CEC Continuing Education Center

DesignNews

Constructing a Raspberry Pi RP2040 Low-Power Sensor Node

Day 2:
RP2040 Internals

Sponsored by

Digi-Key ELECTRONICS

informa markets

Webinar Logistics

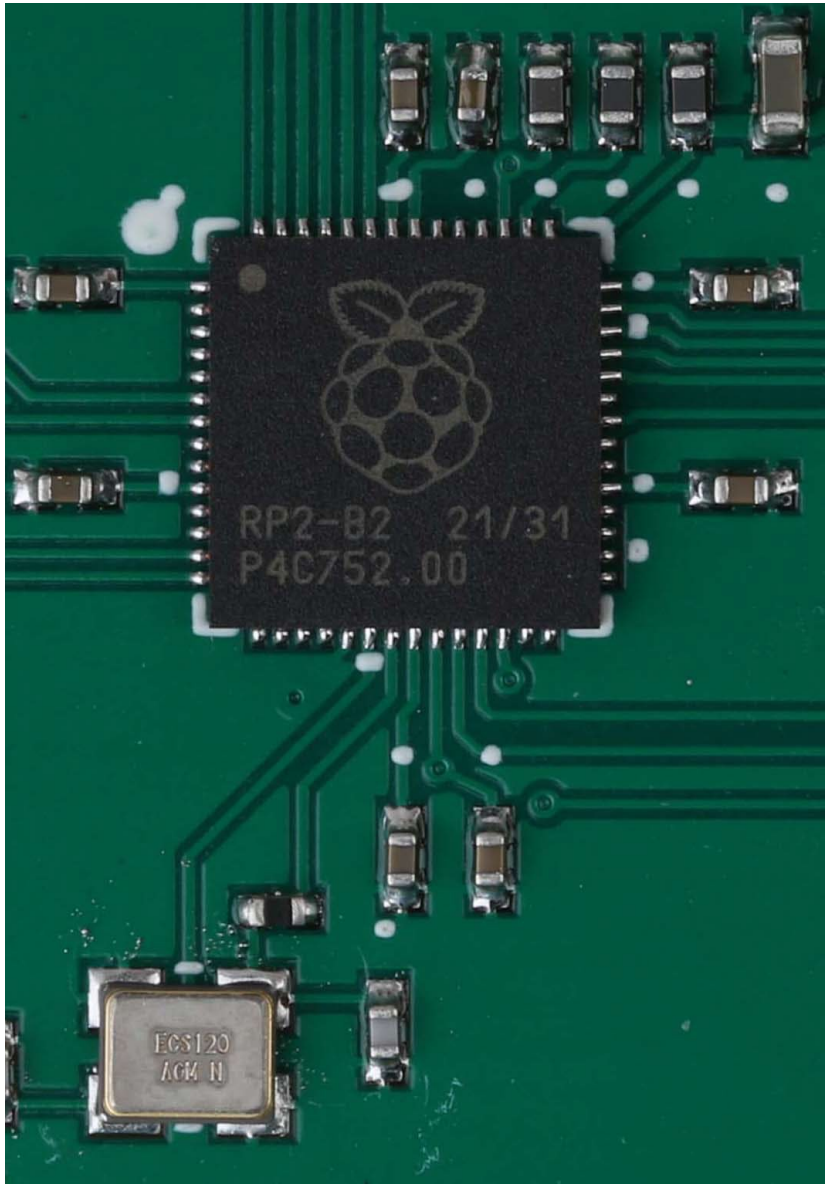
- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Fred Eady

Visit ‘Lecturer Profile’ in your console for more details.

AGENDA

- RP2040 Documentation
  - The API
  - Hardware API
    - ADC
    - GPIO
    - I2C
    - SPI



In the Beginning....

Raspberry Pi Pico SDK: Raspberry Pi

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

Raspberry Pi Pico

SDK Documentation

v1.4.0

Search

- Introduction
- API Documentation
- Examples
- Additional Documentation +
- Web +

Raspberry Pi Pico SDK

The Raspberry Pi Pico SDK (Software Development Kit), hencefort
designed to provide an API (Application Programming Interface) a

A single program runs on the device at a time with a conventional
PIO (Programmable IO).

Additionally the SDK provides higher-level libraries for dealing witl
applications, or full-fledged runtime environments such as MicroP

This documentation is generated from the SDK source tree using

SDK Design

The RP2040 is a powerful chip, however it is an embedded enviro
configuration) are necessarily much more visible to the developer

The intention within the SDK has been for features to just work ou
libraries used.

The Build System

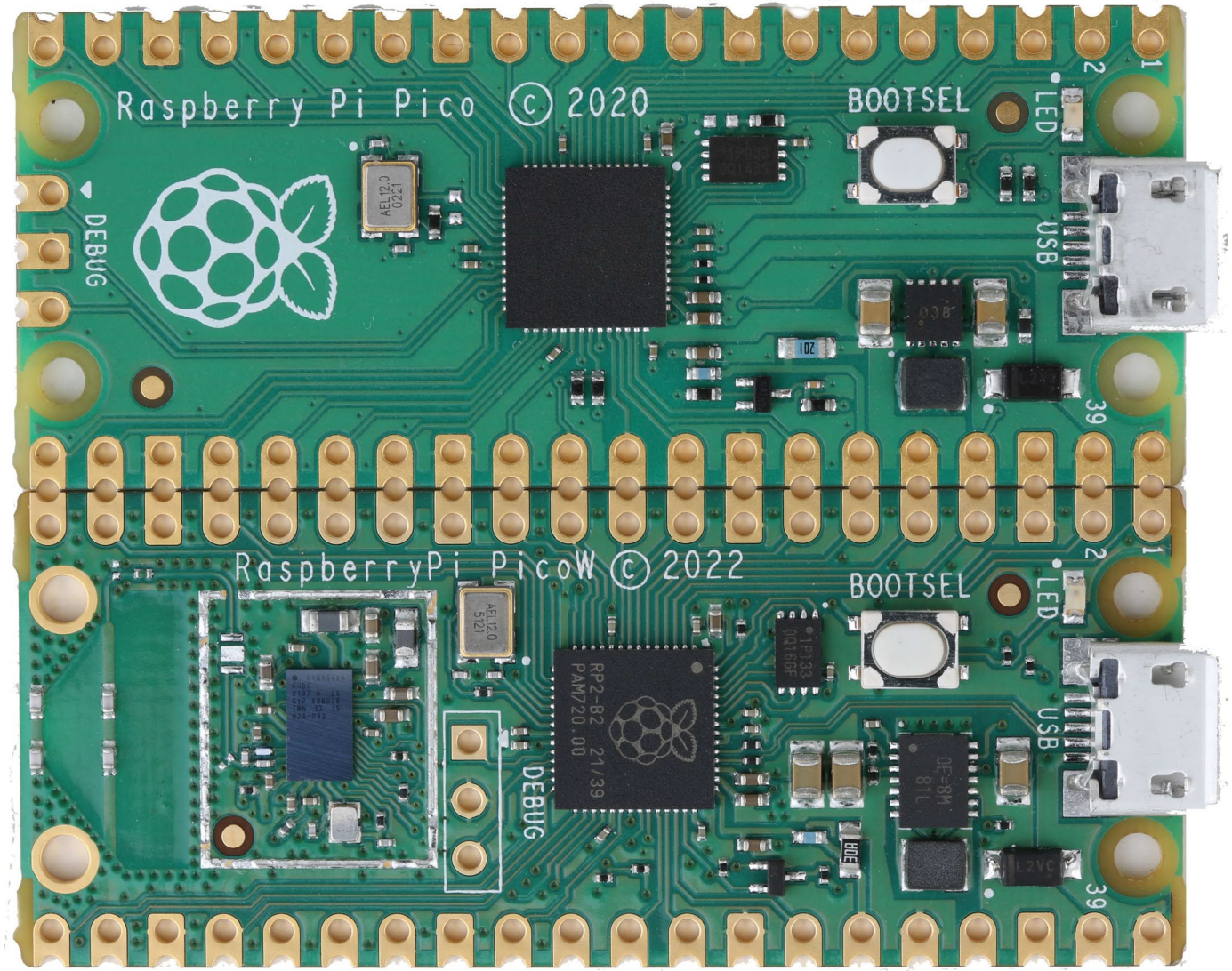
The SDK uses CMake to manage the build. CMake is widely suppo
`make`, `ninja` or other build tools) customized for the platform and

Apart from being a widely-used build system for C/C++ developm

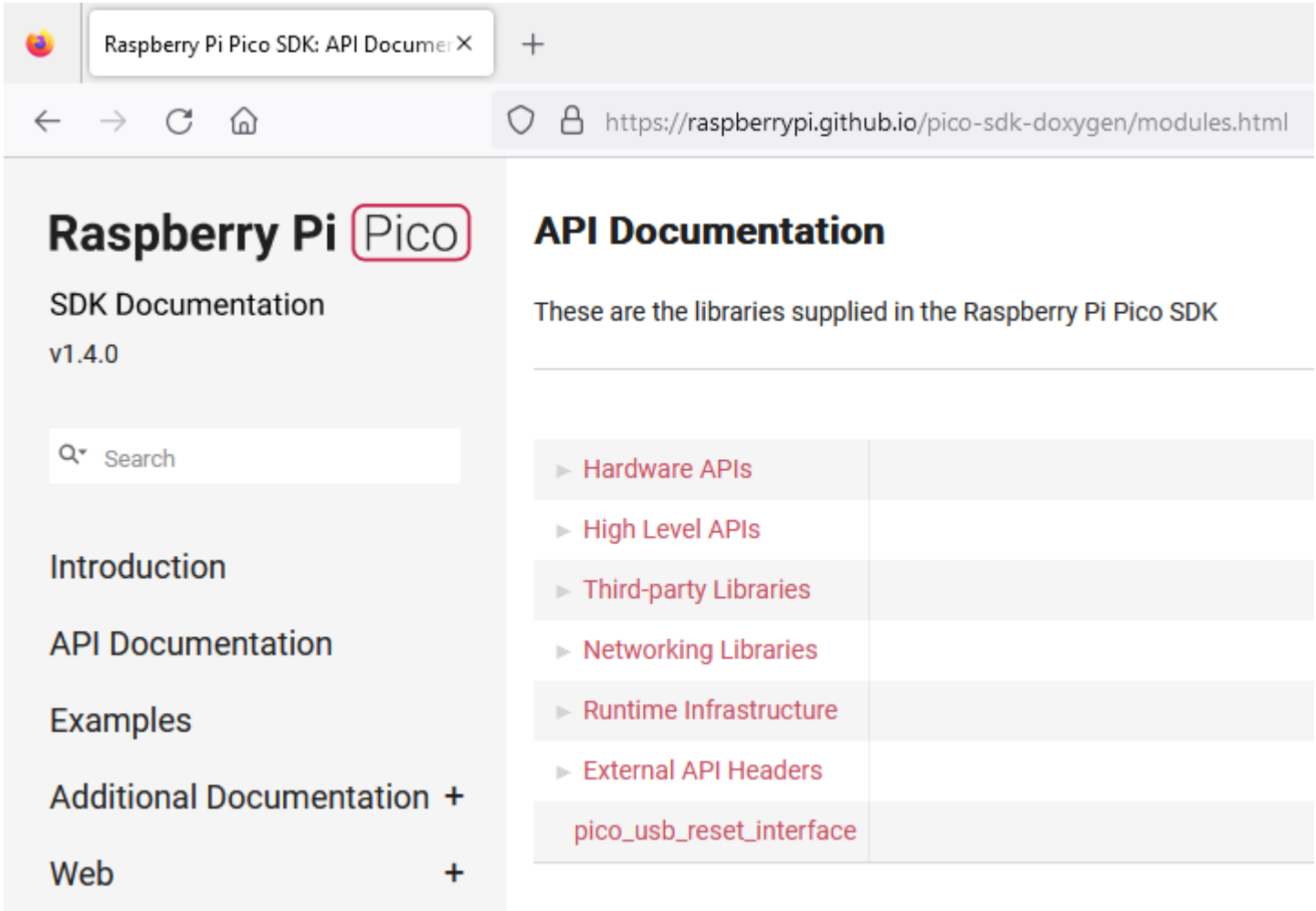
The SDK builds an executable which is bare-metal, i.e. it includes

