Building a minimal blink app for Raspberry Pico 2 (RP2350) with C-SDK 2.1.1 and FreeRTOS 11.1.0

This tutorial is based on the tutorials from Dr Jon EA

https://www.youtube.com/watch?v=OxFwNU18j-c

In short form:

https://www.youtube.com/shorts/lRKw_SS6LBE

and his examples

https://github.com/DrJonEA/RPIPico2FreeRTOSRepoExample

Thanks, Jon, for his great tutorials!

User story

Create a blink program for Raspberry Pico 2 (RP2350) based on the newest versions of Pico C-SDK (2.1.1) and FreeRTOS (11.1.0).

The idea is to give a simple way to build a working application with the newest Pico-SDK/FreeRTOS Versions (end of March 2025), especially the way how to install FreeRTOS11.1.0 under Pico-SDK2.1.1

Git repository:

https://github.com/trimchess/freertos-blink-howto/tree/main

Preconditions

Operating System WIN 11

The Pico C-SDK 2.1.1 is installed in a directory, in my case in

`c:\Users\marti\.pico-sdk\`

The ENV variable `PICO_SDK_ROOT_PATH` points to the Pico-SDK root directory. In my case to

`c:\Users\marti\.pico-sdk\`

The ENV variable `PICO_SDK_PATH` points to the used C-SDK directory. In my case to `c:\Users\marti\.pico-sdk\sdk\2.1.1\`

The ENV variable `PICOTOOL_FETCH_FROM_GIT_PATH` points to picotools depending on the SDK 2.1.1, in my case to `c:\Users\marti\.pico-sdk\picotool\2.1.1\`

The ENV variable `PICO_ARM_TOOLCHAIN_PATH` points to the ARM toolchain root directory. In my case to `c:\Users\marti\.pico-sdk\toolchain\14_2_Rel1\`

Steps to build the project

Step 1

Install the FreeRTOS for Raspberry Pico 2 library step by step in your preferred working directory and finally copy it to your preferred library directory. My working directory for this is
`d:\temp`
the lib directory will be
`e:\projects\pico\c_cpp\external-libs\freertos\FreeRTOS-KernelV11.1.0\`

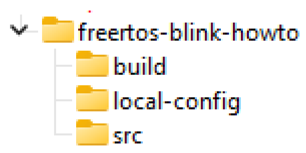
Steps to build the library:

1. go to `d:\temp`, open a cmd shell and execute
   `git clone https://github.com/FreeRTOS/FreeRTOS`
2. change to `d:\temp\FreeRTOS\`. Create a subdirectory `d:\temp\FreeRTOS\lib`
3. change to this directory, open a cmd shell and enter the command
   `git submodule add https://github.com/FreeRTOS/FreeRTOS-Kernel`
4. change to subdirectory
   `cd d:\temp\FreeRTOS\lib\FreeRTOS-Kernel\portable\ThirdParty\Community-Supported-Ports\`
5. open a cmd shell, enter the command
   `git submodule update --init`
6. In your preferred library directory create a directory `FreeRTOS-KernelV11.1.0`
   my preferred lib directory for FreeRTOS libs is
   `e:\projects\pico\c_cpp\external-libs\freertos\`
   and the full path for the new lib directory is
   `e:\projects\pico\c_cpp\external-libs\freertos\FreeRTOS-KernelV11.1.0\`
7. copy all from `d:\temp\FreeRTOS\FreeRTOS\lib\FreeRTOS-Kernel\`
   to the directory created in Step 1 / 6. My directory is
8. Set the ENV variable `FREERTOS_KERNEL_PATH` to the new lib directory, in my case

Step 2

Create the directory structure of the project. The root directory is

- `freertos-blink-howto`



`build`	the projects build directory for CMake builds
`local-config`	the directory for local configurations. In this project only for the local `FreeRTOSConfig.h` file
`src`	the projects source file directory

<u>Step 3</u>

Copy the Pico 2 (RP2350) file

`%FREERTOS_KERNEL_PATH%\portable\ThirdParty\Community-Supported-Ports\GCC\RP2350_ARM_NTZ\FreeRTOS_Kernel_import.cmake`

to your projects root directory `(freertos-blink-howto)`

<u>Step 4</u>

Under the directory `local-config` create the directory `FreeRTOS-Kernel` `(freertos-blink-howto\local-config\FreeRTOS-Kernel)`

Copy file `FreeRTOSConfig.h` from my example project to this directory

<u>Step 5</u>

Create the `CMakeLists.txt` files in the root directory and in the src directory

`\freertos-blink-howto\CMakeLists.txt`

```
#-----------------------------------------------
# SDK definitions
set(PICO_SDK_PATH $ENV{PICO_SDK_PATH})
set(PICO_SDK_ROOT $ENV{PICO_SDK_ROOT_PATH})

# Toolchain definitions
set(PICO_TOOLCHAIN_PATH $ENV{PICO_ARM_TOOLCHAIN_PATH})

# Pull in SDK from the SDK
include(${PICO_SDK_PATH}/external/pico_sdk_import.cmake)

# project definitions
project(${PROJECT_NAME} C CXX ASM)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)
set(PICO_CXX_ENABLE_EXCEPTIONS 1)

# initialize SDK
pico_sdk_init()

# FreeRTOS definitions and pull in FreeRTOS
SET(FREERTOS_CONFIG_FILE_DIRECTORY "${CMAKE_CURRENT_LIST_DIR}/local-config/FreeRTOS-Kernel"
CACHE STRING "Local Config")
include(FreeRTOS_Kernel_import.cmake)

# Output some variables
include(CMakePrintHelpers)
cmake_print_variables(CMAKE_C_STANDARD)
cmake_print_variables(CMAKE_CXX_STANDARD)
cmake_print_variables(PICO_BOARD)
cmake_print_variables(PICO_PLATFORM)
cmake_print_variables(PICO_SDK_PATH)
cmake_print_variables(PICO_SDK_VERSION_STRING)
cmake_print_variables(PICO_TOOLCHAIN_PATH)
cmake_print_variables(PICO_COMPILER)
cmake_print_variables(CMAKE_C_COMPILER_ID)

# include src directory
add_subdirectory(src)
#-----------------------------------------------
```

\freertos-blink-howto\src\CMakeLists.txt

```
#------------------------------------------------
add_executable(${OUTPUT_NAME}
        main.cpp
        )

target_link_libraries(${OUTPUT_NAME} PUBLIC
        pico_stdlib
        FreeRTOS-Kernel-Heap4
        )

pico_enable_stdio_usb(${OUTPUT_NAME} 1)
pico_enable_stdio_uart(${OUTPUT_NAME} 1)

pico_add_extra_outputs(${OUTPUT_NAME})

#------------------------------------------------
```

Step 6

Create file `src\main.cpp`

```
#include <FreeRTOS.h>
#include <task.h>
#include <stdio.h>
#include "pico/stdlib.h"


void led_task()
{
    const TickType_t xDelay = 500 / portTICK_PERIOD_MS;
    const uint LED_PIN = 25; //Pico 2 internal LED
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    while (true)
    {
        gpio_put(LED_PIN, 1);
        vTaskDelay(xDelay);
        gpio_put(LED_PIN, 0);
        vTaskDelay(2*xDelay);
    }
}


int main()
{
    stdio_init_all();
    xTaskCreate((TaskFunction_t)led_task, "LED_Task", 256, NULL, 1, NULL);
    vTaskStartScheduler();
}
```

Step 7

The final project structure is

```
build\
local-config\
       FreeRTOS-Kernel\
              FreeRTOSConfig.h
src\
       CMakeLists.txt
       main.cpp
CMakeLists.txt
FreeRTOS_Kernel_import.cmake
freertos-blink-howto.docx
```

Step 8

Compile and link your project. I use MinGW Make

From the projects root directory:

```
cd build
cmake -G "MinGW Makefiles" ..
cmake --build . --target all -j
```

Step 9

Copy the file

```
e:\projects\pico\c_cpp\projects\pico2\freertos_11.1.0\freertos-blink-
howto\build\src\APP.uf2
```

to your Pico 2 (bootsel/reset method)

Step 10

Start your own project based on the how to project! Have fun!