RASPBERRY PI PICO

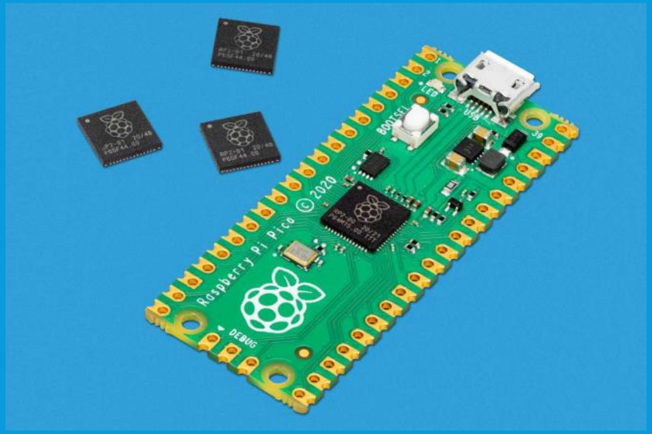
Seminar – Sperlhof

22.11.2023 – 24.11.2023

Dipl.-Ing. Gerald Pracherstorfer

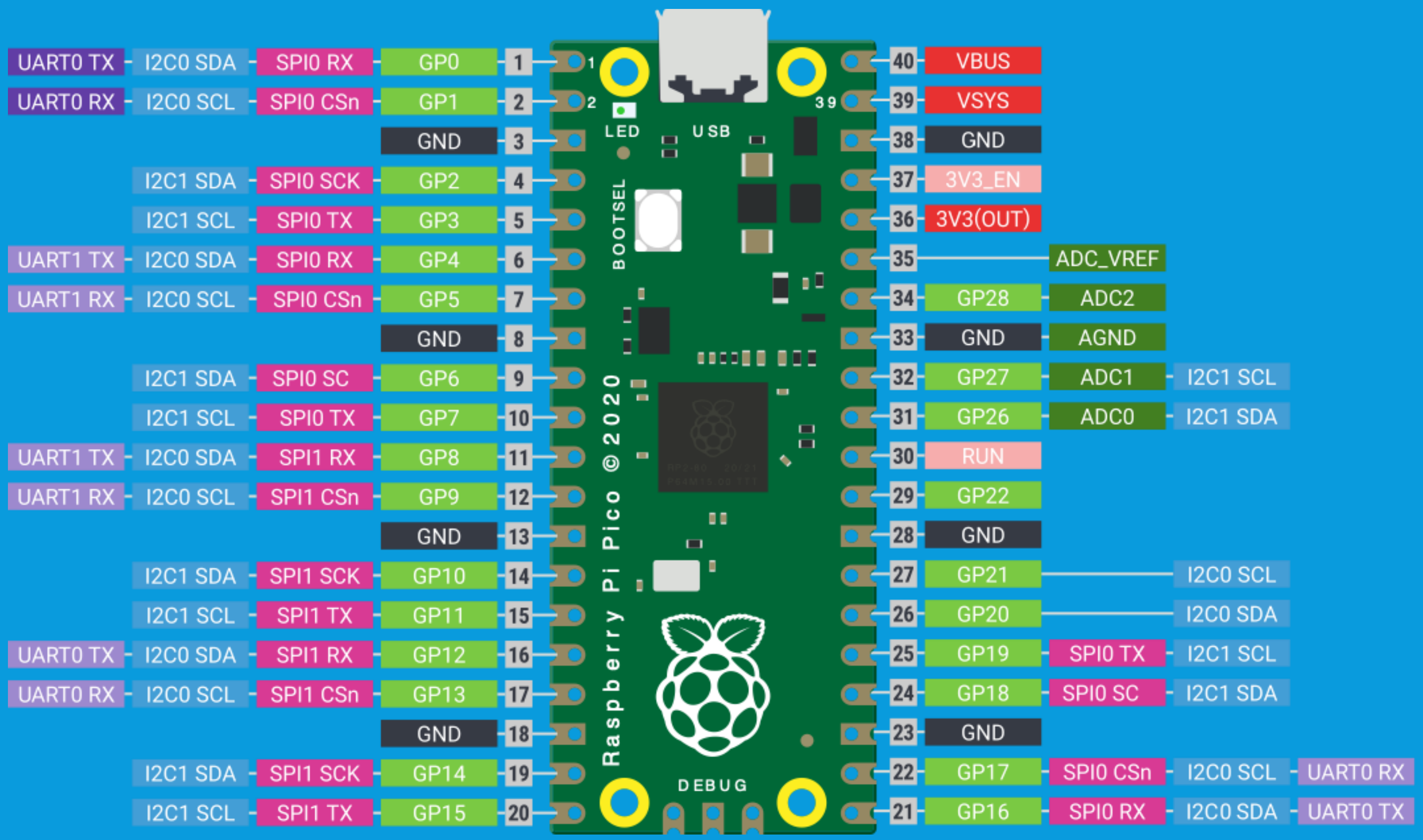
OVERVIEW

- Not a Linux machine: a microcontroller
- Designed by Raspberry Pi Foundation
- RP2040 microcontroller with on-board 2MB Flash memory
- Micro-USB B port for power and data (and for reprogramming the Flash)
- 40 pin 21×51 'DIP' style 1mm thick PCB
- 3-pin ARM Serial Wire Debug (SWD) port
- Simple yet highly flexible power supply architecture
- High quality, low cost, high availability
- Comprehensive SDK, software examples and documentation



PINOUT

- Minimal external circuitry to support the RP2040 chip
  - Flash (Winbond W25Q16JV)
  - Crystal
  - Power supplies and decoupling
  - USB connector
- On-board LED
- Buck-boost SMPS
- FLASH reprogramming
  - USB (Mass storage device)
  - SWD (Serial Wire Debug)
- Standard Interfaces