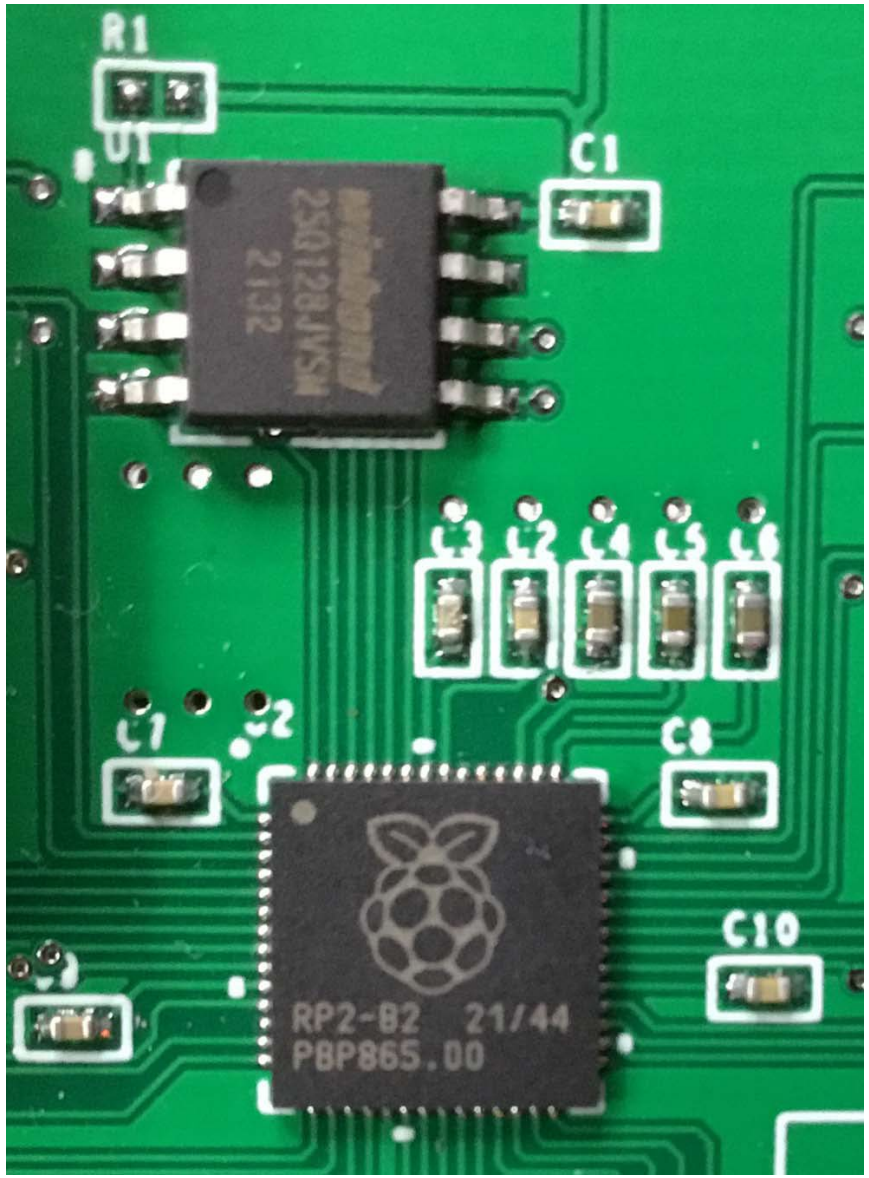
Examples

This SDK documentation contains a number of example code frag
licensed under the 3-Clause BSD license.



The API


Raspberry Pi Pico SDK: API Docume
https://raspberrypi.github.io/pico-sdk-doxygen/modules.html
Raspberry Pi Pico
SDK Documentation
v1.4.0
Search
Introduction
API Documentation
Examples
Additional Documentation +
Web +
API Documentation
These are the libraries supplied in the Raspberry Pi Pico SDK
Hardware APIs
High Level APIs
Third-party Libraries
Networking Libraries
Runtime Infrastructure
External API Headers
pico_usb_reset_interface


R1
U1
C1
C2
C8
C10
RP2-B2 21/44
P8P865.00

Hardware API – ADC

Raspberry Pi Pico SDK: hardware_ad

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

Raspberry Pi Pico

SDK Documentation

v1.4.0

Search

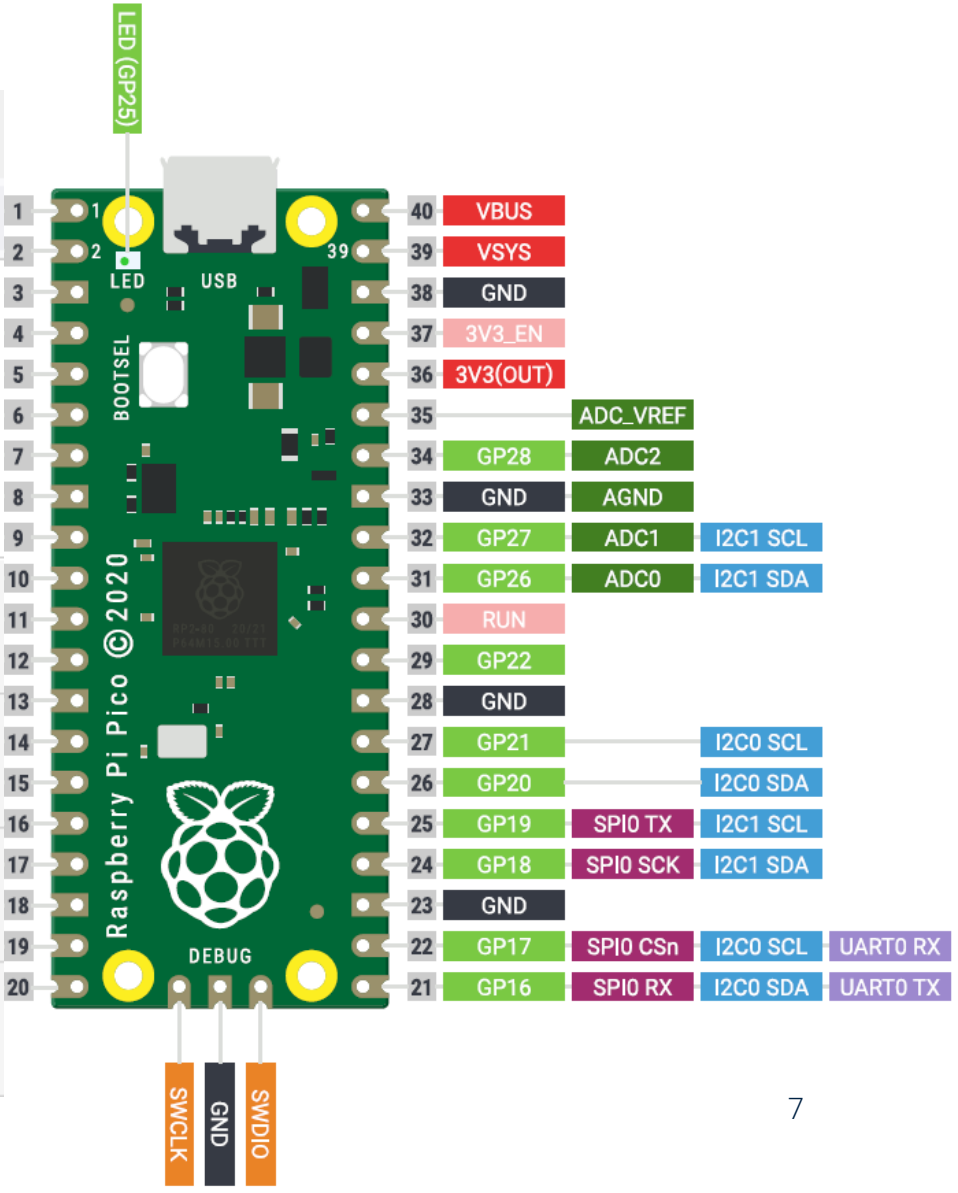
- Introduction
- API Documentation
- Examples
- Additional Documentation +
- Web +

hardware_adc

Hardware APIs

Functions

void	adc_init (void) Initialise the ADC HW.
static void	adc_gpio_init (uint gpio) Initialise the gpio for use as an ADC pin. More...
static void	adc_select_input (uint input) ADC input select. More...
static uint	adc_get_selected_input (void) Get the currently selected ADC input channel. More...

