CEC Continuing Education Center

DesignNews

Getting Started with the Raspberry Pi Pico

DAY 2 : Writing your First Raspberry Pi Pico Application

Sponsored by

Digi-Key ELECTRONICS

BENINGO EMBEDDED GROUP

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.
- Submit questions for the lecturer using the Q&A widget. They will follow-up after the lecture portion concludes.

Course Sessions

- Introduction to the Raspberry Pi Pico
- Writing your First Raspberry Pi Pico Application
- Interfacing with Raspberry Pi Pico Peripherals
- Designing Multicore Raspberry Pi Pico Applications
- Using MicroPython on the Raspberry Pi Pico

1 Installing the Toolchain(s)

Working from a Raspberry Pi 4B or Pi 400

1) A simple script download to do all the work:

```
wget https://raw.githubusercontent.com/raspberrypi/pico-setup/master/pico_setup.sh
chmod +x pico_setup.sh
```

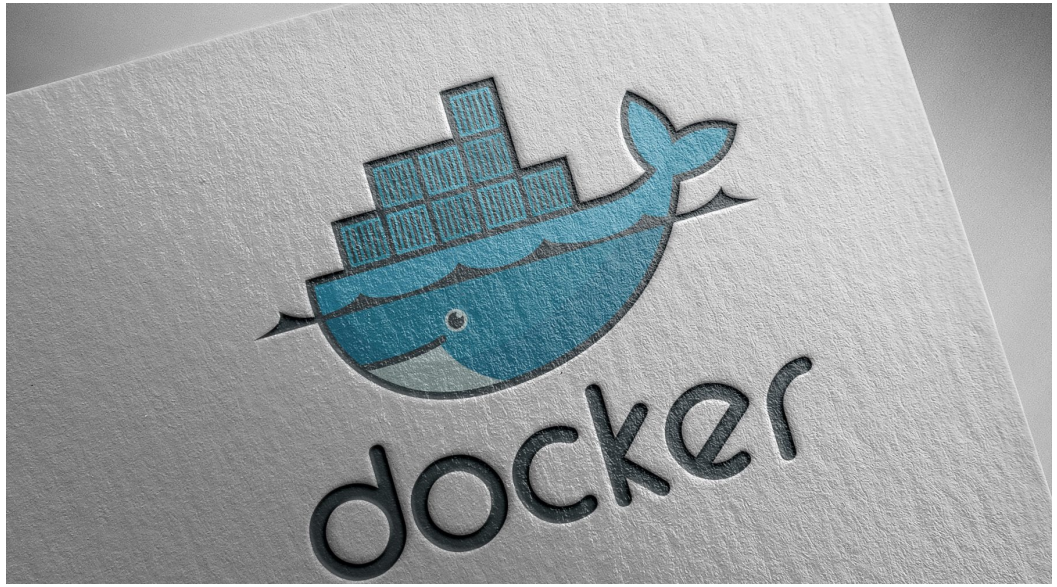
2) Run the script:

```
./pico_setup.sh
```

Working from Linux, Mac, etc

Docker Container with all the setup:

- https://github.com/JacobBeningo/RpiPico
- Run a few simple commands to install everything needed
- Access Pico examples that are linked to the Pico git repo



Setup the Docker Container

Step #1 – Clone the Repo

```
git clone --recurse-submodules https://github.com/JacobBeningo/RpiPico.git
```

Step #2 – Build the Docker Image

```
cd RpiPico
docker build -t rpi/pico .
```

Setup the Docker Container

Step #2 – Build the Docker Image

```
cd RpiPico
docker build -t rpi/pico .
```

```
beningo@Jacobs-MacBook-Pro RpiPico % docker build -t rpi/pico .
[+] Building 0.1s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 37B                                                                     0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                        0.0s
 => [1/5] FROM docker.io/library/ubuntu:latest                                                          0.0s
 => CACHED [2/5] RUN apt-get update &&     apt-get clean &&      apt-get install -y         build-essential         wget      0.0s
 => CACHED [3/5] WORKDIR /home/dev                                                                      0.0s
 => CACHED [4/5] RUN apt-get -y install cmake         gcc-arm-none-eabi         libnewlib-arm-none-eabi         0.0s
 => CACHED [5/5] RUN git clone https://github.com/raspberrypi/pico-sdk /home/sdk/pico-sdk &&      cd /home/sdk/pico-sdk &&    0.0s
 => exporting to image                                                                                  0.0s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:853ceba0f254aa3b9c88feaaa965bff33fb71c76eda2657daa1b5d1de0658f5e            0.0s
 => => naming to docker.io/rpi/pico                                                                     0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
beningo@Jacobs-MacBook-Pro RpiPico %
```

