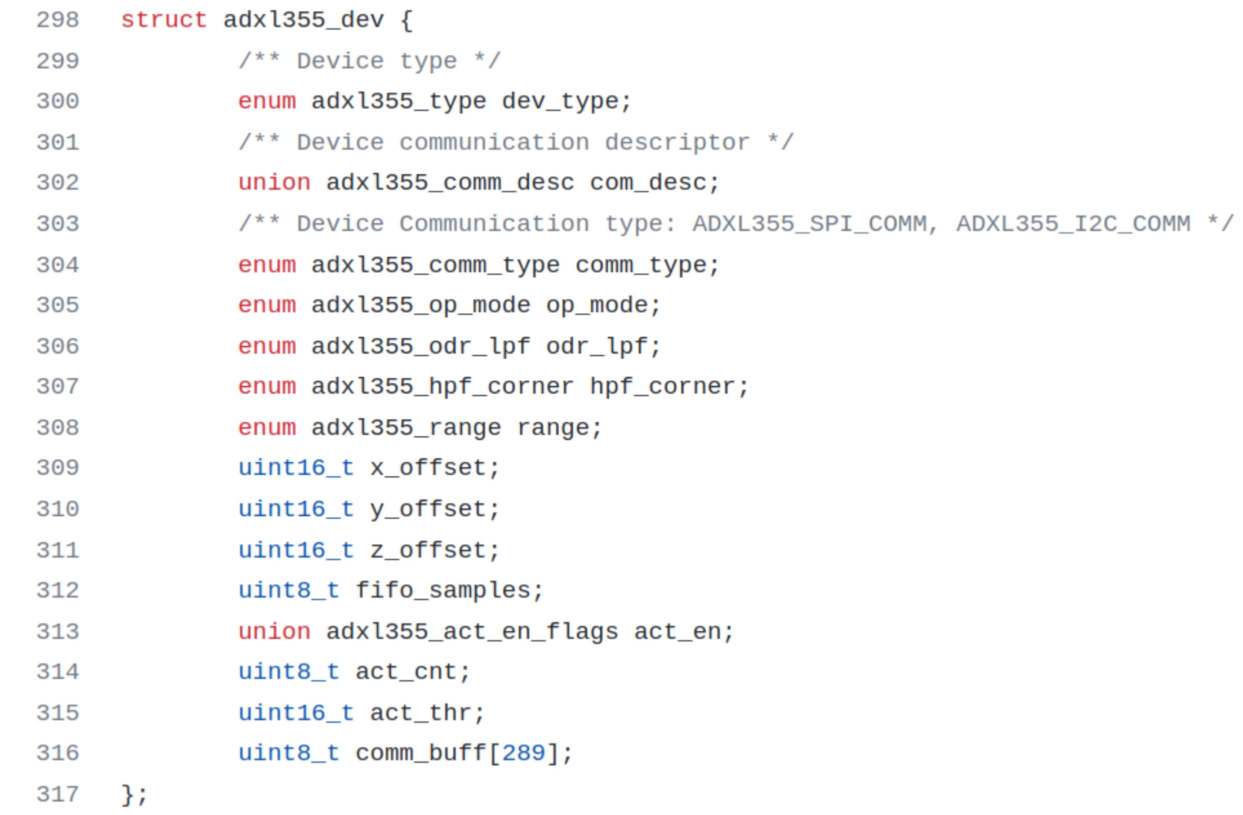
```
int dummy_example_main()
{
        struct adxl355_dev *adxl355_desc;
        int ret;

        ret = adxl355_init(&adxl355_desc, adxl355_ip);
        if (ret)
                goto error;
        ret = adxl355_soft_reset(adxl355_desc);
        if (ret)
                goto error;
        ret = adxl355_set_odr_lpf(adxl355_desc, ADXL355_ODR_3_906HZ);
        if (ret)
                goto error;
        ret = adxl355_set_op_mode(adxl355_desc, ADXL355_MEAS_TEMP_ON_DRDY_OFF);
        if (ret)
                goto error;
```

# No-OS Device Drivers

- *Defines its own init_param and descriptor structures*



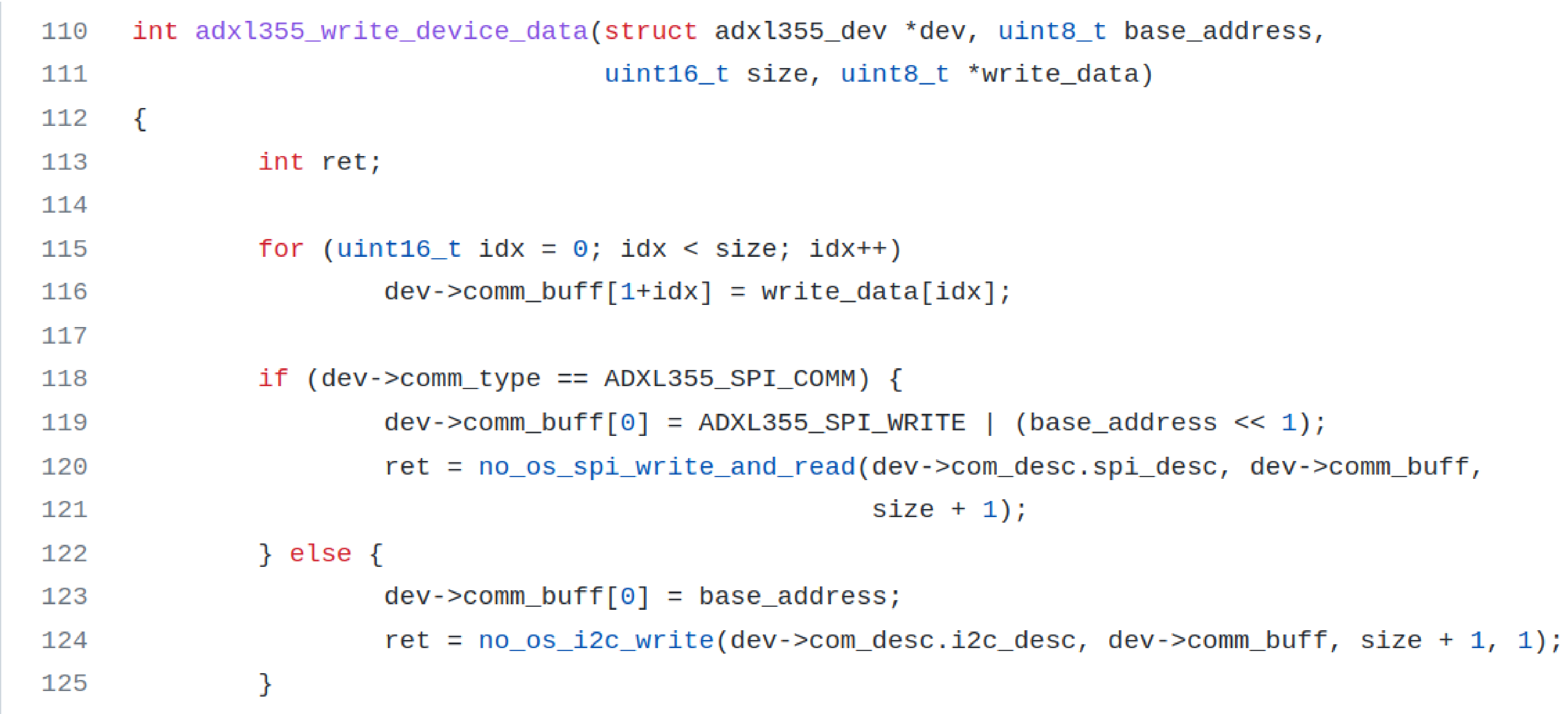
```
struct adxl355_init_param {
	/** Device Communication initialization structure: either SPI or I2C */
	union adxl355_comm_init_param comm_init;
	/** Device Communication type: ADXL355_SPI_COMM, ADXL355_I2C_COMM */
	enum adxl355_comm_type comm_type;
	/** Device type: ADXL355 or 359
	enum adxl355_type dev_type;
};
```



```
struct adxl355_dev {
	/** Device type */
	enum adxl355_type dev_type;
	/** Device communication descriptor */
	union adxl355_comm_desc com_desc;
	/** Device Communication type: ADXL355_SPI_COMM, ADXL355_I2C_COMM */
	enum adxl355_comm_type comm_type;
	enum adxl355_op_mode op_mode;
	enum adxl355_odr_lpf odr_lpf;
	enum adxl355_hpf_corner hpf_corner;
	enum adxl355_range range;
	uint16_t x_offset;
	uint16_t y_offset;
	uint16_t z_offset;
	uint8_t fifo_samples;
	union adxl355_act_en_flags act_en;
	uint8_t act_cnt;
	uint16_t act_thr;
	uint8_t comm_buff[289];
};
```