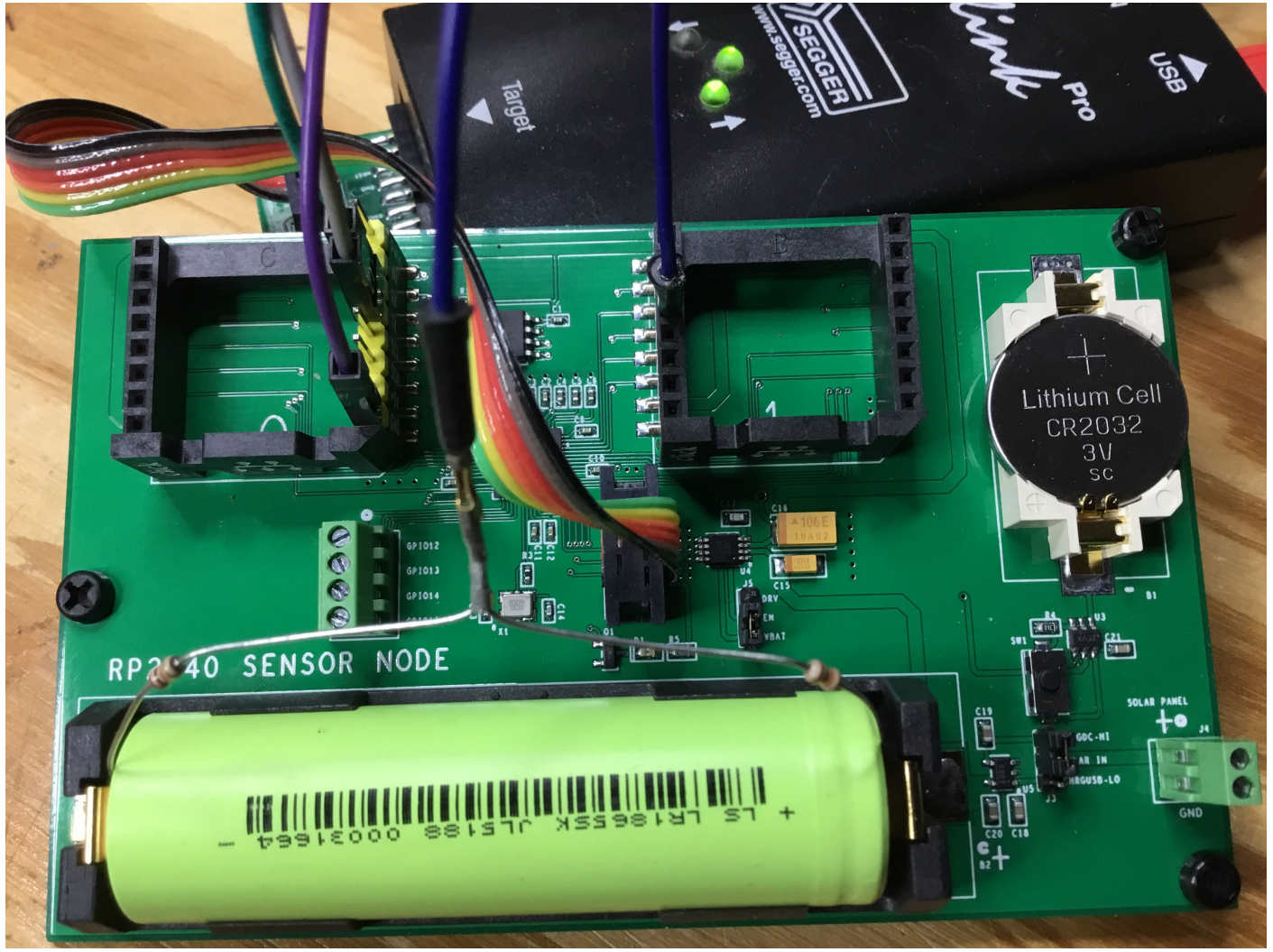


Hardware API – ADC

main.c | CMakeLists.txt

CMakeLists.txt

```
# Set minimum required version of CMake
cmake_minimum_required(VERSION 3.12)

# Include build functions from Pico SDK
include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)

# Set name of project (as PROJECT_NAME) and C/C++ standards
project(adc C CXX ASM)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

# Creates a pico-sdk subdirectory in our project for the libraries
pico_sdk_init()

# Tell CMake where to find the executable source file
add_executable(${PROJECT_NAME}
    main.c
)

# Create map/bin/hex/uf2 files
pico_add_extra_outputs(${PROJECT_NAME})

# Link to pico_stdlib (gpio, time, etc. functions)
target_link_libraries(${PROJECT_NAME}
    pico_stdlib
    hardware_gpio
    hardware_adc
)

# Enable usb output, enable uart output
pico_enable_stdio_usb(${PROJECT_NAME} 0)
pico_enable_stdio_uart(${PROJECT_NAME} 1)
```

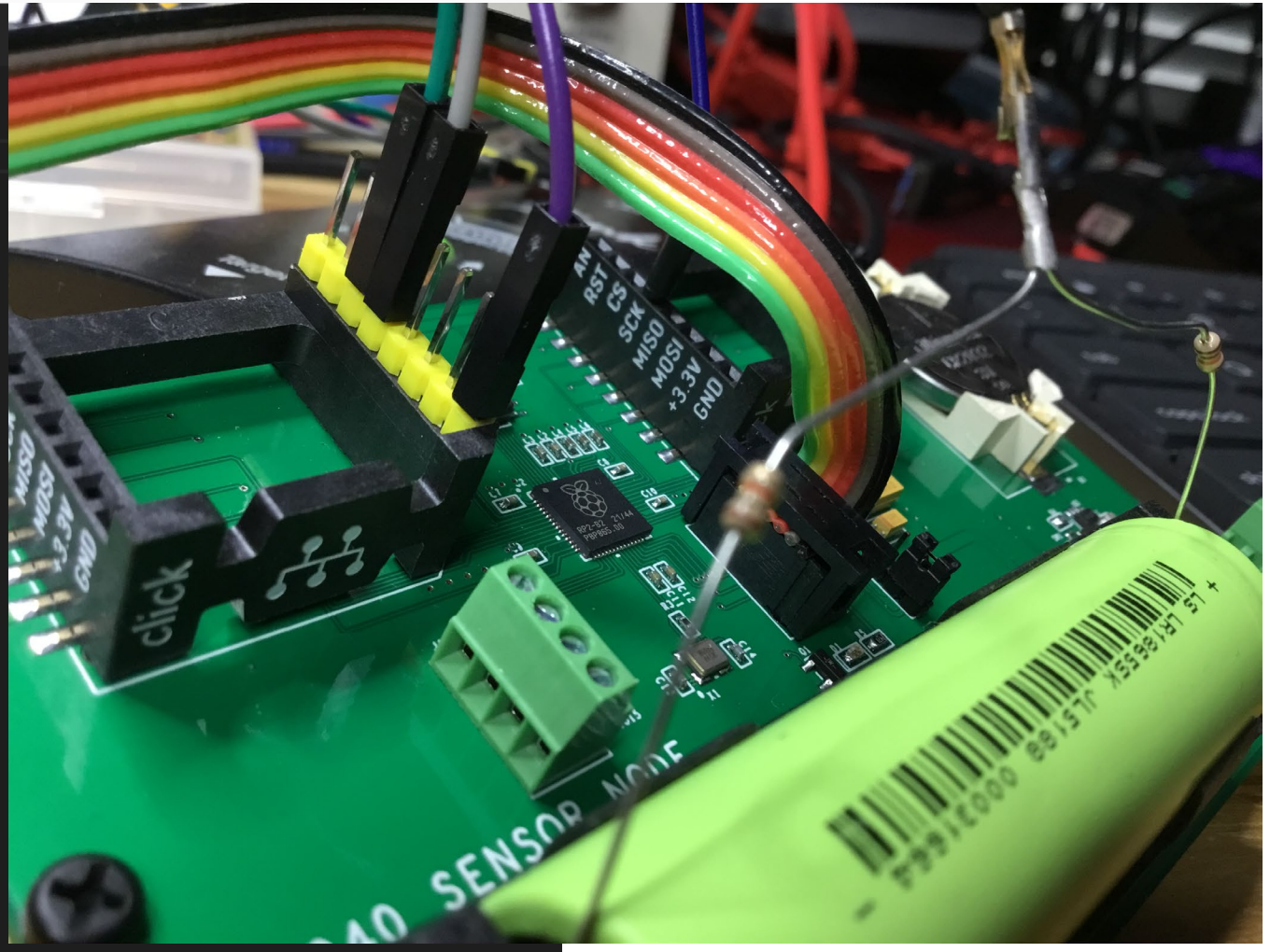


Hardware API – ADC

main.c | CMakeLists.txt

main.c

```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/gpio.h"
#include "hardware/adc.h"

const uint tx0_pin = 0;
const uint rx0_pin = 1;
// 12-bit conversion, ADC_VREF = 3.3 V
const float step12bit = 3.3f / (1 << 12);

uint16_t adc_val;

int main()
{
    // Initialize UART0
    stdio_uart_init_full(uart0, 115200, tx0_pin, rx0_pin);
    // Initialize the ADC
    adc_init();
    adc_gpio_init(26);
    // Select ADC input 0 (GPIO26)
    adc_select_input(0);

    while (1)
    {
        adc_val = adc_read();
        printf("Raw ADC Value = 0x%03x ** 18650 Voltage = %f V\n", adc_val, (adc_val * step12bit)*2);
        sleep_ms(500);
    }
}
```



Hardware API – ADC

Serial Input/Output Monitor

File Edit View Configuration

Toggle Break | Toggle RTS | Toggle DTR | Port Options | Invert Bytes | Local Echo | Macro Manager | Logging Options | RS485