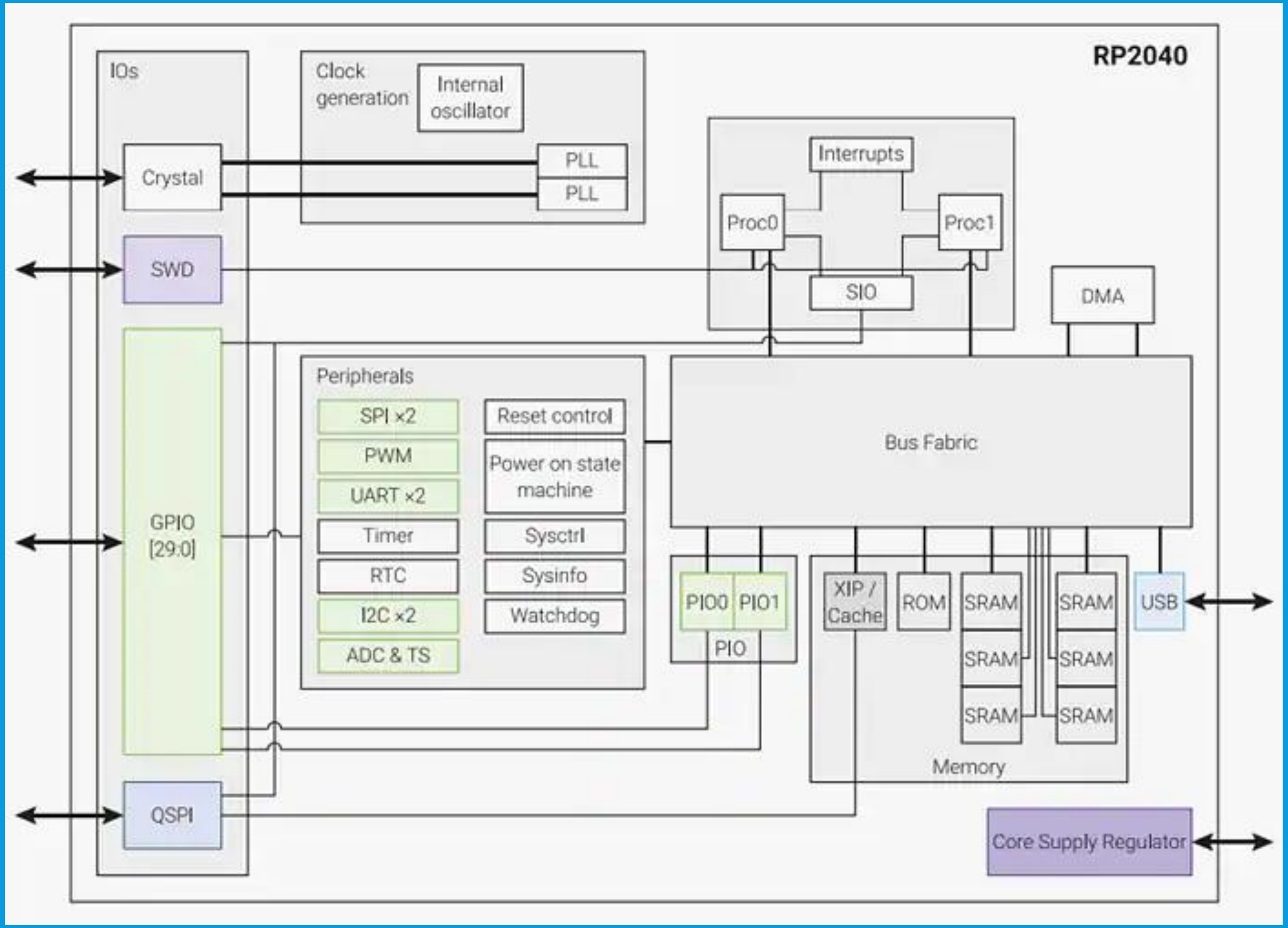
UART0 TX	I2C0 SDA	SPI0 RX	GP0	1
UART0 RX	I2C0 SCL	SPI0 CSn	GP1	2
			GND	3
	I2C1 SDA	SPI0 SCK	GP2	4
	I2C1 SCL	SPI0 TX	GP3	5
UART1 TX	I2C0 SDA	SPI0 RX	GP4	6
UART1 RX	I2C0 SCL	SPI0 CSn	GP5	7
			GND	8
	I2C1 SDA	SPI0 SC	GP6	9
	I2C1 SCL	SPI0 TX	GP7	10
UART1 TX	I2C0 SDA	SPI1 RX	GP8	11
UART1 RX	I2C0 SCL	SPI1 CSn	GP9	12
			GND	13
	I2C1 SDA	SPI1 SCK	GP10	14
	I2C1 SCL	SPI1 TX	GP11	15
UART0 TX	I2C0 SDA	SPI1 RX	GP12	16
UART0 RX	I2C0 SCL	SPI1 CSn	GP13	17
			GND	18
	I2C1 SDA	SPI1 SCK	GP14	19
	I2C1 SCL	SPI1 TX	GP15	20

40	VBUS			
39	VSYS			
38	GND			
37	3V3_EN			
36	3V3(OUT)			
35		ADC_VREF		
34	GP28	ADC2		
33	GND	AGND		
32	GP27	ADC1	I2C1 SCL	
31	GP26	ADC0	I2C1 SDA	
30	RUN			
29	GP22			
28	GND			
27	GP21		I2C0 SCL	
26	GP20		I2C0 SDA	
25	GP19	SPI0 TX	I2C1 SCL	
24	GP18	SPI0 SC	I2C1 SDA	
23	GND			
22	GP17	SPI0 CSn	I2C0 SCL	UART0 RX
21	GP16	SPI0 RX	I2C0 SDA	UART0 TX