# No-OS Device Drivers

- *Performs no-OS API calls, does not perform platform specific function calls, therefore it is platform agnostic*

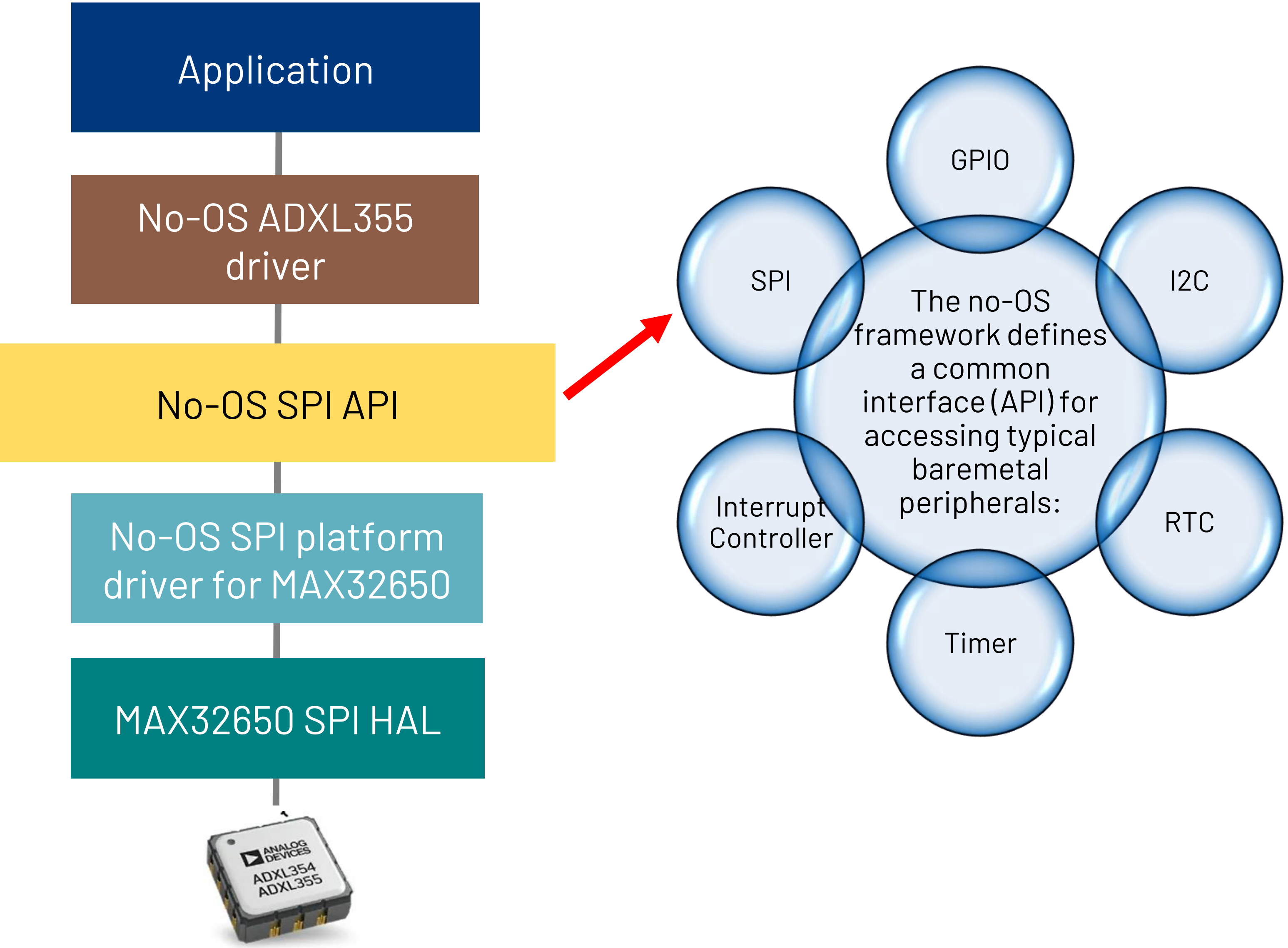


```c
int adxl355_write_device_data(struct adxl355_dev *dev, uint8_t base_address,
                              uint16_t size, uint8_t *write_data)
{
        int ret;

        for (uint16_t idx = 0; idx < size; idx++)
                dev->comm_buff[1+idx] = write_data[idx];

        if (dev->comm_type == ADXL355_SPI_COMM) {
                dev->comm_buff[0] = ADXL355_SPI_WRITE | (base_address << 1);
                ret = no_os_spi_write_and_read(dev->com_desc.spi_desc, dev->comm_buff,
                                               size + 1);
        } else {
                dev->comm_buff[0] = base_address;
                ret = no_os_i2c_write(dev->com_desc.i2c_desc, dev->comm_buff, size + 1, 1);
        }
```

# No-OS API

## API – Application Programming Interface

- No-OS API allows any code that uses it (driver or app) to remain platform agnostic



Application

No-OS ADXL355 driver

No-OS SPI API

No-OS SPI platform driver for MAX32650

MAX32650 SPI HAL

The no-OS framework defines a common interface (API) for accessing typical baremetal peripherals:

GPIO

SPI

I2C

Interrupt Controller

RTC

Timer

# No-OS Platforms

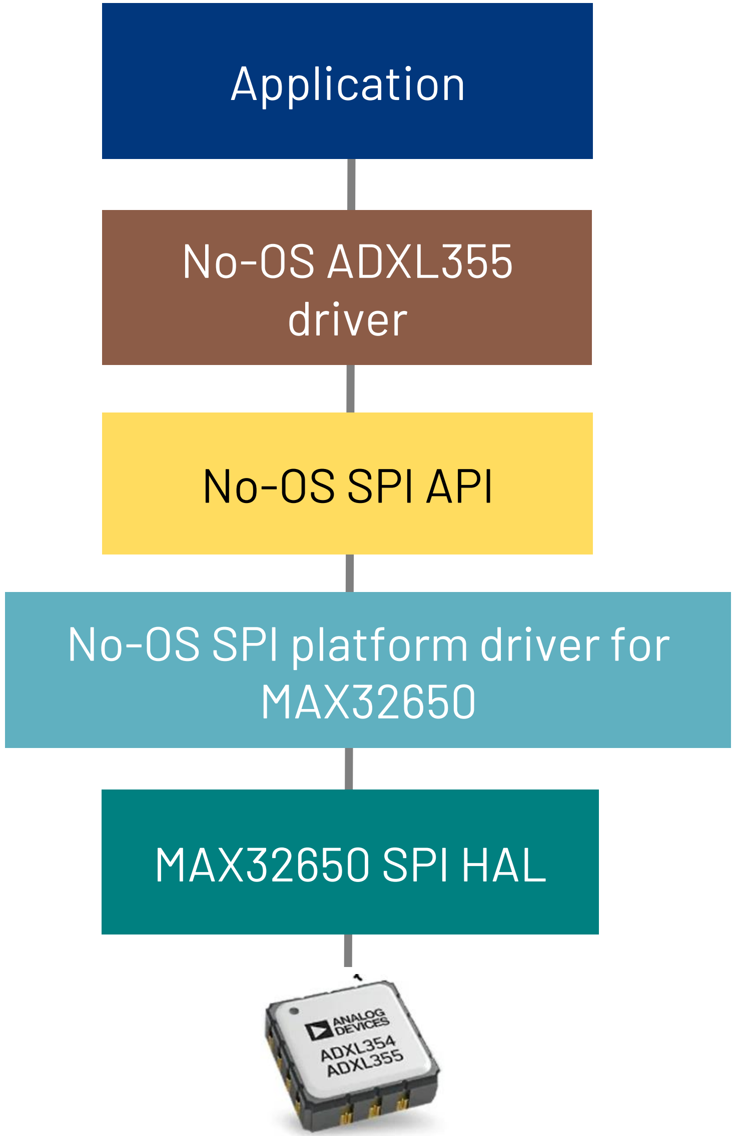
Platform drivers – implementations of peripheral related no-OS API on a specific platform

Platform drivers use vendor HAL - Hardware Abstraction Layer

No-OS platform drivers are implementations of peripheral no-OS API on a particular platform.