Port | Misc

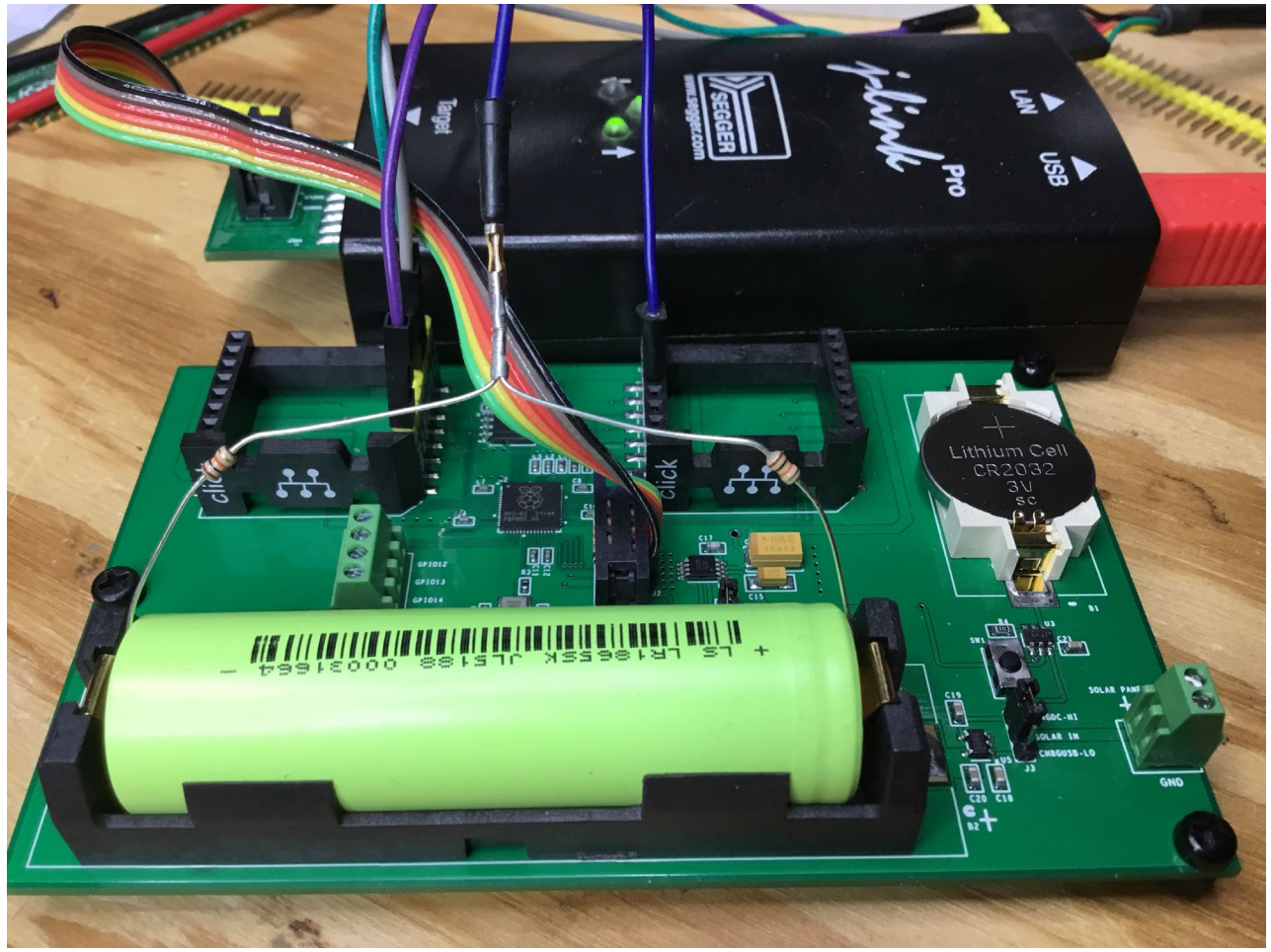
```
Raw ADC Value = 0x9e1 ** 18650 Voltage = 4.075049 V
Raw ADC Value = 0x9b0 ** 18650 Voltage = 3.996094 V
Raw ADC Value = 0x9be ** 18650 Voltage = 4.018652 V
Raw ADC Value = 0x9a1 ** 18650 Voltage = 3.971924 V
Raw ADC Value = 0x9aa ** 18650 Voltage = 3.986426 V
Raw ADC Value = 0x9a7 ** 18650 Voltage = 3.981592 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9ad ** 18650 Voltage = 3.991260 V
Raw ADC Value = 0x9ab ** 18650 Voltage = 3.988037 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9bd ** 18650 Voltage = 4.017041 V
Raw ADC Value = 0x9b8 ** 18650 Voltage = 4.008984 V
Raw ADC Value = 0x9a2 ** 18650 Voltage = 3.973535 V
Raw ADC Value = 0x9aa ** 18650 Voltage = 3.986426 V
Raw ADC Value = 0x9ab ** 18650 Voltage = 3.988037 V
Raw ADC Value = 0x9a1 ** 18650 Voltage = 3.971924 V
Raw ADC Value = 0x9a2 ** 18650 Voltage = 3.973535 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9a3 ** 18650 Voltage = 3.975147 V
Raw ADC Value = 0x9be ** 18650 Voltage = 4.018652 V
Raw ADC Value = 0x9a0 ** 18650 Voltage = 3.970312 V
Raw ADC Value = 0x9a0 ** 18650 Voltage = 3.970312 V
Raw ADC Value = 0x9a2 ** 18650 Voltage = 3.973535 V
Raw ADC Value = 0x9ac ** 18650 Voltage = 3.989648 V
Raw ADC Value = 0x9a5 ** 18650 Voltage = 3.978369 V
Raw ADC Value = 0x9b6 ** 18650 Voltage = 4.005762 V
Raw ADC Value = 0x9bd ** 18650 Voltage = 4.017041 V
Raw ADC Value = 0x9a5 ** 18650 Voltage = 3.978369 V
Raw ADC Value = 0x99c ** 18650 Voltage = 3.963867 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9b0 ** 18650 Voltage = 3.996094 V
Raw ADC Value = 0x9ab ** 18650 Voltage = 3.988037 V
```

Macro List

52 61 77 20 41 44 43
33 2E 39 39 36 30 39

ASCII
HEX
DSR DTR
COM6 8N1 115200

Raw ADC Value = 0x9b6 ** 18650 Voltage = 4.005762 V
Raw ADC Value = 0x9bd ** 18650 Voltage = 4.017041 V
Raw ADC Value = 0x9a5 ** 18650 Voltage = 3.978369 V
Raw ADC Value = 0x99c ** 18650 Voltage = 3.963867 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9bc ** 18650 Voltage = 4.015429 V
Raw ADC Value = 0x9b0 ** 18650 Voltage = 3.996094 V
Raw ADC Value = 0x9ab ** 18650 Voltage = 3.988037 V



Hardware API – GPIO

Raspberry Pi Pico SDK: hardware_gp

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

Raspberry Pi Pico

SDK Documentation

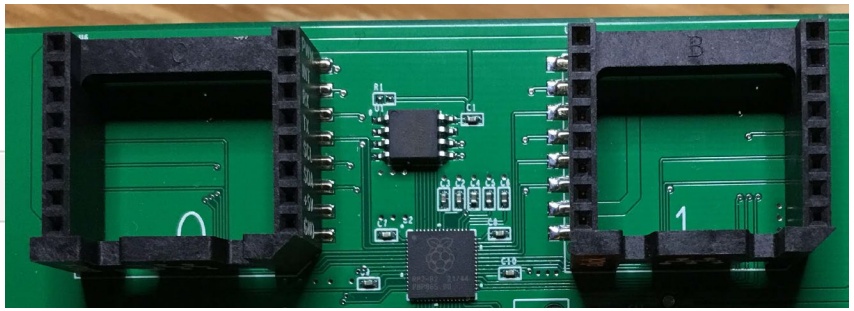
v1.4.0

Search

- Introduction
- API Documentation
- Examples
- Additional Documentation +
- Web +

hardware_gpio

Hardware APIs



Typedefs

typedef void(* gpio_irq_callback_t) (uint gpio, uint32_t event_mask)

Enumerations

enum gpio_function {
GPIO_FUNC_XIP = 0 , **GPIO_FUNC_SPI** = 1 , **GPIO_FUNC_UART** = 2 , **GPIO_FUNC_I2C** = 3 ,
GPIO_FUNC_PWM = 4 , **GPIO_FUNC_SIO** = 5 , **GPIO_FUNC_PIO0** = 6 , **GPIO_FUNC_PIO1** = 7 ,
GPIO_FUNC_GPCK = 8 , **GPIO_FUNC_USB** = 9 , **GPIO_FUNC_NULL** = 0x1f
}

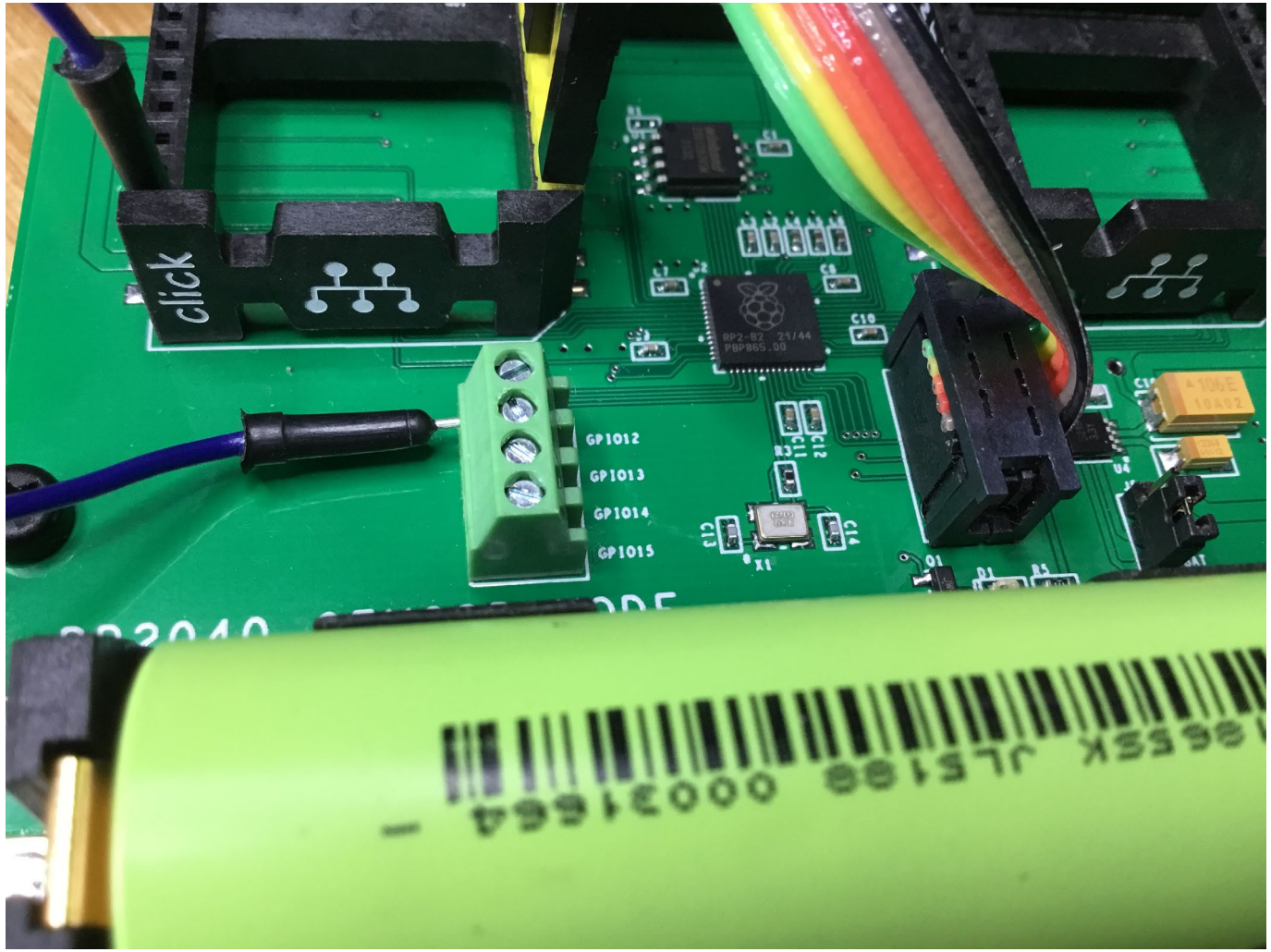
GPIO function definitions for use with function select. More...