RP2040 – KEY FEATURES

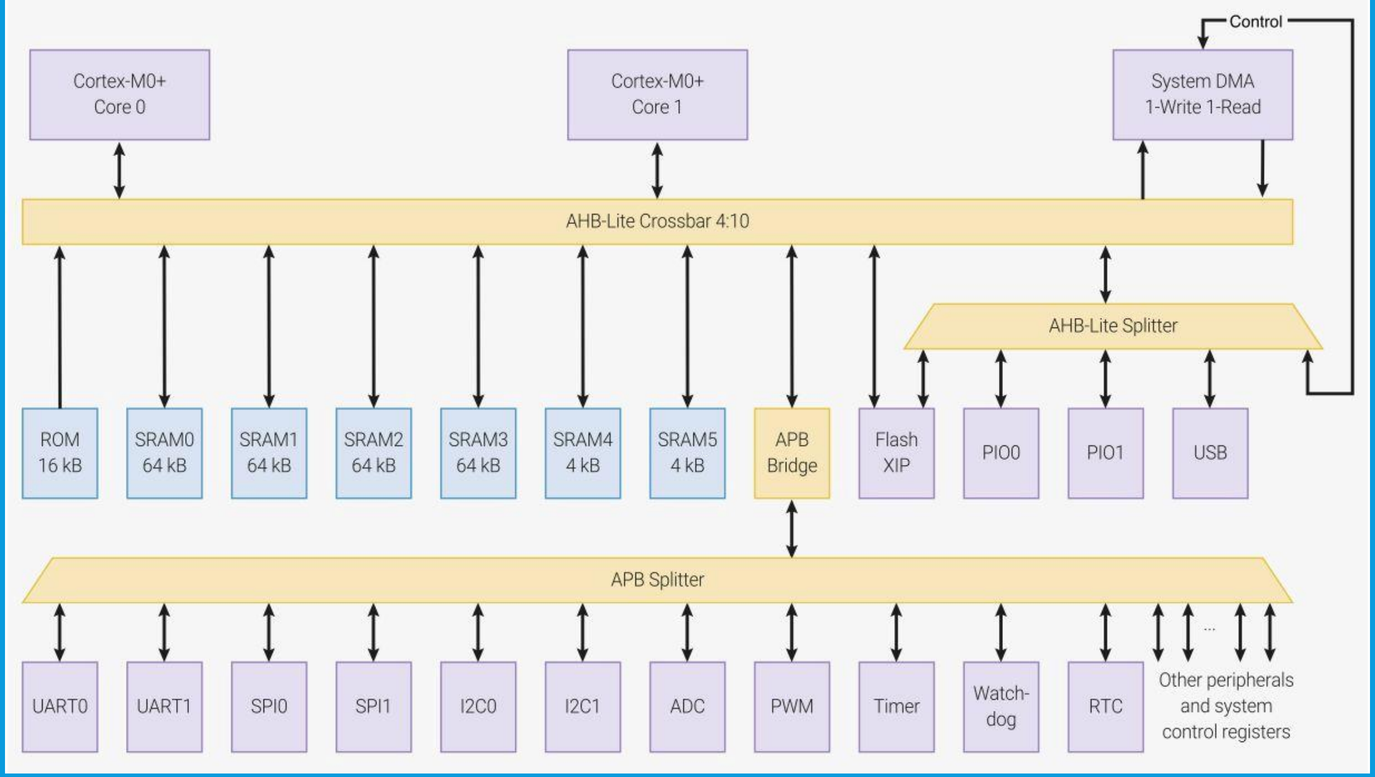
- Dual Arm Cortex-M0+ @ 133MHz
- 264kB on-chip SRAM in six independent banks
- Support for up to 16MB of off-chip Flash memory via dedicated QSPI bus
- DMA controller
- On-chip programmable LDO to generate core voltage
- 30 GPIO pins, 4 can be used as analogue inputs
- Peripherals
  - 2 x UART, 2 x SPI, 2 x I2C, 3 x 12-bit ADC, 16 x PWM
  - USB 1.1 controller and PHY, with host and device support
  - 8 x PIO state machines

RP2040 - ARCHITECTURE



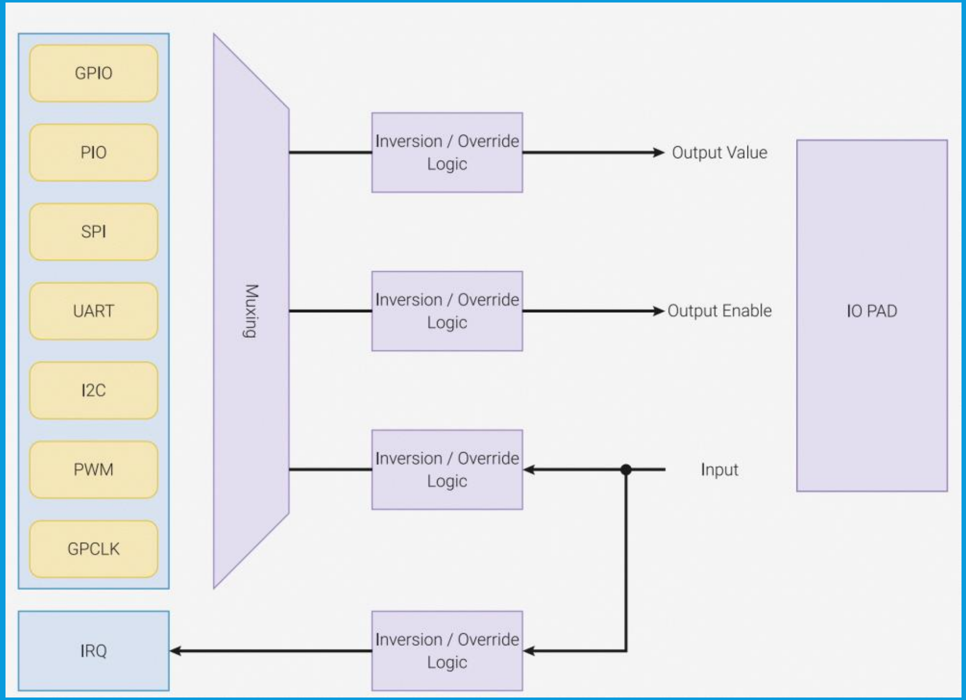
RP2040 – BUS FABRIC

- AHB (Advanced High-performance Bus): master connection
  - Core 0
  - Core 1
  - DMA controller read port
  - DMA controller write port
- APB (Advanced Peripheral Bus): connect to lower-bandwidth peripherals
  - UART
  - SPI
  - …