Setup the Docker Container

Step #3 – Run the Docker Image

```
docker-compose up pico
docker run --rm -it --privileged -v "$(PWD):/home/dev" rpi/pico
```

```
beningo@Jacobs-MacBook-Pro RpiPico % docker run --rm -it --privileged -v "$(PWD):/home/dev" rpi/pico
root@8d60fece4dd0:/home/dev# ls
Dockerfile  LICENSE  README.md  pico-examples
root@8d60fece4dd0:/home/dev#
```

The pico-examples directory:

```
root@8d60fece4dd0:/home/dev# cd pico-examples/
root@8d60fece4dd0:/home/dev/pico-examples# ls
CMakeLists.txt   adc     cmake                       flash        ide                   picoboard  rtc     uart
CONTRIBUTING.md  blink   divider                     gpio         interp                pio        spi     usb
LICENSE.TXT      build   dma                         hello_world  multicore             pwm        system  watchdog
README.md        clocks  example_auto_set_url.cmake  i2c          pico_sdk_import.cmake  reset      timer
root@8d60fece4dd0:/home/dev/pico-examples#
```

What method do you prefer for setting up the build chain?
- Manual installation
- Scripts locally (not in container)
- Containerized
- Other

2 Writing a “Blinky” Application

The Blink Application

blink.c

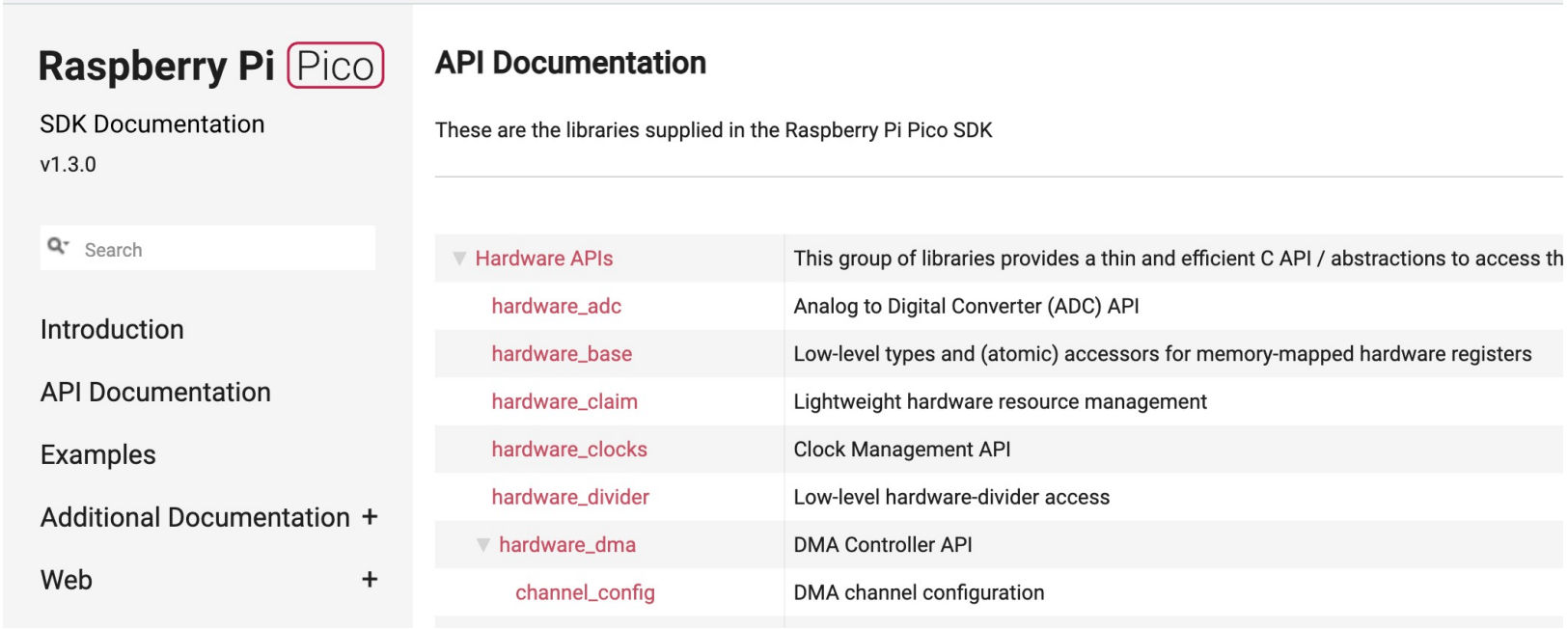
```c
/**
 * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "pico/stdlib.h"

int main() {
#ifndef PICO_DEFAULT_LED_PIN
#warning blink example requires a board with a regular LED
#else
    const uint LED_PIN = PICO_DEFAULT_LED_PIN;
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    while (true) {
        gpio_put(LED_PIN, 1);
        sleep_ms(250);
        gpio_put(LED_PIN, 0);
        sleep_ms(250);
    }
#endif
}
```

blink.c | CMakeLists.txt

```
add_executable(blink
        blink.c
        )

# pull in common dependencies
target_link_libraries(blink pico_stdlib)

# create map/bin/hex file etc.
pico_add_extra_outputs(blink)

# add url via pico_set_program_url
example_auto_set_url(blink)

```

Pico APIs

https://raspberrypi.github.io/pico-sdk-doxygen/modules.html



Pico API's

◆ gpio_init()

```
void gpio_init ( uint  gpio )
```

Initialise a GPIO for (enabled I/O and set func to GPIO_FUNC_SIO)

Clear the output enable (i.e. set to input). Clear any output value.

Parameters

gpio GPIO number

◆ gpio_set_dir()

```
static void gpio_set_dir ( uint  gpio,
                           bool  out
                         )
```

Set a single GPIO direction.

Parameters

gpio GPIO number

out true for out, false for in

◆ gpio_put()

```
static void gpio_put ( uint  gpio,
                       bool  value
                     )
```

Drive a single GPIO high/low.

Parameters

gpio GPIO number

value If false clear the GPIO, otherwise set it.

What makes a good API?
- Easy to remember format
- As many functions as possible
- Limited feature set with expandability
- None of the above
- Other

3 Building and Running Blink

Building and Running Blink

Step #1 – Create a build directory

```
mkdir build
cd build
```

Step #2 – Prepare cmake

```
cmake ..
```

```
root@8d60fece4dd0:/home/dev/pico-examples/build# cmake ..
PICO_SDK_PATH is /home/sdk/pico-sdk
PICO platform is rp2040.
PICO target board is pico.
Using board configuration from /home/sdk/pico-sdk/src/boards/include/boards/pico.h
TinyUSB available at /home/sdk/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; enabling build support for USB.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/dev/pico-examples/build
root@8d60fece4dd0:/home/dev/pico-examples/build#
```

Building and Running Blink

Step #3 – Build the Raspberry Pi Blink Application

```
cd blink
make
```

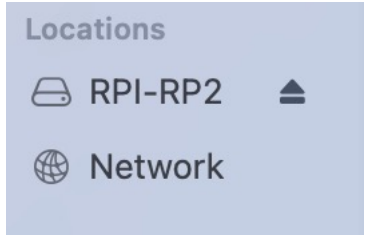
```
root@8d60fece4dd0:/home/dev/pico-examples/build# cd blink/
root@8d60fece4dd0:/home/dev/pico-examples/build/blink# make
Performing build step for 'ELF2UF2Build'
[100%] Built target elf2uf2
No install step for 'ELF2UF2Build'
Completed 'ELF2UF2Build'
Built target ELF2UF2Build
Built target bs2_default
Built target bs2_default_padded_checksummed_asm
Built target blink
root@8d60fece4dd0:/home/dev/pico-examples/build/blink#
```

```
root@8d60fece4dd0:/home/dev/pico-examples/build/blink# ls
CMakeFiles  Makefile  blink.bin  blink.dis  blink.elf  blink.elf.map
 blink.hex  blink.uf2  cmake_install.cmake  elf2uf2
root@8d60fece4dd0:/home/dev/pico-examples/build/blink#
```

Building and Running Blink

Programming the board:

1) Plug in the Pico board while holding the BOOTSEL button
2) It will enumerate as a USB MSD



3) Drag the blink.uf2 file into the RPI-RP2 drive

What types of projects are you interested in using the Raspberry Pi Pico on?

- Production projects
- Proof-of-concept prototypes
- Rapid prototyping
- Maker projects
- Other

4 Going Further

Thank you for attending

Please consider the resources below:

- www.beningo.com
  - Blog, White Papers, Courses
  - Embedded Bytes Newsletter
    - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Getting Started with the Raspberry Pi Pico

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

BENINGO EMBEDDED GROUP

informa markets