Hardware API – GPIO

main.c | CMakeLists.txt

CMakeLists.txt

```
# Set minimum required version of CMake
cmake_minimum_required(VERSION 3.12)

# Include build functions from Pico SDK
include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)

# Set name of project (as PROJECT_NAME) and C/C++ standards
project(blinkirq C CXX ASM)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

# Creates a pico-sdk subdirectory in our project for the libraries
pico_sdk_init()

# Tell CMake where to find the executable source file
add_executable(${PROJECT_NAME}
    main.c
)

# Create map/bin/hex/uf2 files
pico_add_extra_outputs(${PROJECT_NAME})

# Link to pico_stdlib (gpio, time, etc. functions)
target_link_libraries(${PROJECT_NAME}
    pico_stdlib
    hardware_gpio
)

# Enable usb output, enable uart output
pico_enable_stdio_usb(${PROJECT_NAME} 0)
pico_enable_stdio_uart(${PROJECT_NAME} 1)
```

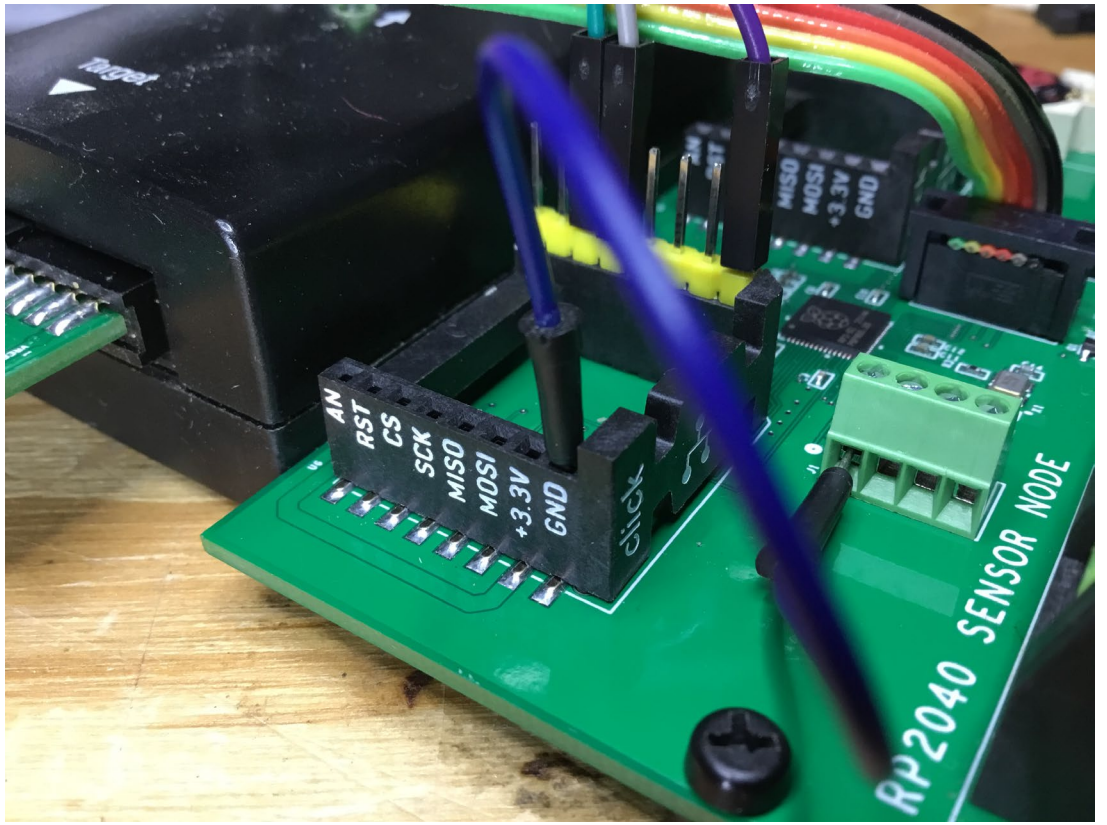


Hardware API – GPIO

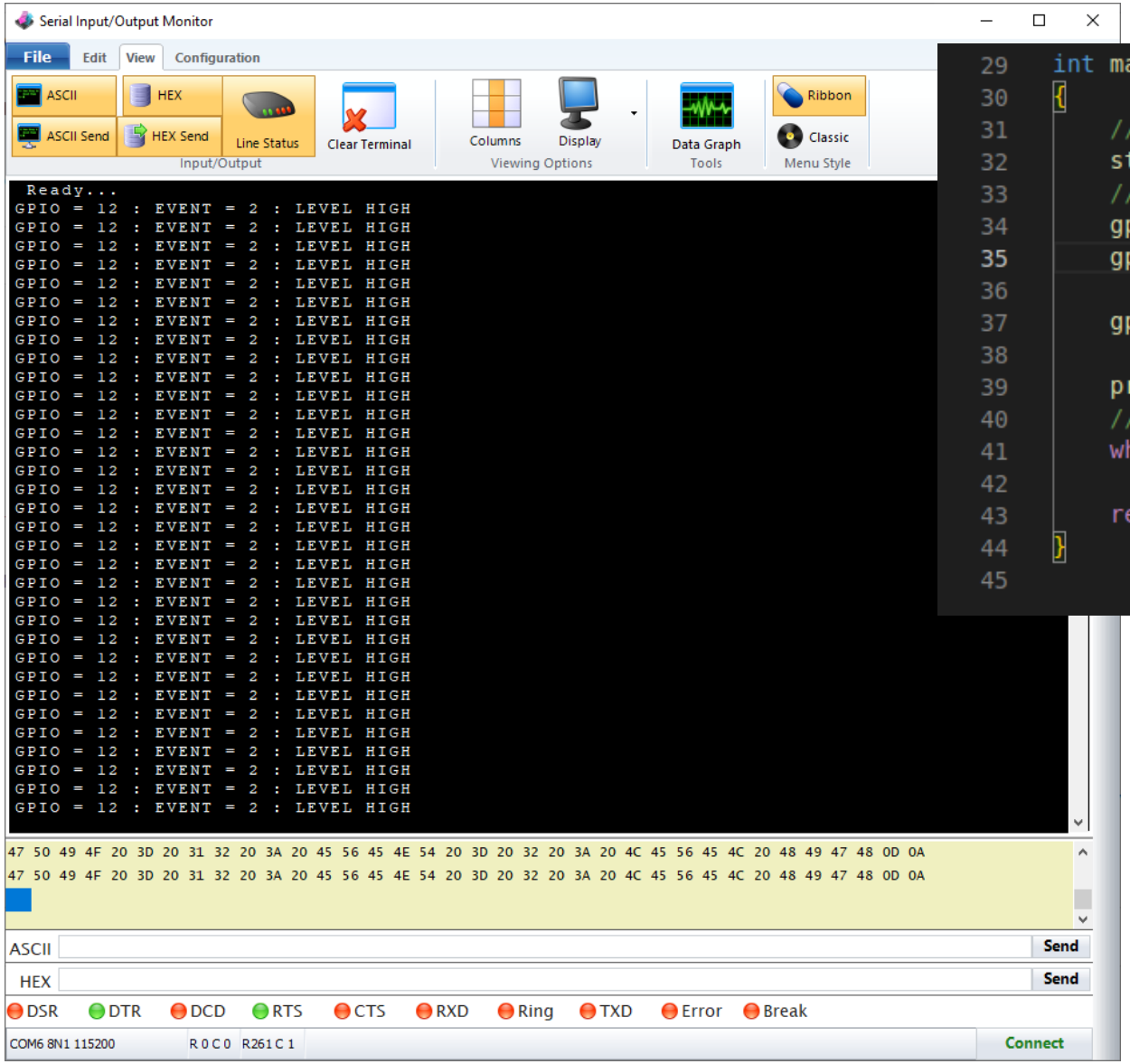
main.c CMakeLists.txt

main.c

```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/gpio.h"

const uint led_pin = 17;
const uint tx0_pin = 0;
const uint rx0_pin = 1;

void gpio_callback(uint gpio, uint32_t events)

{
    switch(events)
    {
        case 1:
            printf("GPIO = %d : EVENT = %d : LEVEL LOW\r\n",gpio,events);
        break;
        case 2:
            printf("GPIO = %d : EVENT = %d : LEVEL HIGH\r\n",gpio,events);
        break;
        case 3:
            printf("GPIO = %d : EVENT = %d : FALLING EDGE\r\n",gpio,events);
        break;
        case 4:
            printf("GPIO = %d : EVENT = %d : RISING EDGE\r\n",gpio,events);
        break;
    }
}
```



Hardware API – GPIO




```c
int main()
{
    // Initialize chosen serial port
    stdio_uart_init_full(uart0, 115200, tx0_pin, rx0_pin);
    // Initialize LED pin
    gpio_init(led_pin);
    gpio_set_dir(led_pin, GPIO_OUT);

    gpio_set_irq_enabled_with_callback(12, GPIO_IRQ_LEVEL_HIGH, true, &gpio_callback);

    printf("Ready...\r\n");
    // Wait forever
    while (1);

    return 0;
}
```

```
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
GPIO = 12 : EVENT = 2 : LEVEL HIGH
```