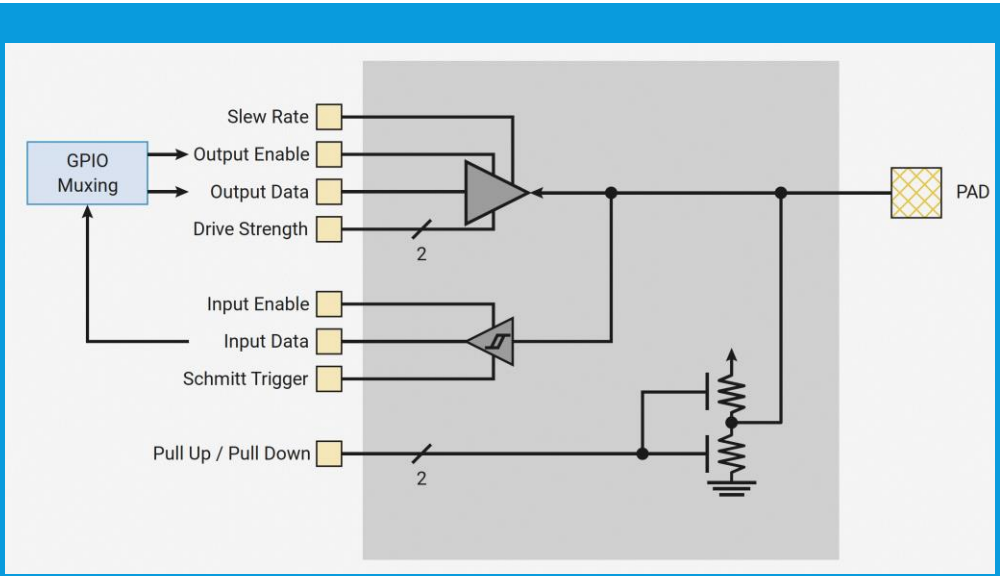


RP2040 - GPIO

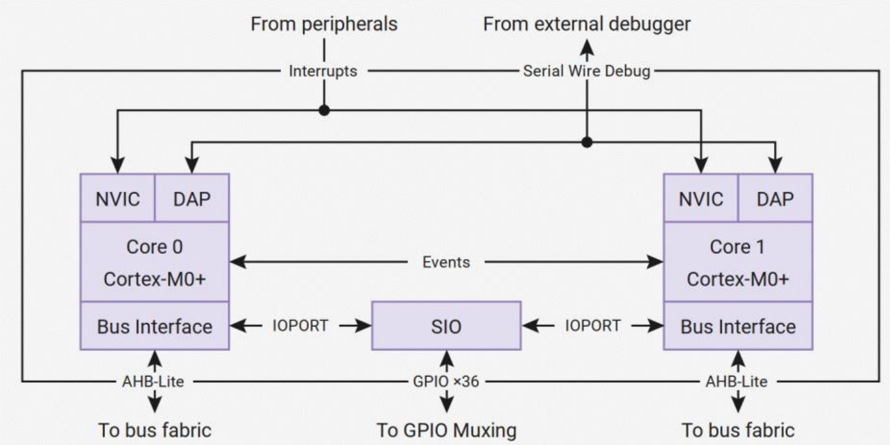
Logical Structure



GPIO
PIO
SPI
UART
I2C
PWM
GPCLK
IRQ
Muxing
Inversion / Override Logic
Inversion / Override Logic
Inversion / Override Logic
Inversion / Override Logic
Output Value
Output Enable
Input
IO PAD