No-OS modularity allow it to run a lot of its code on different platforms like:

- Xilinx (Zynq7000, ZynqMP, Microblaze)
- Maxim (32650, 32655, 32660, 32670, 78000)
  ADuCM (3029)
- STM32 (almost any)
- RaspberryPi Pico
- Mbed



Application

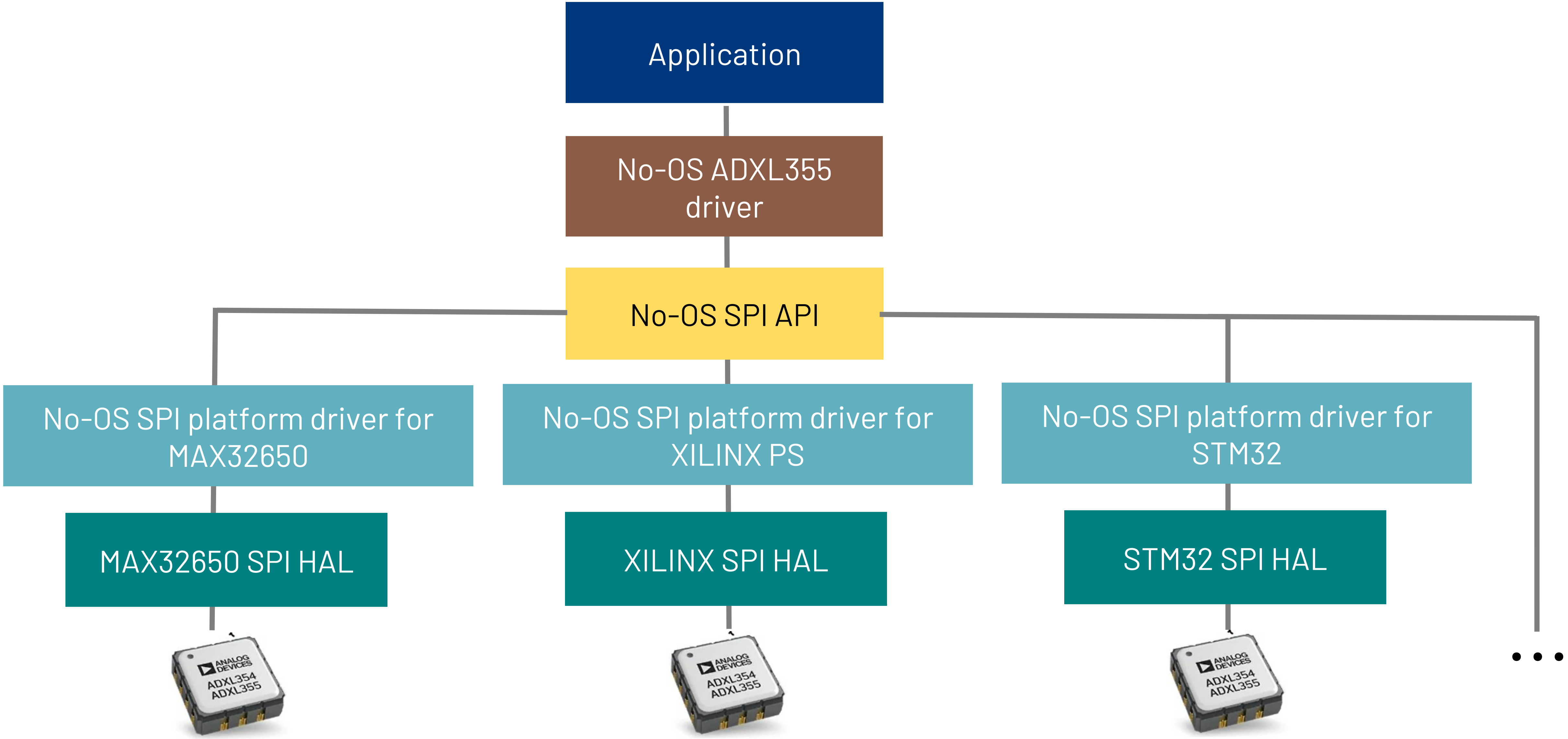
No-OS ADXL355 driver

No-OS SPI API

No-OS SPI platform driver for MAX32650

MAX32650 SPI HAL

# No-OS Platforms


Application
No-OS ADXL355 driver
No-OS SPI API
No-OS SPI platform driver for MAX32650
No-OS SPI platform driver for XILINX PS
No-OS SPI platform driver for STM32
MAX32650 SPI HAL
XILINX SPI HAL
STM32 SPI HAL
ANALOG DEVICES ADXL354 ADXL355
ANALOG DEVICES ADXL354 ADXL355
ANALOG DEVICES ADXL354 ADXL355
• • •

# No-OS Projects

## An application can be built, run and debugged on hardware

- Located under projects/
- It has a main() function
- Uses drivers/ and drivers/platforms directories
- Uses no-OS API
- Uses various libraries
- User interaction – serial, iio-oscilloscope
- makefiles