Hardware API – I²C

Raspberry Pi Pico SDK: hardware_i2c

https://raspberrypi.github.io/pico-sdk-doxygen/group__hardware__i2c.html

Raspberry Pi Pico

SDK Documentation

v1.4.0

Search

- Introduction
- API Documentation
- Examples
- Additional Documentation +
- Web +

hardware_i2c

Hardware APIs

Functions

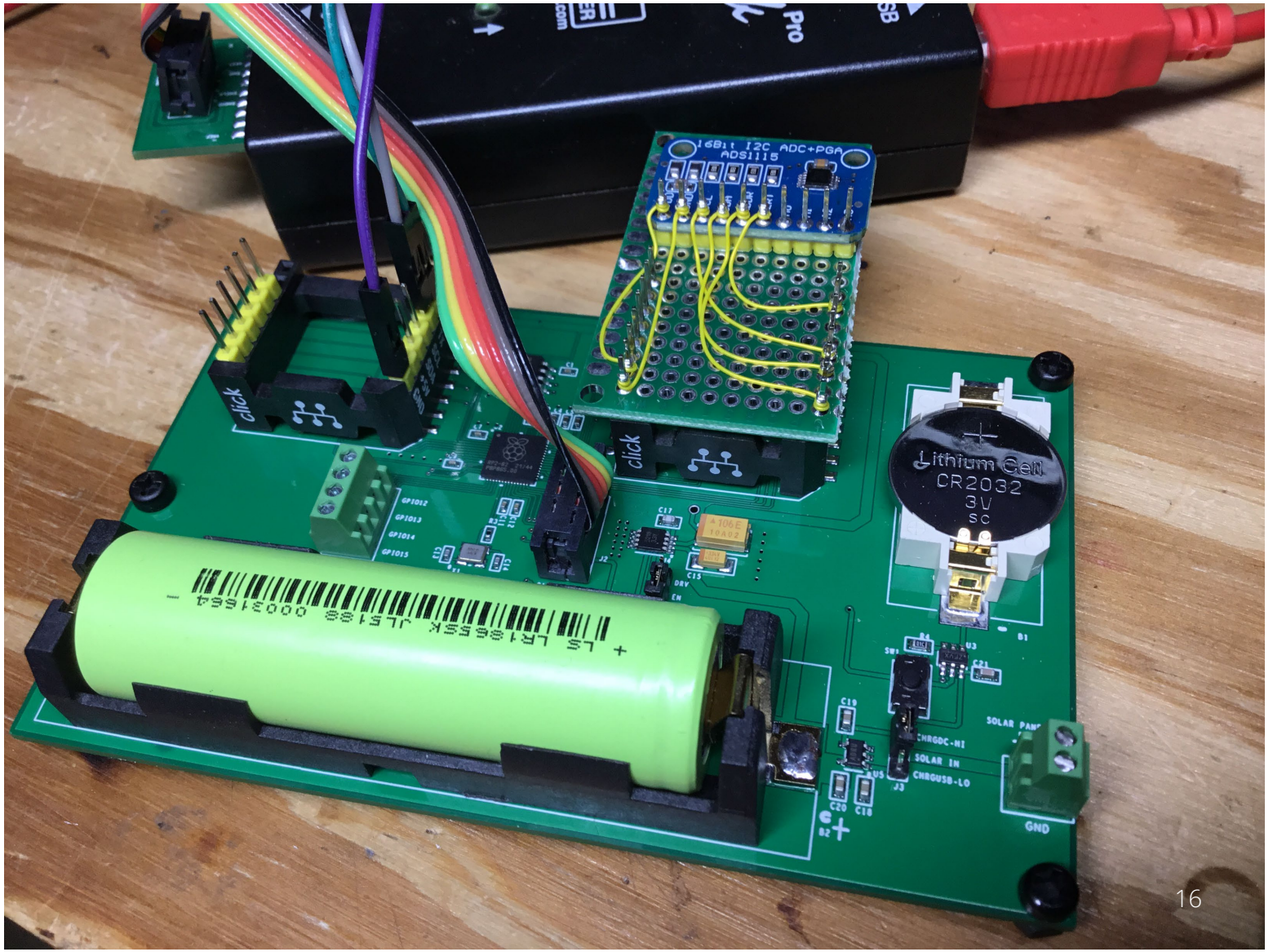
uint	i2c_init (i2c_inst_t *i2c, uint baudrate) Initialise the I2C HW block. More...
void	i2c_deinit (i2c_inst_t *i2c) Disable the I2C HW block. More...
uint	i2c_set_baudrate (i2c_inst_t *i2c, uint baudrate) Set I2C baudrate. More...
void	i2c_set_slave_mode (i2c_inst_t *i2c, bool slave, uint8_t addr) Set I2C port to slave mode. More...

Hardware API – I²C

M CMakeLists.txt | C main.c ×

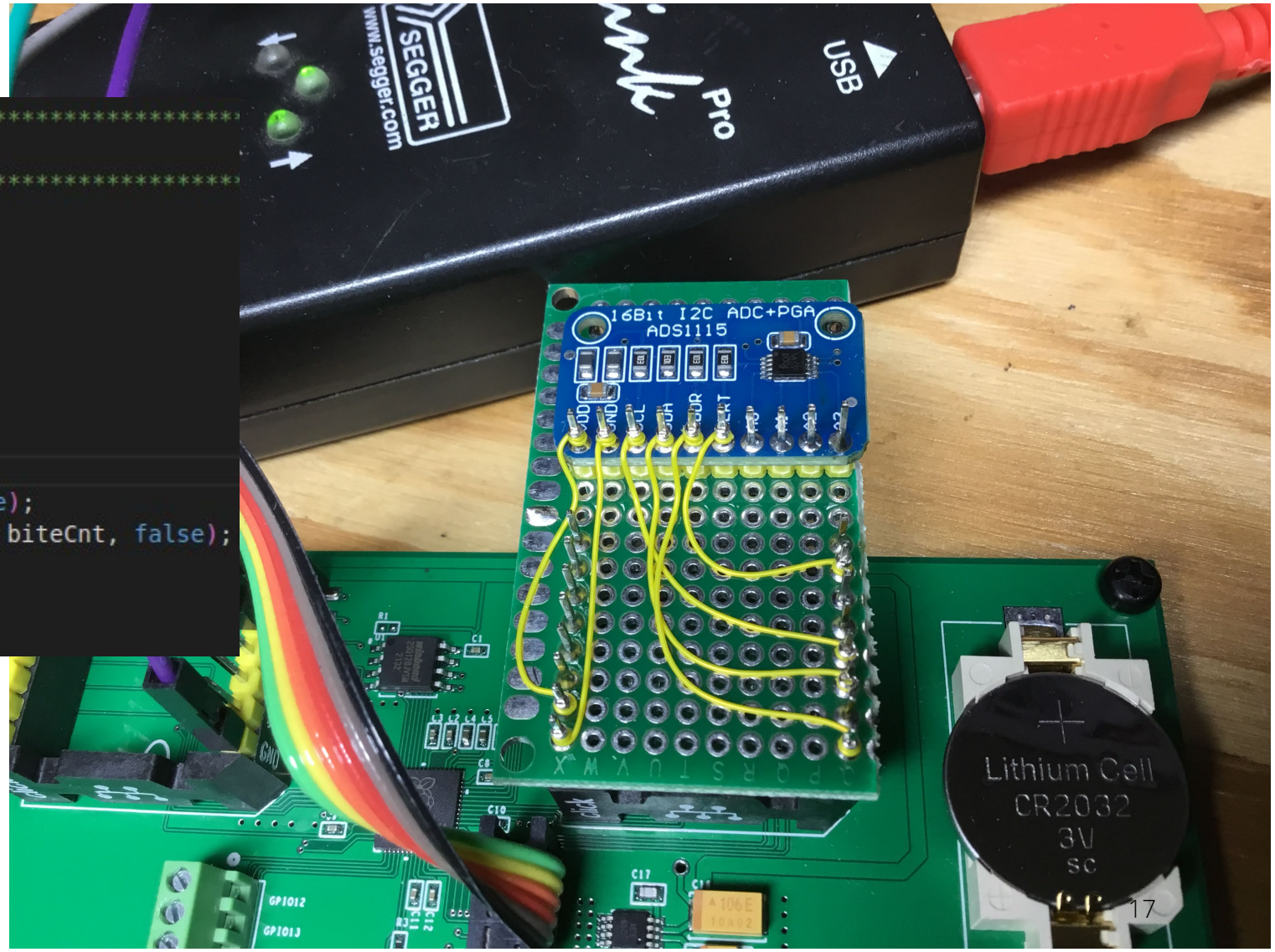
C main.c

```c
i2c_inst_t *i2c = i2c0;

const uint8_t i2c0_sda_pin = 24;
const uint8_t i2c0_scl_pin = 25;
uint8_t i2cBuf[64];
uint8_t rxBuf[8];

const uint8_t tx0_pin = 0;
const uint8_t rx0_pin = 1;

//****************************************
//* FUNCTION PROTOTYPES
//****************************************
int i2cWrite(i2c_inst_t *i2c,
             uint8_t devAddr,
             uint8_t regAddr,
             uint8_t *buf,
             uint8_t biteCnt);

int i2cRead(i2c_inst_t *i2c,
            uint8_t devAddr,
            uint8_t regAddr,
            uint8_t *buf,
            uint8_t biteCnt);
```