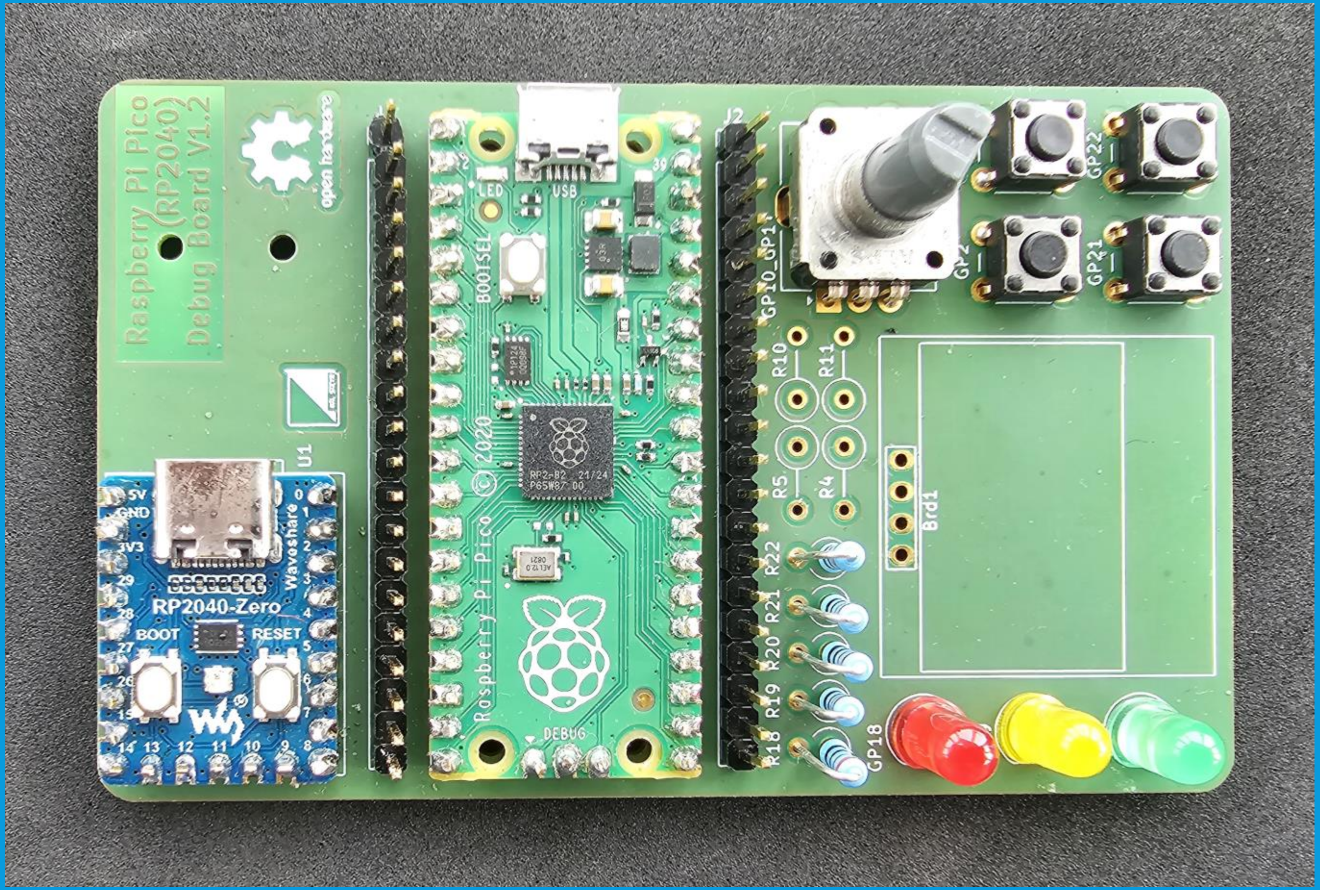


SIO



Pad

DEBUG BOARD


Raspberry Pi Pico
(RP2040)
Debug Board V1.2
open hardware
RP2040-Zero
Waveshare
BOOT
RESET
BOOTSEL
LED
USB
Raspberry Pi Pico
DEBUG
J2
GP10_GP1
GP22
GP21
GP2
R10
R11
R5
R4
Brd1
R22
R21
R20
R19
R18
GP18

RP2040 – BASIC FUNCTIONS

There are many functions provided by the SDK for configuring and working with GPIOs:

```
void gpio_init(uint gpio);

void gpio_set_function(uint gpio, enum gpio_function fn);

void gpio_set_dir(uint gpio, bool out);

bool gpio_get(uint gpio);

void gpio_put(uint gpio, bool value);

…
```

RP2040 – MASK FUNCTIONS

There are a range of mask functions which affect multiple GPIO lines:

```
void gpio_init_mask(uint32_t gpio_mask);
void gpio_set_dir_masked(uint32_t mask, uint32_t value);
void gpio_set_mask(uint32_t mask);
void gpio_clr_mask(uint32_t mask);
…
```

Example:

```
uint32_t mask = (1 << 3) | (1 << 5);
gpio_init_mask(mask); // initialise GPIO pin 3 and pin 5
```

RP2040 – TIME DELAY FUNCTIONS

- Sleep functions: for delaying execution in a lower power state

```
void sleep_ms(uint32_t ms);
void sleep_us(uint64_t us);
void sleep_until(absolute_time_t target);
```

- Busy Wait functions: to keep the processor alive during the wait phase

```
void busy_wait_ms(uint32_t delay_ms);
void busy_wait_us(uint64_t delay_us);
void busy_wait_until(absolute_time_t t);
```

C - PROGRAMMING

Example: Blinking light with state machine

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

#define LED_RED_PIN 18
#define LED_GREEN_PIN 20
#define DELAY_TIME 200

typedef enum  {LED_red, LED_green} state_t;

void setup();

int main() {
   state_t state = LED_red;
   bool led_red_on;
   bool led_green_on;

   setup();
   while (true) {

      switch(state){
        case LED_red:   state = LED_green;
                        break;
        case LED_green: state = LED_red;
                        break;
      }

      if (state == LED_red) led_red_on = true;
      else                  led_red_on = false;

      if (state == LED_green) led_green_on = true;
      else                    led_green_on = false;

      gpio_put(LED_RED_PIN, led_red_on);
      gpio_put(LED_GREEN_PIN, led_green_on);
      sleep_ms(DELAY_TIME);
   }
}
```

C - PROGRAMMING