## Project hardware typically is made of

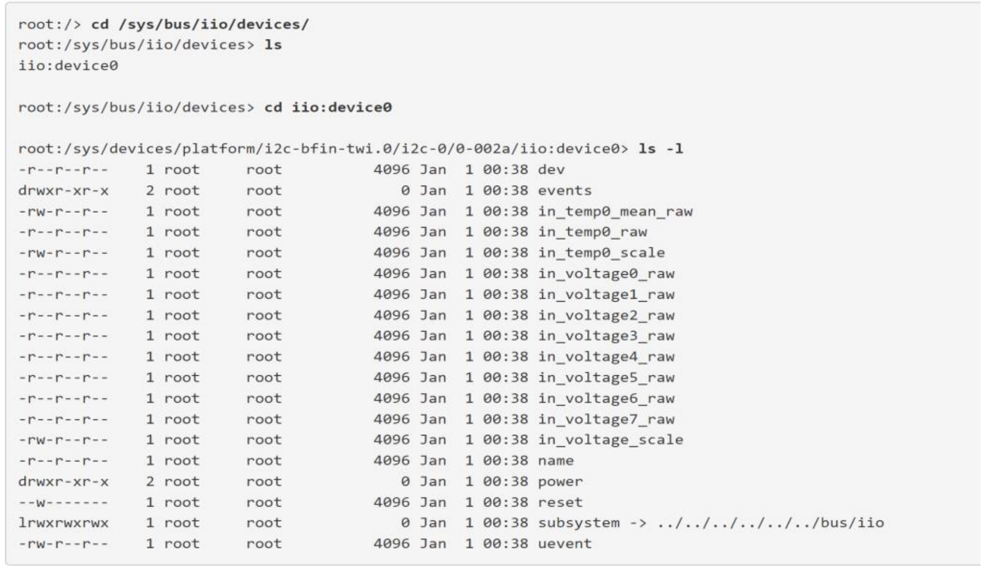
- An evaluation board
- A carrier board

## No-OS projects are used for

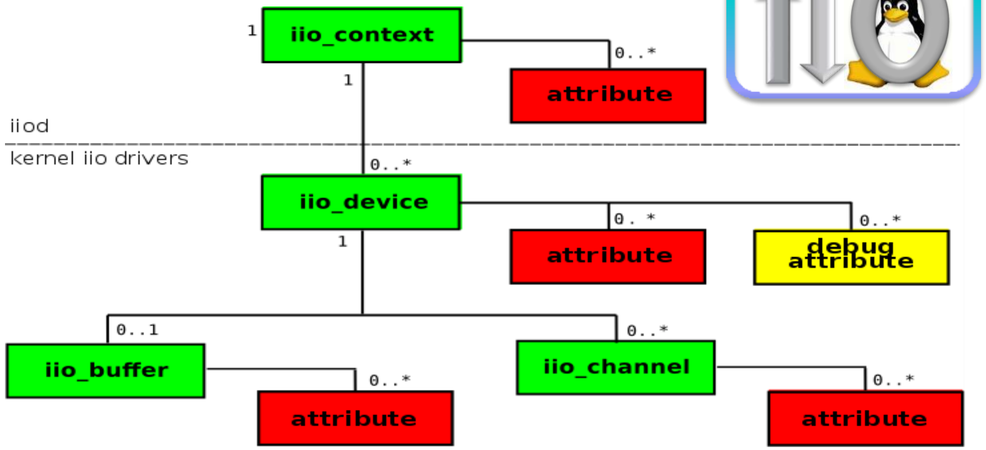
- ADI parts evaluation
- Starting development based on a no-OS project

# IIO Concepts

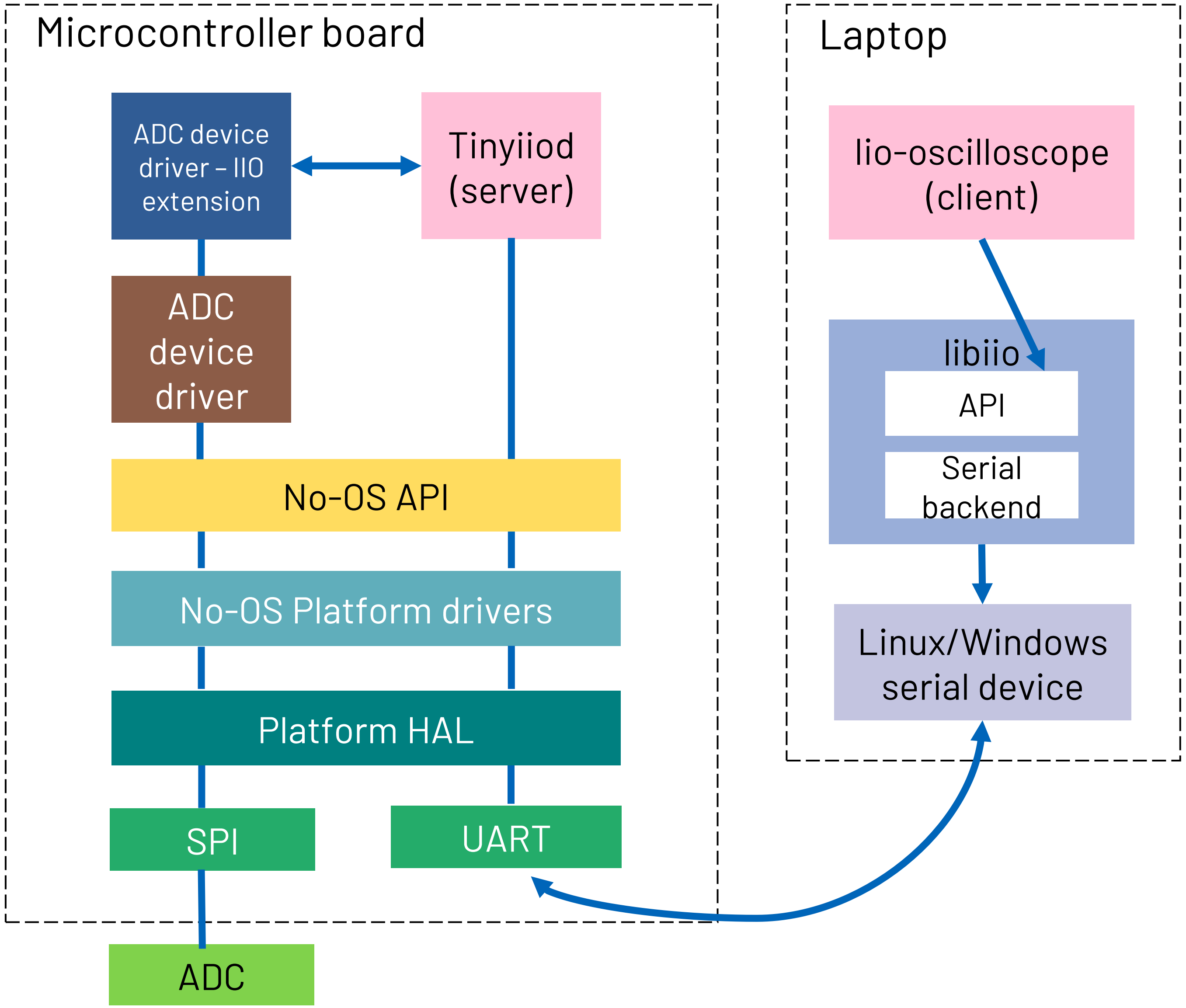
- The Linux Industrial I/O (IIO) subsystem is intended to provide support for devices that, in some sense, are analog-to-digital or digital-to-analog converters
  - Devices that fall into this category are:
    - ADCs
    - DACs
    - Accelerometers, gyros, IMUs
    - Capacitance-to-Digital converters (CDCs)
    - Pressure, temperature, and light sensors, etc.
    - RF Transceivers (like the AD9361 / AD9364 / AD9371 / ADRV9009)
  - Can be used on ADCs ranging from a 1MSPS SoC ADC to >5 GSPS ADCs



```
root:/> cd /sys/bus/iio/devices/
root:/sys/bus/iio/devices> ls
iio:device0

root:/sys/bus/iio/devices> cd iio:device0

root:/sys/devices/platform/i2c-bfin-twi.0/i2c-0/0-002a/iio:device0> ls -l
-r--r--r--    1 root     root          4096 Jan  1 00:38 dev
drwxr-xr-x    2 root     root             0 Jan  1 00:38 events
-rw-r--r--    1 root     root          4096 Jan  1 00:38 in_temp0_mean_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_temp0_raw
-rw-r--r--    1 root     root          4096 Jan  1 00:38 in_temp0_scale
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage0_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage1_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage2_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage3_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage4_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage5_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage6_raw
-r--r--r--    1 root     root          4096 Jan  1 00:38 in_voltage7_raw
-rw-r--r--    1 root     root          4096 Jan  1 00:38 in_voltage_scale
-r--r--r--    1 root     root          4096 Jan  1 00:38 name
drwxr-xr-x    2 root     root             0 Jan  1 00:38 power
--w-------    1 root     root          4096 Jan  1 00:38 reset
lrwxrwxrwx    1 root     root             0 Jan  1 00:38 subsystem -> ../../../../../../bus/iio
-rw-r--r--    1 root     root          4096 Jan  1 00:38 uevent
```



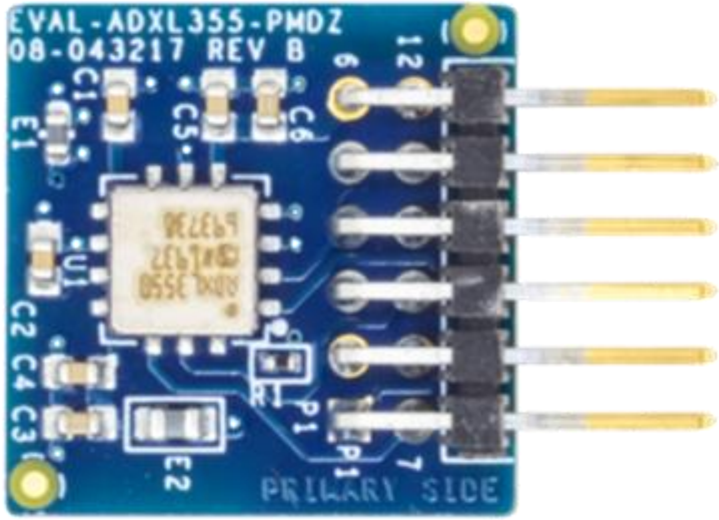
# libIIO and applications



# ADXL355 part

Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers

- ADXL355 digital output features
  - Digital SPI and $I^2C$ interfaces supported
  - 20-bit ADC
  - Data interpolation routine for synchronous sampling
  - Programmable high- and low-pass digital filters
- 0 *g* offset vs. temperature (all axes): 0.15 m*g*/°C maximum
- $V_{SUPPLY}$ with internal regulators: 2.25 V to 3.6



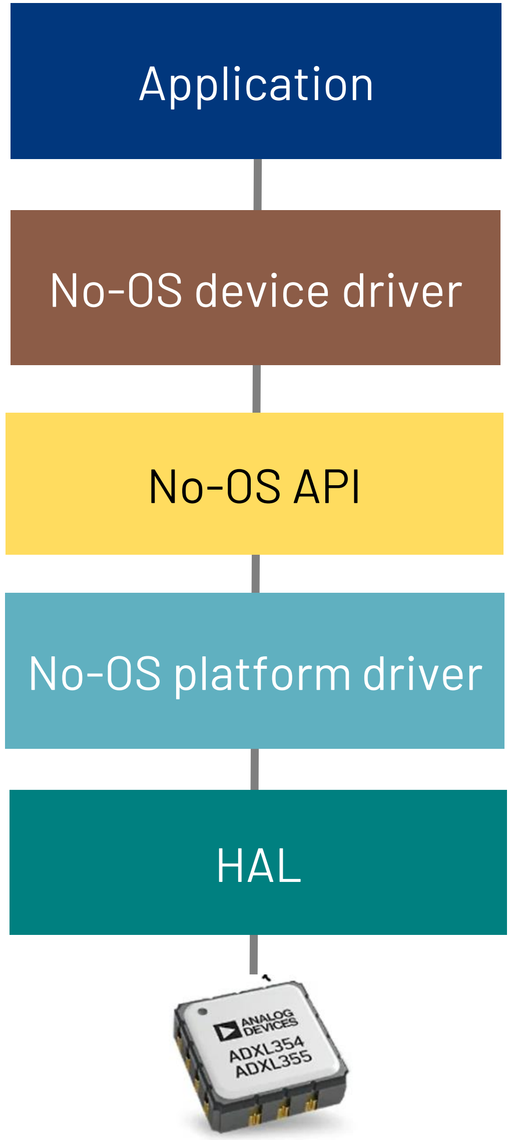
# No-OS Device Drivers - takeaways

No-OS device drivers are bits of platform agnostic code

Device drivers can be used to control ADI parts purely from software

No-OS platform drivers are implementations of peripheral no-OS API on a particular platform

No-OS has a build system for projects to be built and run on hardware



ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

# Hands-on Workshop: Introduction to no-OS Embedded Applications

## Part 2

Hands on Lab

# Hands On Lab

## By the end of this lab, you will learn:

- How to build a no-OS project
- How to run a no-OS application on a specific hardware
- How to use a no-OS demo running on a specific hardware
- How to modify written code to do a certain task

http://www.iconarchive.com/show/noto-emoji-objects-icons-by-google/62807-radio-icon.html
http://www.streamlineicons.com
http://pixelkit.com

# Hands On Lab – Getting started steps

Wait for your Raspberry Pi 5 workstation to boot.

Open a terminal by clicking the *Terminal Emulator* icon.

Before building a project, set the MAXIM_LIBRARIES, PLATFORM and TARGET environment variables with the following command:

```
export MAXIM_LIBRARIES=~/workshop_baremetal/MAX78000SDK/Libraries
PLATFORM=maxim TARGET=max78000
```

# Hands On Lab

Equipment used:

- EVAL-ADXL355-PMDZ
- MAX78000FTHR
- USB cable

**Demo 1**

Prints a “Hello World” message over the serial UART.

**Demo 2**

Reads temperature from the internal temperature sensor of the ADXL355 device.

**Demo 3**

Reads temperature and the acceleration values from the ADXL355 and converts the data from raw values into user readable values.

**Demo 4**

Accelerometer-enabled game that lets you place components on a circuit by physically tilting the accelerometer.

http://www.iconarchive.com/show/noto-emoji-objects-icons-by-google/62807-radio-icon.html
http://www.streamlineicons.com
http://pixelkit.com

# Hands On Lab

## Demo 1

Prints a "Hello World" message over the serial UART.

## Procedure:

- **Move to the no-OS workshop project location:**
  - cd ~/workshop_baremetal/no-OS/projects/workshop
- **Reset the workspace:**
  - make reset
- **Build the first project example:**
  - make EXAMPLE=example_1
- **Connect the MAX78000FTHR to one of the Raspberry Pi 5's USB ports using a USB cable.**
- **Open a new Terminal and start serial communication by running the command:**
  - picocom –b 57600 /dev/ttyACM0
- **In the initial terminal write the following command to program MAX78000:**
  - make EXAMPLE=example_1 run

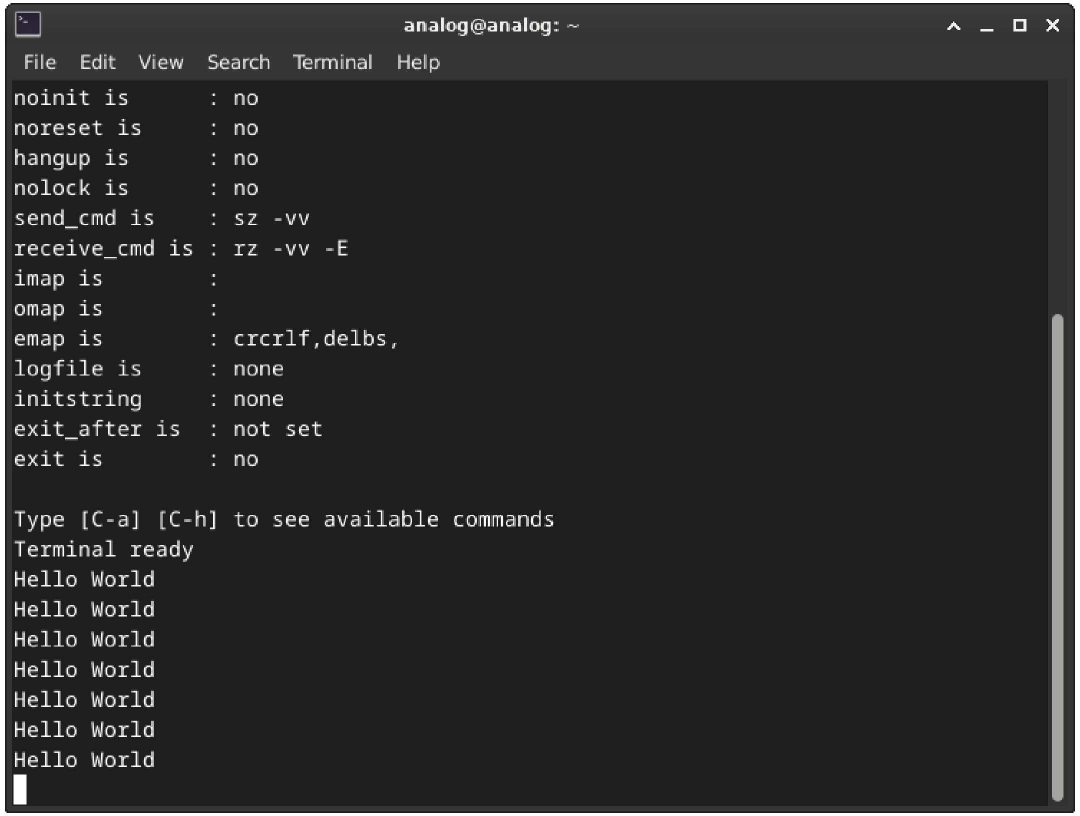
# Hands On Lab

### Demo 1

Prints a “Hello World” message over the serial UART.

## Procedure:

- You will see a new “Hello World” message every second on the terminal running picocom



```
noinit is       : no
noreset is      : no
hangup is       : no
nolock is       : no
send_cmd is     : sz -vv
receive_cmd is  : rz -vv -E
imap is         :
omap is         :
emap is         : crcrlf,delbs,
logfile is      : none
initstring      : none
exit_after is   : not set
exit is         : no

Type [C-a] [C-h] to see available commands
Terminal ready
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

# Hands On Lab

## Demo 2

Reads temperature from the internal temperature sensor of the ADXL355 device.

EVAL-ADXL355-PMDZ pinout:

| Pin Number | Pin Function | Mnemonic |
|---|---|---|
| Pin 1 | Chip Select | CS |
| Pin 2 | Master Out Slave In | MOSI |
| Pin 3 | Master In Slave Out | MISO |
| Pin 4 | Serial Clock | SCLK |
| Pin 5 | Digital Ground | DGND |
| Pin 6 | Digital Power | VDD |
| Pin 7 | Interrupt 1 | INT1 |
| Pin 8 | Not Connected | NC |
| Pin 9 | Interrupt 2 | INT2 |
| Pin 10 | Data Ready | DRDY |
| Pin 11 | Digital Ground | DGND |
| Pin 12 | Digital Power | VDD |

## Procedure:

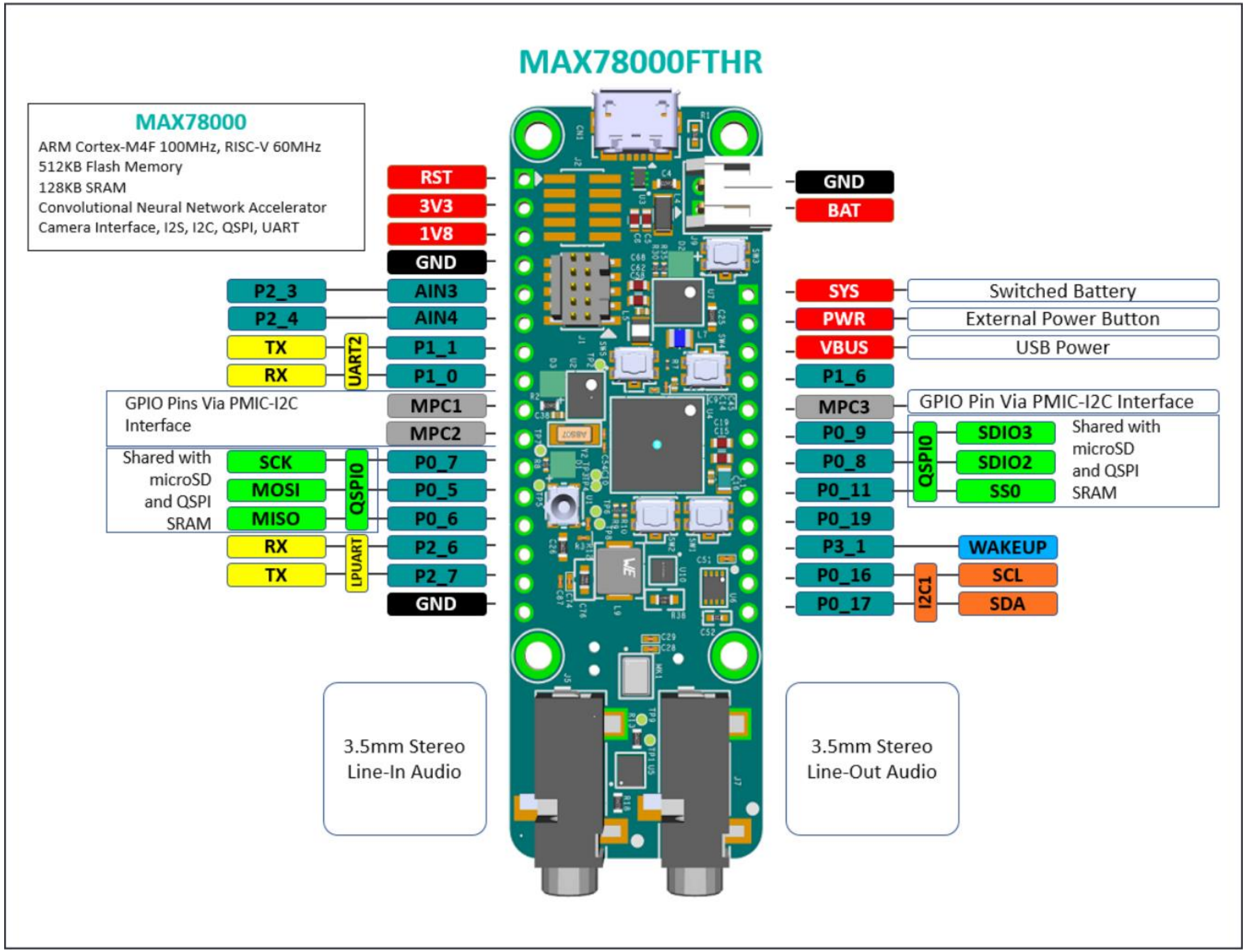
- **Disconnect the MAX78000FTHR USB cable from the Raspberry Pi 5.**
- **Connect the EVAL-ADXL355-PMDZ to the MAX78000FTHR by using the information below:**
  - EVAL-ADXL355-PMDZ Pinout
  - MAX78000FTHR Pinout
  - Pin correspondence table

# Hands On Lab

## Demo 2

Reads temperature from the internal temperature sensor of the ADXL355 device.

MAX78000FTHR Pinout:


MAX78000FTHR
MAX78000
ARM Cortex-M4F 100MHz, RISC-V 60MHz
512KB Flash Memory
128KB SRAM
Convolutional Neural Network Accelerator
Camera Interface, I2S, I2C, QSPI, UART
RST
3V3
1V8
GND
P2_3
AIN3
P2_4
AIN4
TX
P1_1
RX
P1_0
UART2
GPIO Pins Via PMIC-I2C Interface
MPC1
MPC2
Shared with microSD and QSPI SRAM
SCK
P0_7
MOSI
P0_5
MISO
P0_6
QSPI0
RX
P2_6
TX
P2_7
LPUART
GND
GND
BAT
SYS
Switched Battery
PWR
External Power Button
VBUS
USB Power
P1_6
MPC3
GPIO Pin Via PMIC-I2C Interface
P0_9
SDIO3
P0_8
SDIO2
P0_11
SS0
QSPI0
Shared with microSD and QSPI SRAM
P0_19
P3_1
WAKEUP
P0_16
SCL
P0_17
SDA
I2C1
3.5mm Stereo Line-In Audio
3.5mm Stereo Line-Out Audio

# Hands On Lab

## Demo 2

Reads temperature from the internal temperature sensor of the ADXL355 device.

Pin correspondence table:

| MAX78000FTHR | Signal | EVAL-ADXL355-PMDZ |
|---|---|---|
| 3V3 | Digital power | 6 or 12 |
| GND | Digital ground | 5 or 11 |
| P0_11/SS0 | SPI Chip Select | 1 |
| P0_5/MOSI | SPI Master Out Slave In | 2 |
| P0_6/MISO | SPI Master In Slave Out | 3 |
| P0_7/SCK | SPI Serial Clock | 4 |

## Procedure:

- **Disconnect the MAX78000FTHR USB cable from the Raspberry Pi 5.**
- **Connect the EVAL-ADXL355-PMDZ to the MAX78000FTHR by using the information below:**
  - EVAL-ADXL355-PMDZ Pinout
  - MAX78000FTHR Pinout
  - Pin correspondence table

# Hands On Lab

## Demo 2

Reads temperature from the internal temperature sensor of the ADXL355 device

## Procedure:

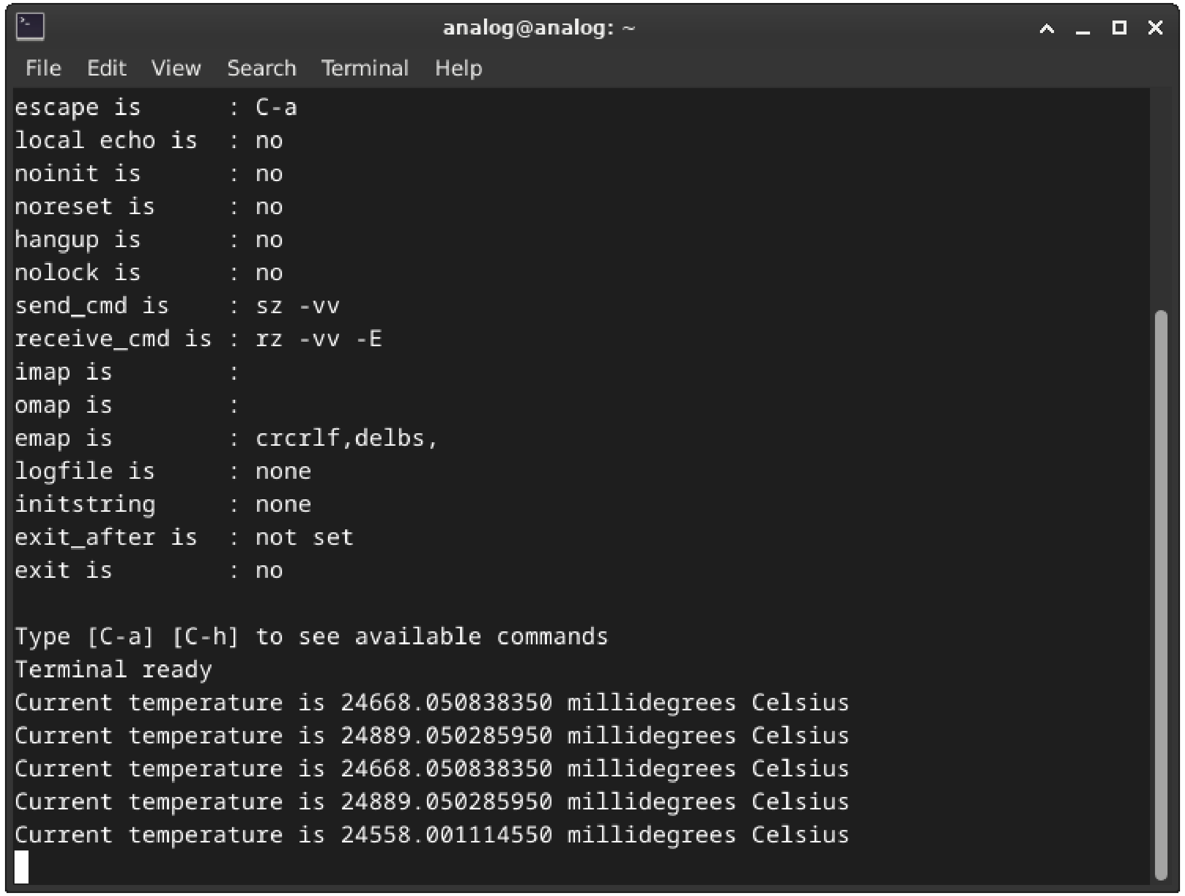
- Plug the MAX78000FTHR into one of the Raspberry Pi 5's USB ports using the USB cable.
- **Go to the no-OS workshop project location and reset the workspace**:
  - cd ~/workshop_baremetal/no-OS/projects/workshop
  - make reset
- **Build the example**:
  - make EXAMPLE=example_2
- **Load the example onto the board**:
  - make EXAMPLE=example_2 run

# Hands On Lab

## Demo 2

Reads temperature from the internal temperature sensor of the ADXL355 device

Serial terminal output:



```
escape is       : C-a
local echo is   : no
noinit is       : no
noreset is      : no
hangup is       : no
nolock is       : no
send_cmd is     : sz -vv
receive_cmd is  : rz -vv -E
imap is         :
omap is         :
emap is         : crcrlf,delbs,
logfile is      : none
initstring      : none
exit_after is   : not set
exit is         : no

Type [C-a] [C-h] to see available commands
Terminal ready
Current temperature is 24668.050838350 millidegrees Celsius
Current temperature is 24889.050285950 millidegrees Celsius
Current temperature is 24668.050838350 millidegrees Celsius
Current temperature is 24889.050285950 millidegrees Celsius
Current temperature is 24558.001114550 millidegrees Celsius
```

# Hands On Lab

**Demo 2**

Reads temperature from the internal temperature sensor of the ADXL355 device

## Challenge:

- Change the current format of the printed temperature from millidegrees to degrees.
- **Example:**
  - the current format: *27545.032056750 millidegrees*
  - the new format: *27.54 degrees*

# Hands On Lab

## Demo 3

Read temperature from the ADXL355 internal temperature sensor and convert it from raw values into user readable values.

## Procedure:

- **Change the working directory and reset the workspace:**
  - cd ~/workshop_baremetal/no-OS/projects/workshop
  - make reset
- **Build the example**:
  - make EXAMPLE=example_3
- **Load the example onto the board:**
  - make EXAMPLE=example_3 run

# Hands On Lab

## Demo 3

Read temperature from the ADXL355 internal temperature sensor and convert it from raw values into user readable values.

## Challenge:

- **Challenge:** compute the temperature and the accelerations from the raw values.
- **Hint:** For temperature you need to compute the *temp_dividend* and *temp_divisor.*
- For accelerometer values you need to compute the *x_dividend*, *y_dividend*, *z_dividend* and *accel_divisor.*

**The formula for the temperature:**

$$TEMPERATURE = (RAW + OFFSET) \cdot SCALE$$

$$TEMPERATURE = \left(RAW + \frac{OFFSET}{OFFSET\_DIV}\right) \cdot \frac{SCALE\_FACTOR}{SCALE\_FACTOR\_DIV}$$

$$\boxed{TEMPERATURE = \frac{(RAW \cdot OFFSET\_DIV + OFFSET) \cdot SCALE\_FACTOR}{OFFSET\_DIV \cdot SCALE\_FACTOR\_DIV}}$$

**The formula for the acceleration:**

$$ACCELERATION = RAW \cdot SCALE$$

$$\boxed{ACCELERATION = \frac{RAW \cdot SCALE\_FACTOR\_MUL}{SCALE\_FACTOR\_DIV}}$$

# Hands On Lab

## Demo 3

Read temperature from the ADXL355 internal temperature sensor and convert it from raw values into user readable values.

Parameter correspondence table:

| PARAMETER | VALUE |
|---|---|
| TEMPERATURE OFFSET | - 2111.25 |
| TEMPERATURE SCALE | - 110.497238 |
| ACCELERATION SCALE | 0.00003824593 |

## Challenge:

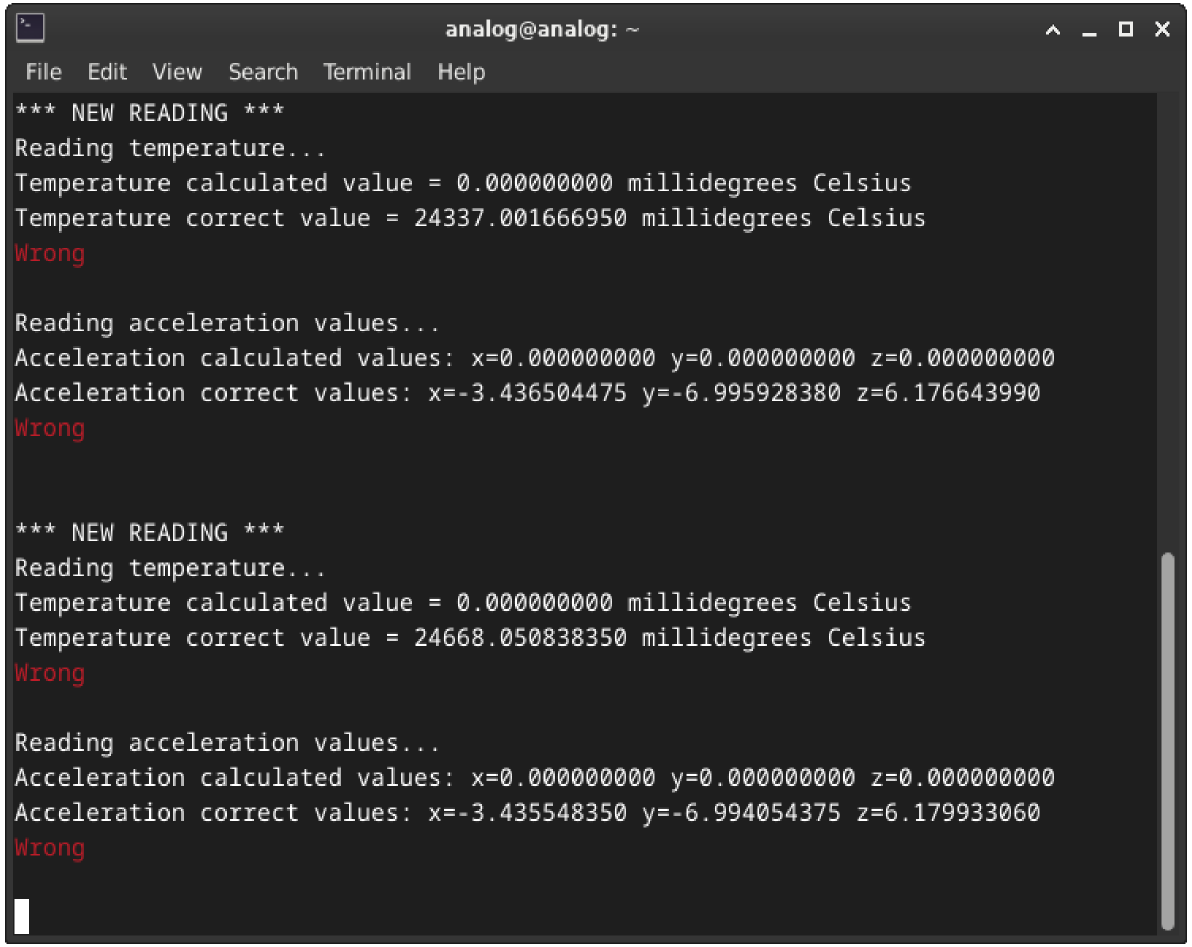
Macro correspondence table:

| MACRO | VALUE |
|---|---|
| ADXL355_TEMP_OFFSET | - 211125 |
| ADXL355_TEMP_OFFSET_DIV | 100 |
| ADXL355_TEMP_SCALE_FACTOR | -110497238 |
| ADXL355_TEMP_SCALE_FACTOR_DIV | 1000000 |
| ADXL355_ACC_SCALE_FACTOR_MUL | 38245 |
| ADXL355_ACC_SCALE_FACTOR_DIV | 1000000000 |

# Hands On Lab

## Demo 3

Read temperature and acceleration from the ADXL355 internal temperature sensor and convert it from raw values into user readable values.

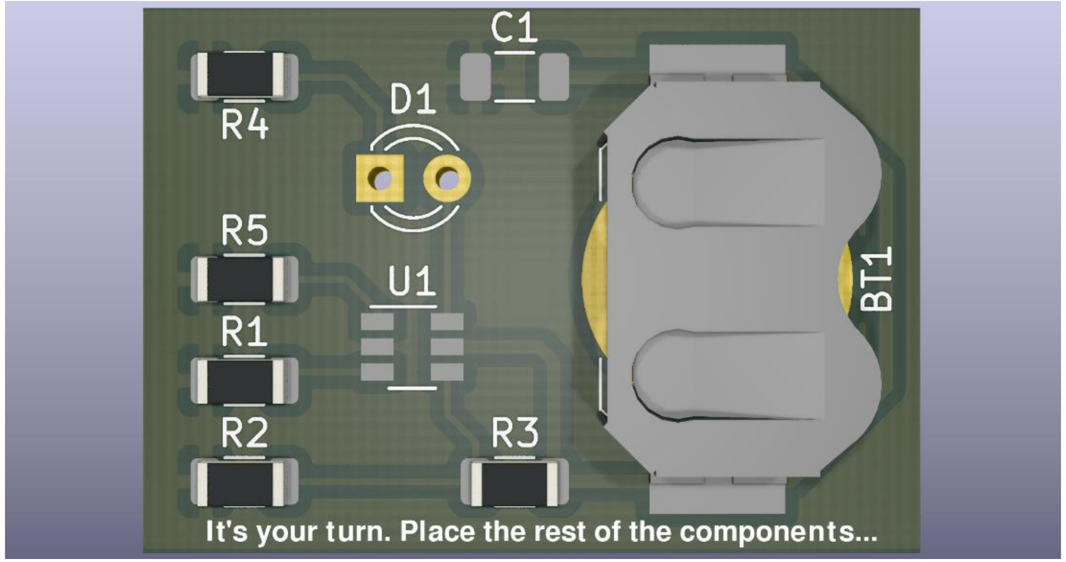


```
*** NEW READING ***
Reading temperature...
Temperature calculated value = 0.000000000 millidegrees Celsius
Temperature correct value = 24337.001666950 millidegrees Celsius
Wrong

Reading acceleration values...
Acceleration calculated values: x=0.000000000 y=0.000000000 z=0.000000000
Acceleration correct values: x=-3.436504475 y=-6.995928380 z=6.176643990
Wrong


*** NEW READING ***
Reading temperature...
Temperature calculated value = 0.000000000 millidegrees Celsius
Temperature correct value = 24668.050838350 millidegrees Celsius
Wrong

Reading acceleration values...
Acceleration calculated values: x=0.000000000 y=0.000000000 z=0.000000000
Acceleration correct values: x=-3.435548350 y=-6.994054375 z=6.179933060
Wrong
```

# Hands On Lab

## Demo 4

An accelerometer-enabled game that lets you place components on a circuit by physically tilting the accelerometer.



## Procedure:

- Close the terminal running picocom.
- Make sure you are in the *~/workshop_baremetal/no-OS/projects/workshop* directory.
- **Build the IIO_EXAMPLE of this project**:
  - make EXAMPLE=iio_example
- **Program the board:**
  - make EXAMPLE=iio_example run
- **Change the directory and run the game:**
  - cd ~/workshop_baremetal/play
  - python3 play.py

# Resources

## No-OS Wiki:

https://wiki.analog.com/resources/no-os/api

https://wiki.analog.com/resources/no-os?s[ ]=no&s[ ]=os

https://www.analog.com/en/analog-dialogue/articles/understanding-and-using-the-no-os-and-platform-drivers.html

https://github.com/analogdevicesinc/no-OS/tree/master/projects

## *Specific hardware resources:*

https://wiki.analog.com/resources/eval/user-guides/eval-adxl355-pmdz/no-os-setup?s[ ]=no&s[ ]=os#adxl355_driver

# Opportunities at ADI

- Internships
- Jobs
- Summer practice
- Get hardware and support from ADI to develop your own projects

ANALOG
DEVICES

AHEAD OF WHAT'S POSSIBLE™

Our departments:

- Hardware Design
- FPGA Digital Design
- Embedded Software
- Applications Software
- Applications Engineering

Send us your CV!

To: *office.romania@analog.com*

Subject: Internship/Practica

Check our available positions with the QR code:

# Thank You!
# Questions?