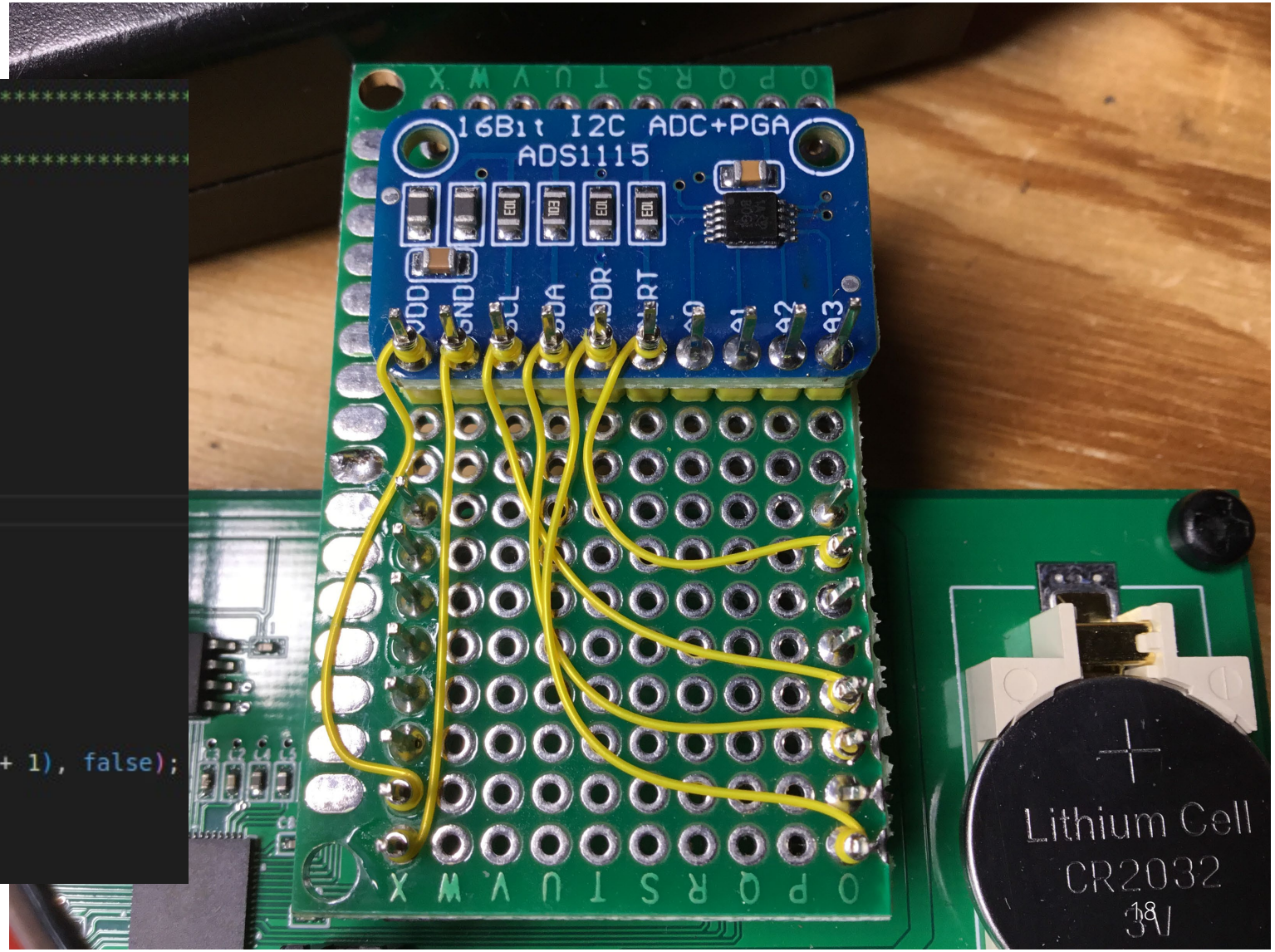
Hardware API – I²C

```
66  //*****************************************************************************
67  //* I2C READ
68  //*****************************************************************************
69  int i2cRead(i2c_inst_t *i2c,
70              uint8_t devAddr,
71              uint8_t regAddr,
72              uint8_t *buf,
73              uint8_t biteCnt)
74  {
75      uint16_t bites_read = 0;
76
77      // Read data
78      i2c_write_blocking(i2c, devAddr, &regAddr, 1, true);
79      bites_read = i2c_read_blocking(i2c, devAddr, buf, biteCnt, false);
80
81      return bites_read;
82  }
```

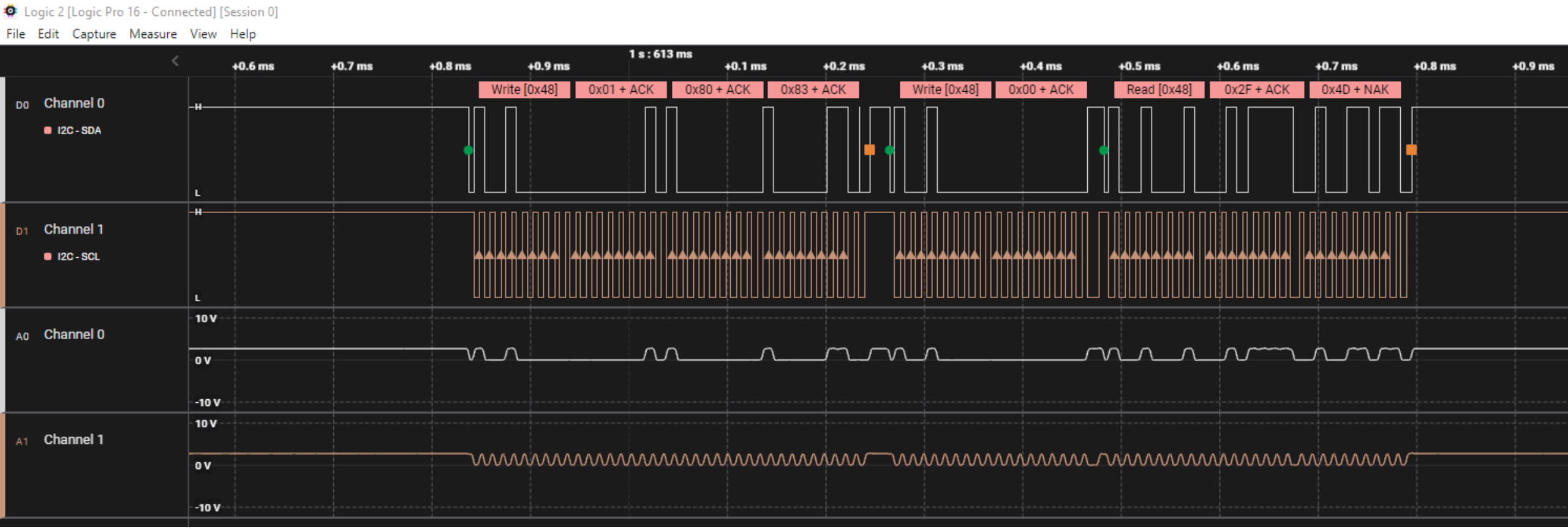


Hardware API – I²C

```
40  //*****************************************************************
41  //* I2C WRITE
42  //*****************************************************************
43  int i2cWrite(i2c_inst_t *i2c,
44               uint8_t devAddr,
45               uint8_t regAddr,
46               uint8_t *buf,
47               uint8_t biteCnt)
48  {
49      uint16_t i;
50      uint16_t bites_read = 0;
51      uint8_t txBuf[biteCnt + 1];
52
53      // Insert 8-bit addr at front of data packet
54      txBuf[0] = regAddr;
55      for (i = 0; i < biteCnt; i++)
56      {
57          txBuf[i + 1] = i2cBuf[i];
58      }
59
60      // Write data
61      i2c_write_blocking(i2c, devAddr, txBuf, (biteCnt + 1), false);
62
63      return bites_read;
64  }
```



Hardware API – I²C



Hardware API – SPI

Raspberry Pi Pico SDK: hardware_spi

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

Raspberry Pi Pico

SDK Documentation

v1.4.0

Search

- Introduction
- API Documentation
- Examples
- Additional Documentation +
- Web +

hardware_spi

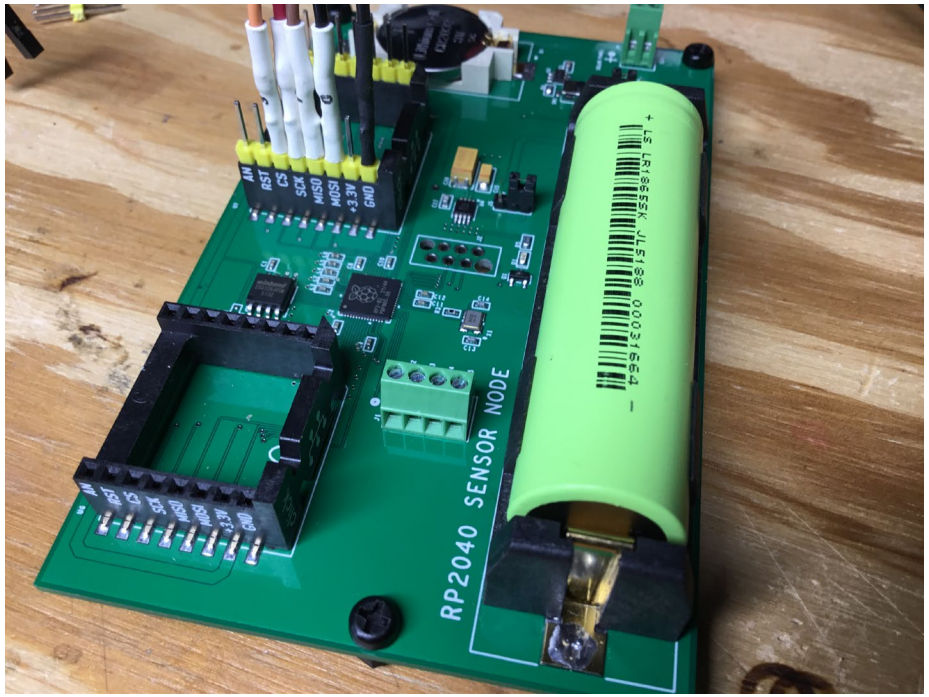
Hardware APIs

Macros

#define	spi0 ((spi_inst_t *)spi0_hw)
#define	spi1 ((spi_inst_t *)spi1_hw)

Enumerations

enum	spi_cpha_t { **SPI_CPHA_0** = 0 , **SPI_CPHA_1** = 1 } Enumeration of SPI CPHA (clock phase) values.
enum	spi_cpol_t { **SPI_CPOL_0** = 0 , **SPI_CPOL_1** = 1 } Enumeration of SPI CPOL (clock polarity) values.
enum	spi_order_t { **SPI_LSB_FIRST** = 0 , **SPI_MSB_FIRST** = 1 } Enumeration of SPI bit-order values.