```c
void setup() {
    gpio_init(LED_RED_PIN);
    gpio_set_dir(LED_RED_PIN, GPIO_OUT);
    gpio_init(LED_GREEN_PIN);
    gpio_set_dir(LED_GREEN_PIN, GPIO_OUT);
}
```

TOOL - CHAIN

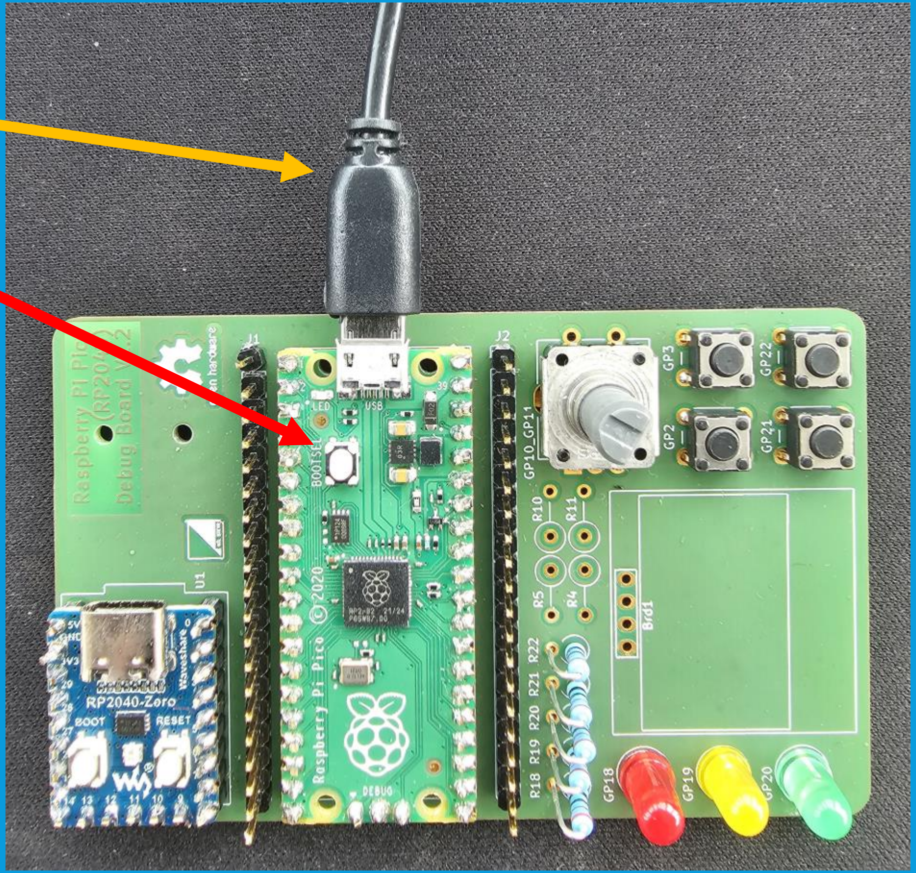
- CMake, GNU Embedded Toolchain for Arm (GCC, …)
- Pico – SDK
- Visual Studio Code
  - Create source code
  - Adjust project settings (CMakeLists.txt)
  - Build target (.uf2 file)
  - Debugging
- Serial Wire Debug
  - OpenOCD
  - GDB

FLASH – PROGRAMMING WITHOUT SWD

① Hold down the BOOTSEL button and connect the Pico to the development device

② Pico is forced into USB Mass Storage Mode

③ Drag and drop .uf2 file onto the Mass Storage Device



FLASH – PROGRAMMING / DEBUGGING WITH SWD

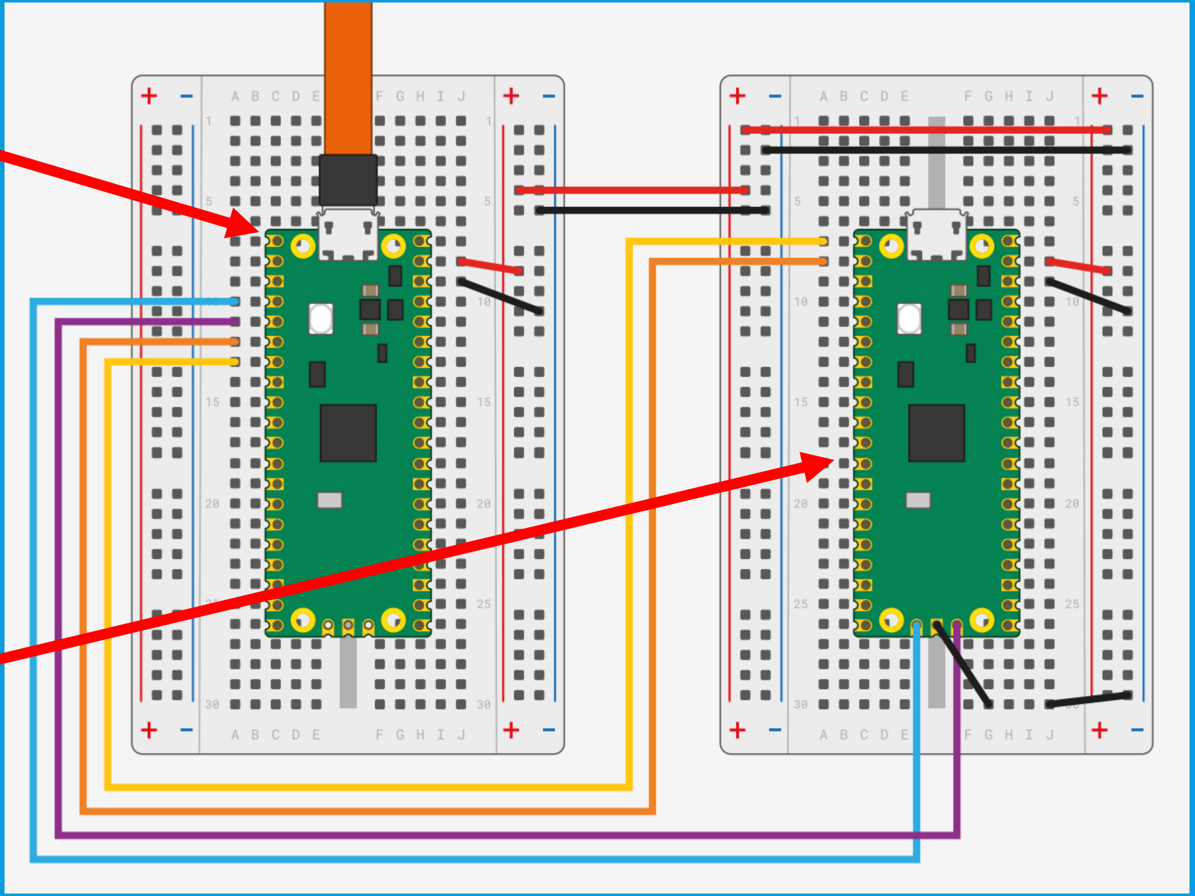
USB -> SWD and UART-Bridge

containing

Picoprobe firmware

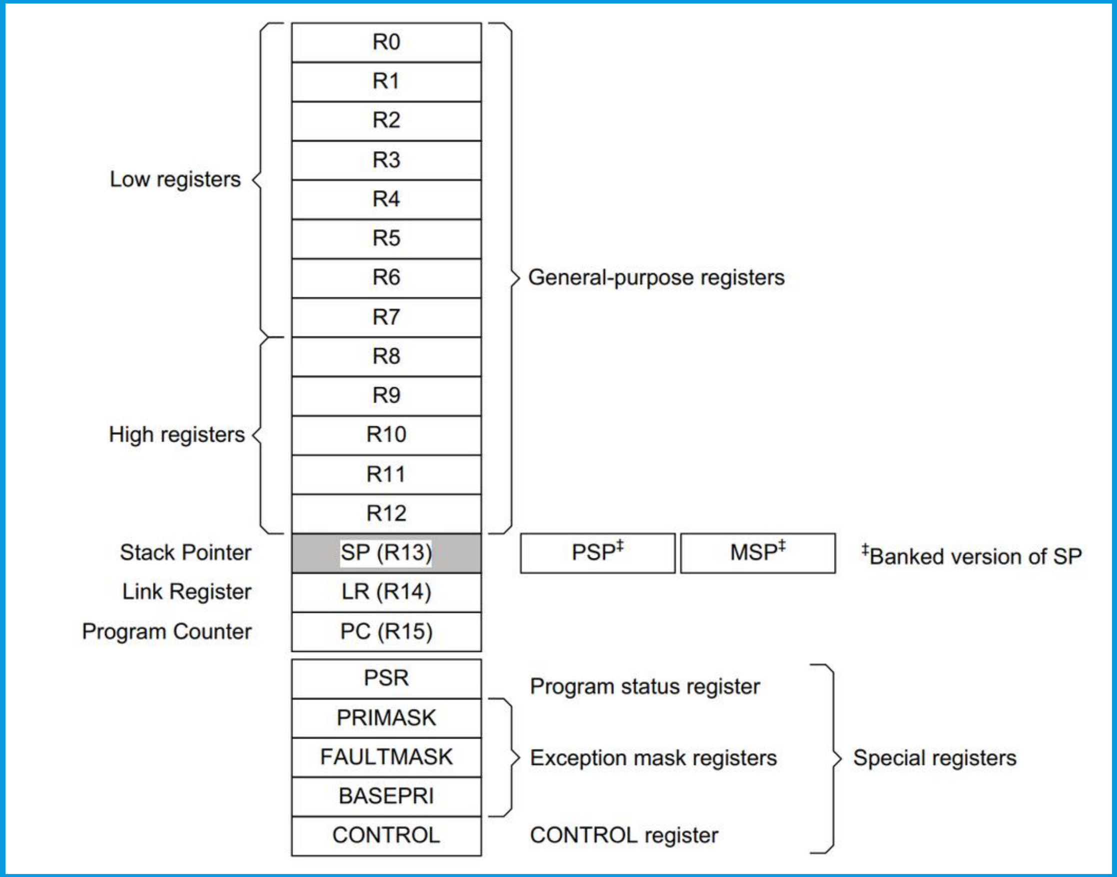
Target Pico

OpenOCD and GDB -> SWD protocol



ASSEMBLY PROGRAMMING

- Cortex Core Registers



Low registers

R0
R1
R2
R3
R4
R5
R6
R7

High registers

R8
R9
R10
R11
R12

General-purpose registers

Stack Pointer SP (R13) PSP‡ MSP‡

‡Banked version of SP

Link Register LR (R14)

Program Counter PC (R15)

PSR Program status register

PRIMASK
FAULTMASK
BASEPRI

Exception mask registers

CONTROL CONTROL register

Special registers

ASSEMBLY PROGRAMMING

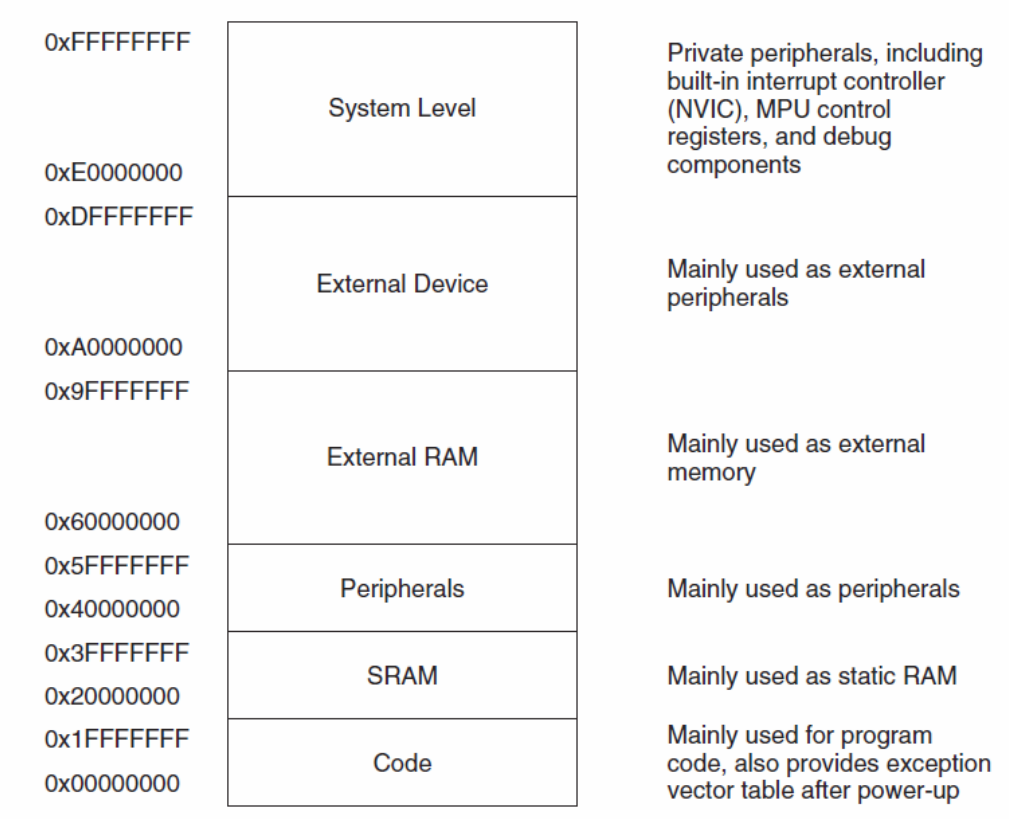
- ARM instruction set
  - 32-bit RISC based processor (Cortex-M0: 56 instructions)
  - 3-operand machine
- Addressing modes

Immediate addressing	e.g. MOV R1, #0x05
Register (Direct) addressing	e.g. ADD R5, R2, R3
Indirect addressing	e.g. LDR R8, [R10]

- Little endian format
- Instruction set summary: RP2040 datasheet pages 69 - 71

ASSEMBLY PROGRAMMING

- Memory Organization



0xFFFFFFFF 0xE0000000	System Level	Private peripherals, including built-in interrupt controller (NVIC), MPU control registers, and debug components
0xDFFFFFFF 0xA0000000	External Device	Mainly used as external peripherals
0x9FFFFFFF 0x60000000	External RAM	Mainly used as external memory
0x5FFFFFFF 0x40000000	Peripherals	Mainly used as peripherals
0x3FFFFFFF 0x20000000	SRAM	Mainly used as static RAM
0x1FFFFFFF 0x00000000	Code	Mainly used for program code, also provides exception vector table after power-up

ASSEMBLY PROGRAMMING

HTL STEYR

Example:

```c
unsigned int r1 = 0x0A;
unsigned int r2 = 0x05;

do {
   r1 = r1 * 2;
   r2 = r2 - 1;
} while (r2 > 0);
```

```asm
.thumb_func
.global main

main:      MOV R1, #0x0A
           MOV R2, #0x05
loop:      LSL R1, #1
           SUB R2, #1
           CMP R2, #0
           BNE loop
```

INLINE ASSEMBLY

Example:

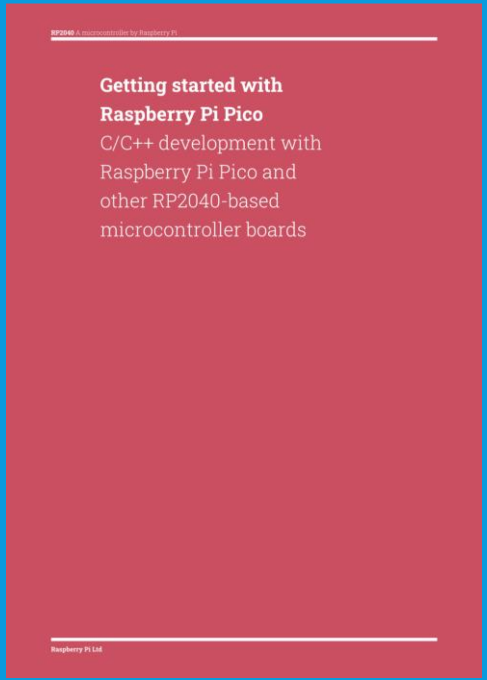
```
void main(){
  volatile int x = 10;
  volatile int y = 20;
  volatile int z;

  asm  ("MOV R0, %[x]\n"
        "MOV R1, %[y]\n"
        "ADD R2, R1, R0\n"
        "MOV %[z], R2"
        : [z] "=r" (z)
        : [x] "r" (x), [y] "r" (y)
        : "r0", "r1", "r2"
  );
  x = x + 1;
  asm ("end: B end");
}
```

DOCUMENTATION

HTL STEYR

- Data sheets, API guides, Tutorials, …

  https://www.raspberrypi.com/documentation/microcontrollers/

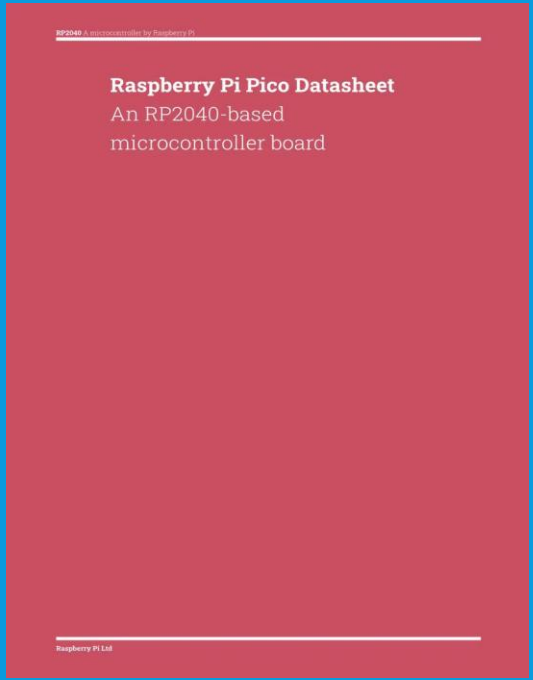


RP2040 A microcontroller by Raspberry Pi

Getting started with Raspberry Pi Pico

C/C++ development with Raspberry Pi Pico and other RP2040-based microcontroller boards

Raspberry Pi Ltd

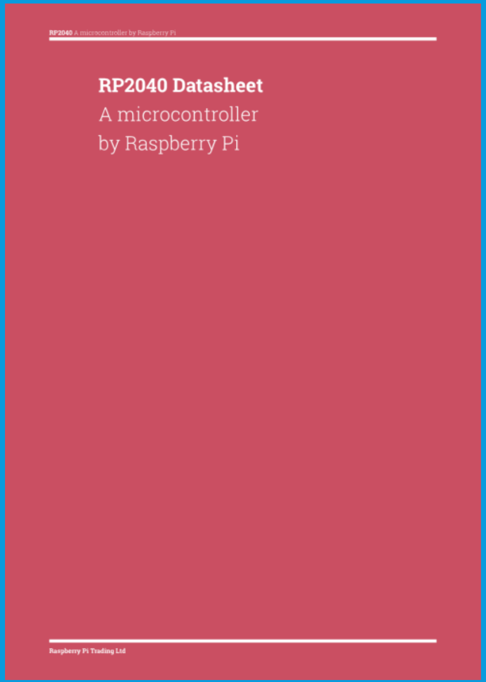


RP2040 A microcontroller by Raspberry Pi

Raspberry Pi Pico Datasheet

An RP2040-based microcontroller board

Raspberry Pi Ltd

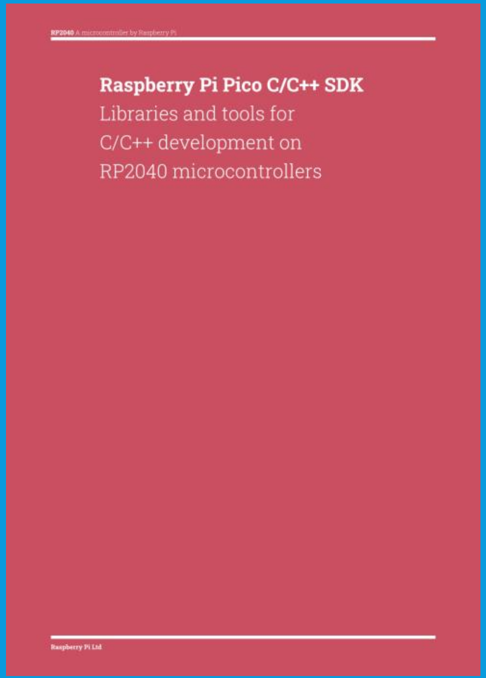


RP2040 A microcontroller by Raspberry Pi

RP2040 Datasheet

A microcontroller by Raspberry Pi

Raspberry Pi Trading Ltd



RP2040 A microcontroller by Raspberry Pi

Raspberry Pi Pico C/C++ SDK

Libraries and tools for C/C++ development on RP2040 microcontrollers

Raspberry Pi Ltd