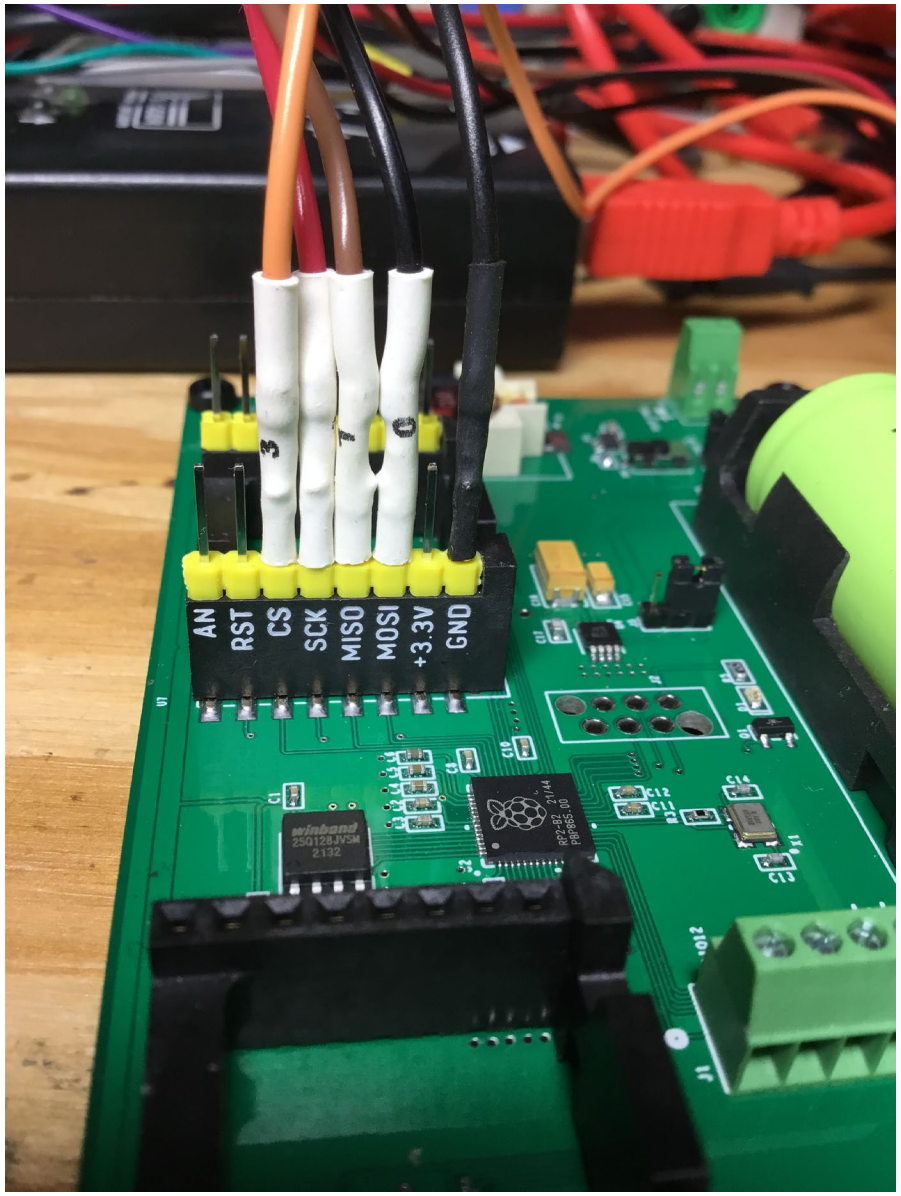


Hardware API – SPI

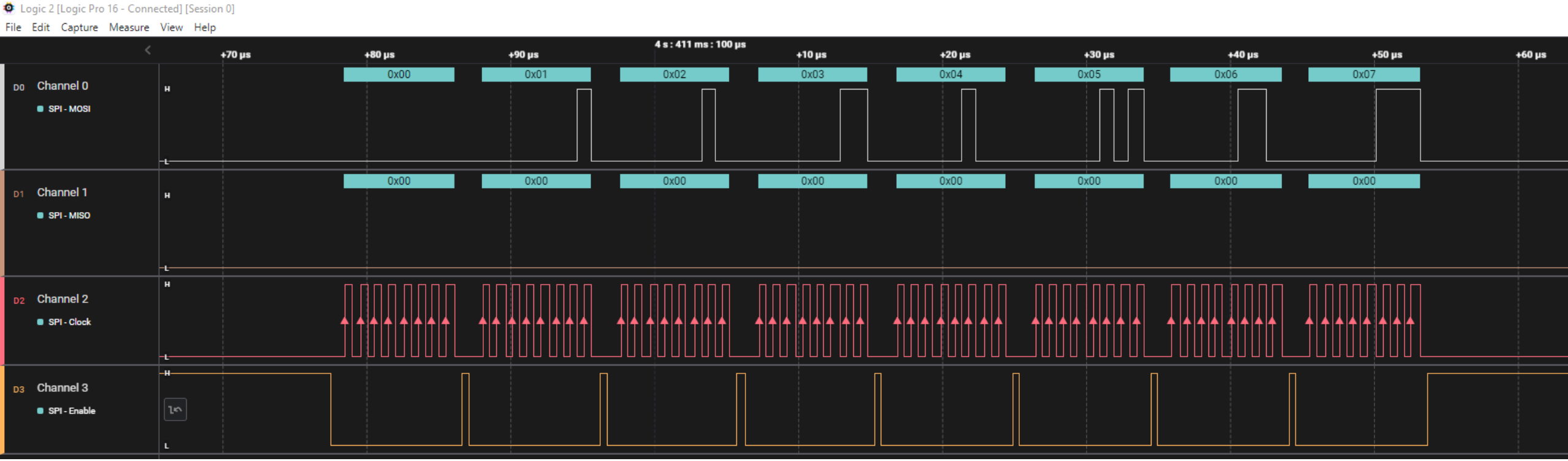
CMakeLists.txt | main.c

main.c

```c
#include <stdio.h>
#include <stdint.h>
#include "pico/stdlib.h"
#include "hardware/gpio.h"
#include "hardware/spi.h"

#define SCK_1_PIN   26
#define MOSI_1_PIN  27
#define MISO_1_PIN  28
#define CS_1_PIN    29

uint8_t txBuf[8];
uint8_t rxBuf[8];

int main()
{
    uint8_t i;
    // Enable SPI 1 at 1 MHz and connect to GPIOs
    spi_init(spi1, 1000 * 1000);
    gpio_set_function(SCK_1_PIN, GPIO_FUNC_SPI);
    gpio_set_function(MOSI_1_PIN, GPIO_FUNC_SPI);
    gpio_set_function(MISO_1_PIN, GPIO_FUNC_SPI);
    gpio_set_function(CS_1_PIN, GPIO_FUNC_SPI);

    // Initialize output buffer
    for (i = 0; i < 8; ++i)
    {
        txBuf[i] = i;
    }

    while(1)
    {
        // Write the output buffer to MOSI, and at the same time read from MISO.
        spi_write_read_blocking(spi1, txBuf, rxBuf, 8);
        sleep_ms(100);
    }
}
```



Hardware API – SPI

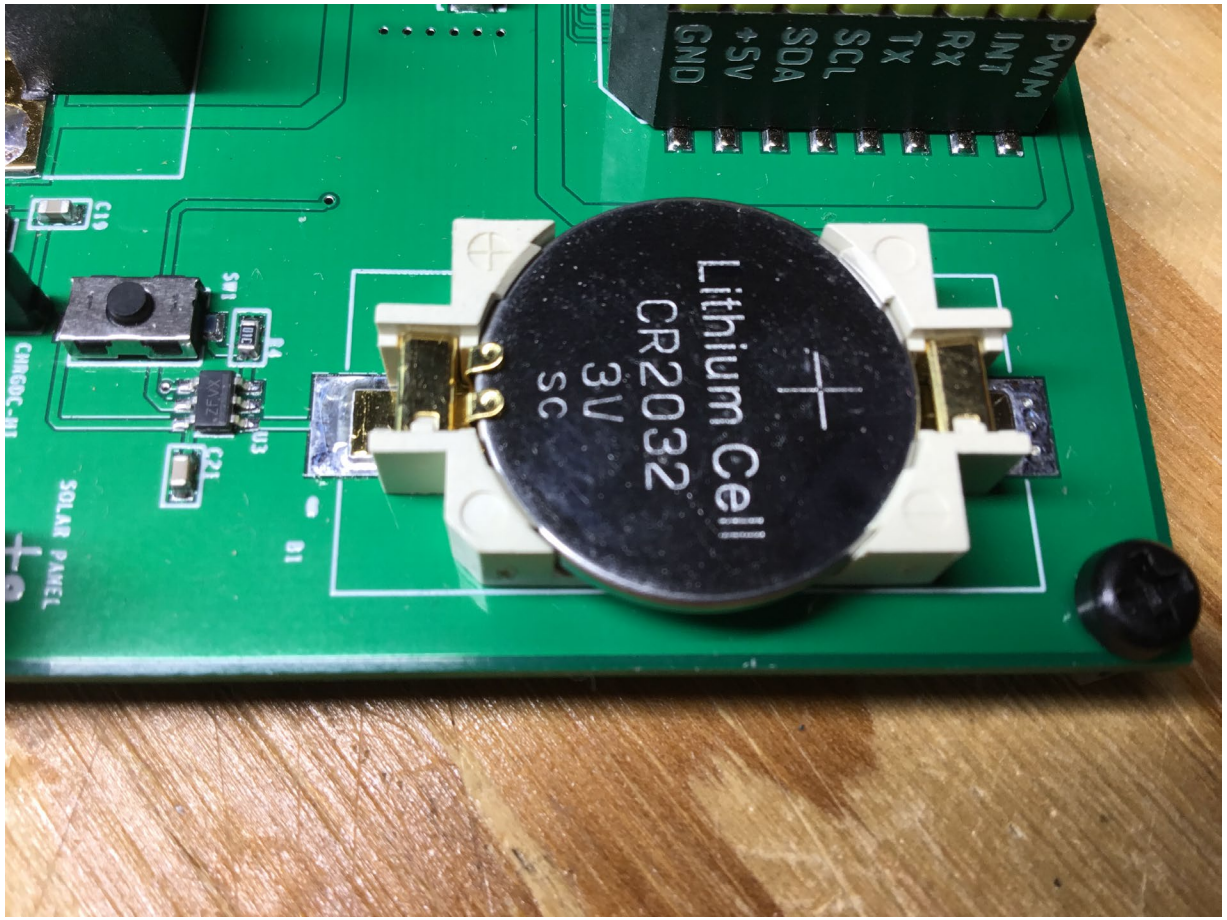


Thank you for attending!!!

MORE TO COME..

Please consider the resources below:

- raspberrypi.org
- RP2040 Data Sheet
- Raspberry Pi Pico C/C++ SDK
- https://raspberrypi.github.io/pico-sdk-doxygen/index.html



CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets