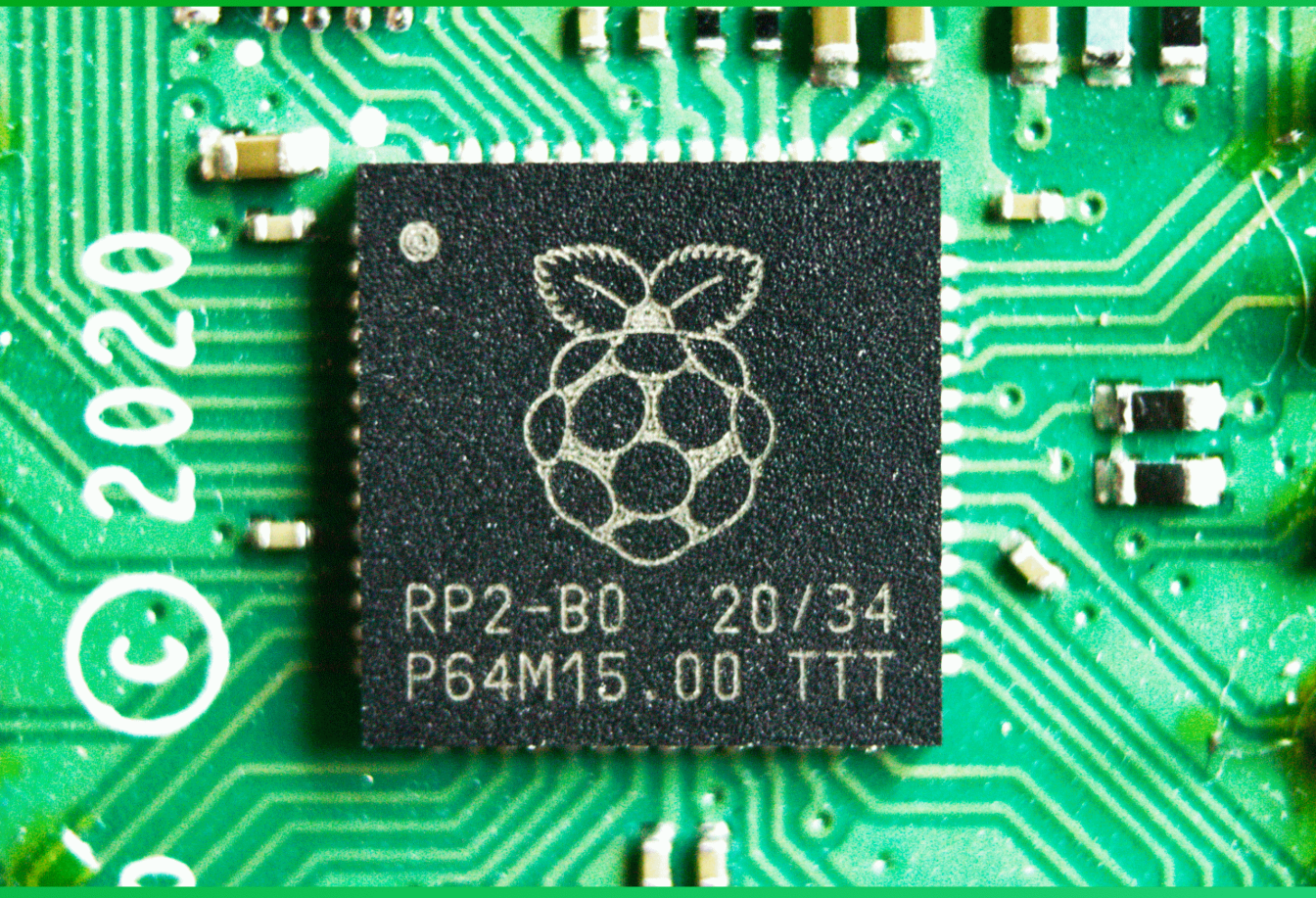
Knowing the RP2040

A Guide for Programmers



Daniel Quadros

Knowing the RP2040

A Guide for Programmers

Daniel Quadros

This book is for sale at http://leanpub.com/rp2040

This version was published on 2023-03-06

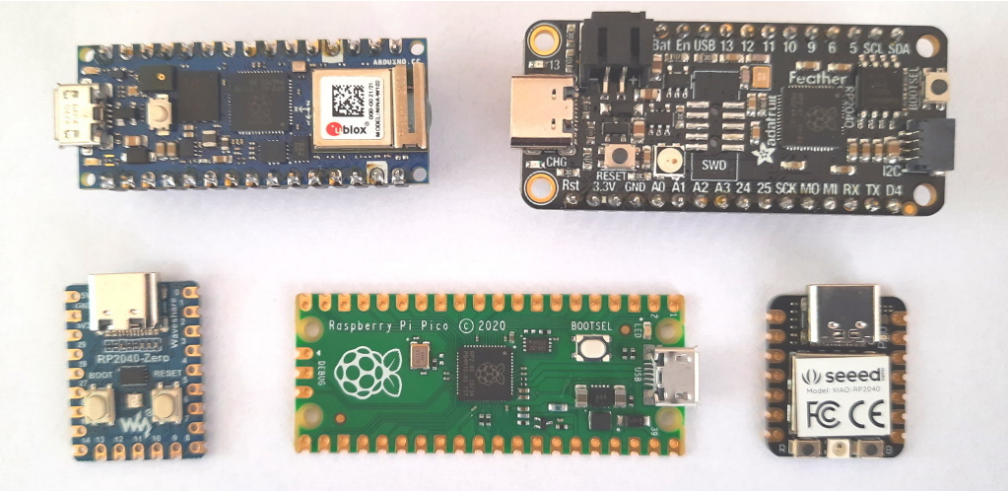
This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

On January 21th 2021 the Raspberry Pi Foundation announced a microcontroller board, the Raspberry Pi Pico. At its heart there is a brand new microcontroller, the RP2040. A quick browsing on the specs will show that it's very powerful: two ARM M0+ running at 133MHz, 264kbytes of Ram, all the popular interfaces (UART, I2C, SPI, ADC) and a somewhat magical Programmable I/O (PIO) subsystem.

At the same time the Pi Pico was launched, a few other companies (close partners of the Raspberry Pi Foundation) announced their own boards with the RP2040. Later on, the RP2040 was made available to everyone and there is now more than a dozen boards on the market, with more sure to come.



A Few RP2040 Boards

What this Book is About

The Raspberry Pi Foundation provides some pretty good documentation, including a datasheet and an SDK user guide, so you may ask why I am writing this book.

The answer is that the official documentation is more about “what things are” than “why this is important” or “how do I use it”. I’ve tried to explain things in a logical and clear way so you can get a better knowing of what the RP2040 is capable of and how to use it.

This is not a “project book”, so code examples are short and focused on an specific feature. This is also not a “hardware book”, you will not found here much talk about designing a board around the RP2040 (but I will talk a little about hardware on some points).

The SDK functions

The C/C++ SDK includes many libraries. Most of the functions in these libraries provides a way to interact with the hardware, abstracting the low level registers in the RP2040.

I will not try to cover every function in the SDK. Instead I will focus on the functions I believe are the most useful for typical programs.

You can check the full list of SDK functions in the official documentation at https://raspberrypi.github.io/pico-sdk-doxygen/

The Examples

The examples where written in C, using the Pico SDK version 1.5.0. Like many, I am not particularly fond of the manual installation process for the SDK (specially on Windows) and the use of CMake may be a hurdle to those more accustomed to programming under the nice umbrella of an user-friendly IDE or know only about makefiles. But the Pico SDK is the official way to access the low level stuff we are going to see.

The examples were tested on a Pi Pico and should run on other boards, eventually with changes regarding the available pins.

All code from the examples can be download from https://github.com/dquadros/KnowingRP2040

Whom this Book is For

This is what I would call an **intermediate book**, going into a few advanced topics.

A assume the reader has a little experience with microcontrollers and some very basic knowledge of electronics.

Anyone who knows the basics of the C language should have no trouble understanding the examples.

Acknowledgments

Looking back, there are too many people that, one way or another, have helped me come to the point where a could write this book. This is where I mention and thank a few of them.

First, my mother and father who nurtured my curiosity and addicted me to reading.

There were many teachers that not only gave important lessons, but also encouraged me to learn more.

In my professional life I am grateful for all that believed I could deliver and those that helped me do so.

A special mention to the late Alberto Fabiano, who introduced me to the wild community of hackerspaces. And to Fabio Souza and Tiago Lima at Embarcados for all their work at spreading knowledge and their support to my technical writings.

Mauricio Aniche, my son-in-law, awarded professor and famous author, was a constant encourager when I was giving up to procrastination.

Of course this book would not exist if not for the patience of my wife Cecilia while I sent days in front of a PC and playing with all those “little boards”.

Updates

This is the second update after the book was “finished”.

The changes for the first update include:

- A more precise explanation of the SIO registers
- Details on GPIO interrupts in Chapter 7, including a new example.
- The new Appendix D on how to access the RP2040 registers.
- Correction of typing errors and small improvements on explanations.

The changes for the second update include:

- Using the SDK v1.5:
  - Updating the description of SDK functions where necessary.
  - Updating the code where necessary (in particular the USB host example).
  - Adding the description and an example of the new library for I2C slave.
- Mentioning the “Raspberry Debug Probe” in Appendix C
- Correction of typing errors and small improvements on explanations.

How to Send Feedback

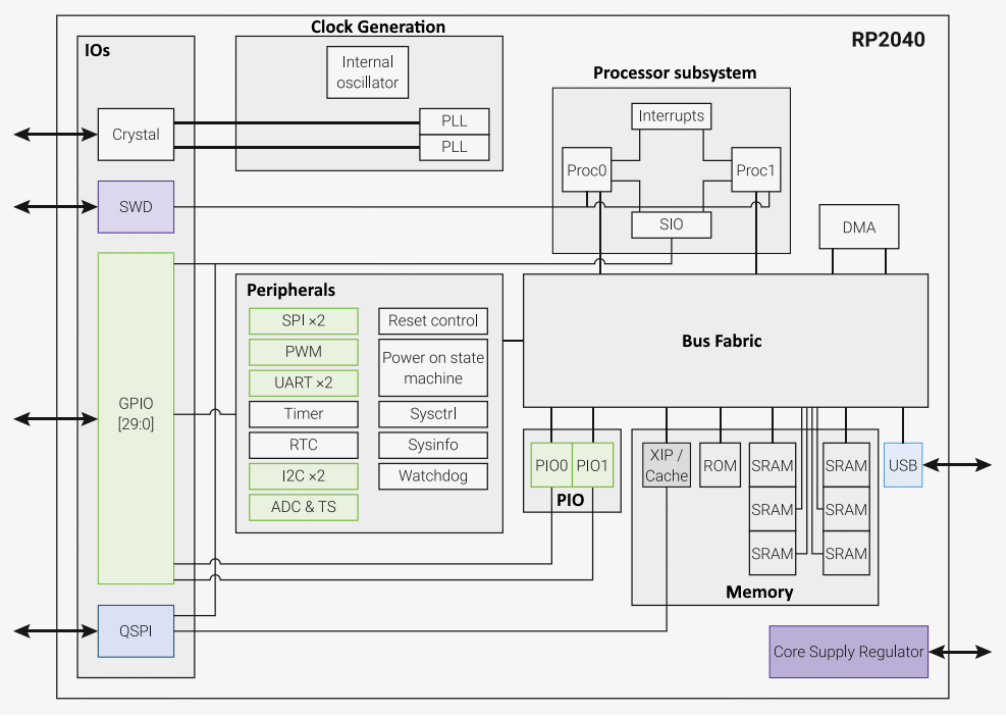
If you find an error, have a suggestion or comment, you can reach me by:

- Clicking in the “Email the Author” button in the book page in leanpub.com
- Sending an email to dqsoft.blogspot@gmail.com
- Sending me a message in Twitter (@DQSoft)

The RP2040 Architecture

An overall view is usually a good starting point. When it comes to microcontrollers, an architectural diagram will show important things like how the memories are connect to the processor and what kind of peripherals are available.

The following picture is adapted from the RP2040 datasheet and shows the RP2040 architecture:



The RP2040 Architecture

In this chapter I will not delve deep into each part, I will just give a general idea of what they do and mention some important points. The next chapters will go into the details.

Processor Subsystem

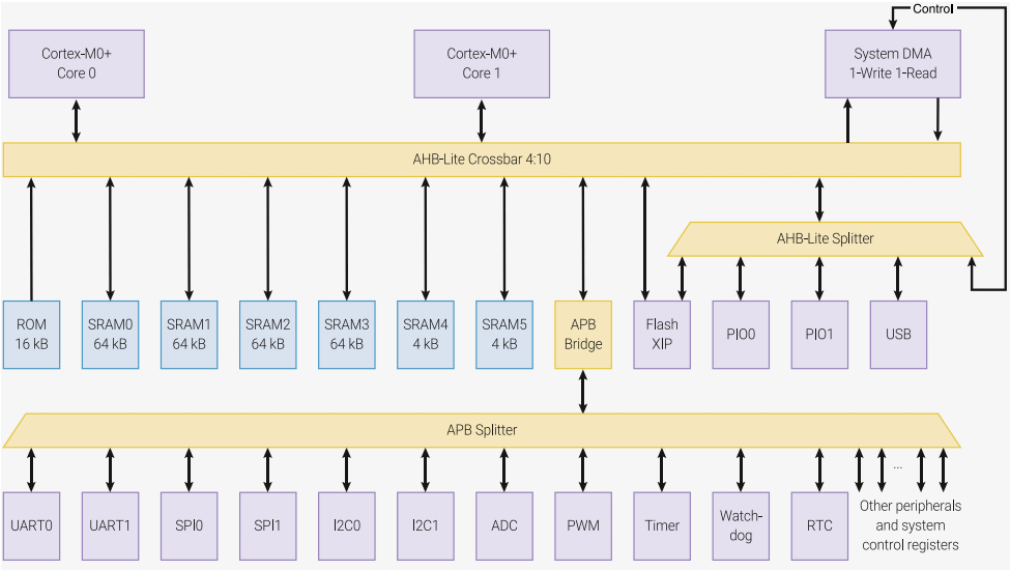
Let's start by the block called **Processor Subsystem**. Here we have the two **ARM Cortex M0+** cores and, connected to both, the **SIO** (Single-cycle IO block).

The ARM core is where the instructions in our software will be decoded and executed. As we will see later, the ARM M0+ is a very powerful RISC processor designed for embedded applications.

The SIO gives the cores low-latency, deterministic access to some peripherals (in the form of memory mapped registers), including the GPIO. In other words, the SIO makes it easier to write code that will run in both cores and use the same peripherals. It is designed for fast synchronization and communication between the two cores.

Bus Fabric

The **Bus Fabric** gives the processor cores access to memory and other peripherals. Like many ARM processors, there are two main buses, the **AHB** (called here *AHB-Lite Crossbar*) and the **APB**.



The RP2040 Bus Fabric

The AHB-Lite Crossbar routes addresses and data between 4 upstream ports (the two cores and DMA read and write) and 10 downstream ports. Up to four AHB bus transfers can take place each cycle.

The APB routes access to the more slower peripherals, an APB bridge connects the two buses.

Address Map

All resources on the RP2040 are mapped to memory addresses between 0x00000000 and 0xF0000000:

Address	Resource
0x00000000	ROM
0x10000000	XIP
0x20000000	SRAM
0x40000000	APB Peripherals
0x50000000	AHB-Lite Peripherals
0xD0000000	IOPORT Registers
0xE0000000	Cortex-M0+ Registers

Clock Generation

The RP2040 has a sophisticated clock generation subsystem. There are three basic clock sources:

- Ring oscillator: this internal oscillator needs no external components but has very vague frequency guarantees (typically 6MHz, expected between 4 and 8MHz, can be anywhere from 1.8 to 12 MHz, can change with voltage or temperature).
- Crystal oscillator: the hardware reference design (and all boards so far) have an external 12MHz crystal connected to this oscillator. The output of the crystal oscillator is fed to two PLLs that can generate higher frequencies for USB and system clock.
- External clocks: up to three clocks, with frequencies up to 50MHz.

These sources are connected to ten clock generators; each generator can be configured to select the source and the clock divisor. The output of these generators goes to the other subsystems.

Memory

In this subsystem we have the elements: RAM, ROM and XIP/Cache.

There are six RAM blocks (four with 64kB and two with 4kB). As we will see, this allows the two cores to access Ram with no contentions.

The 16kbytes ROM comes from factory with the startup firmware. Among other things, it will create an USB mass storage device used to write software to the Flash memory.

This Flash memory is not inside the RP2040. It is an external component, attached to GPIO pins controlled by the QSPI interface. Software can be run directly from the external Flash (eXecute In Place or XIP). To alleviate the delays of serial access to the external Flash, there is a 16kbytes cache.

PIO

The **Programmable Input Out** allows the RP2040 to efficiently implement many hardware protocols that need to be implemented by "bit-banging" (careful manual control of GPIOs pins) in other microcontrollers.

The RP2040 has two PIOs, each with 4 **state machines**. This state machines can interact with GPIOs by executing short programs. All the state machines run in parallel to the ARM processors. Data exchange between the state machines and the processors are done through hardware queues to alleviate timing requirements.

The result is a system where precise timing and constant signal monitoring can be achieved with minimum processor overhead, even when very short times are required.

Peripherals

The RP2040 has the following peripherals:

- 2 UARTs (Universal Asynchronous Receiver Transmitter)
- 2 SPI (Serial Peripheral Interface)
- 2 I^2C (Inter-integrated Circuit)
- PWM (Pulse Width Modulation)
- ADC (Analog to Digital Converter)
- Timer
- RTC (Real Time Clock)

IOs

The **IOs** subsystem includes

- The crystal driver
- GPIOs - the general purpose I/Os and the pin drivers (called **Pad** in the documentation)
- QSPI - High speed SPI for connecting the Flash memory where firmware will reside
- SWD - Serial Wire Debug. This interface gives an external debug the ability to load software in Ram or Flash, control processor execution, access memory.

Future RP Microcontrollers?

While there is no official word (so far) about other RP microcontroller, the coding of the name gives some idea of what the Raspberry Foundation expects to change in future chips:

- Different number of cores. A single core could be an even cheaper option, more than 2 cores seem a little overkill, but who knows?
- A different type of core. The probable candidates would be the M3 (more computing power) and the M4 (DSP instructions and, optionally, floating point).
- More (or less) Ram. Less Ram would make the chip cheaper and/or open space to other features. More Ram may be interesting for a more powerful core type.
- Addition of nonvolatile memory, probably Flash. This would make boards simpler and also give faster access to nonvolatile code and data.

The Cortex-M0+ Processor Cores

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Unprivileged and Privileged Execution

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Debugger Support

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Memory Protection Unit (MPU)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Instruction Set

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SIO

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Hardware Spinlocks

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Inter-processor FIFOs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Systick Timer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

pico_multicore

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

pico_sync

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Reset, Interrupts and Power Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Reset

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Interrupts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Power Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Top-level Clock Gates

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Sleep States

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Memory, Addresses and DMA

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Memory in the RP2040

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The ROM

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The SRAM

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Flash Memory

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Addresses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

XIP

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SRAM

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

APB Peripherals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

AHB-Lite Peripherals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Direct Memory Access (DMA)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Channel Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Configuring and Starting a Channel

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Data Request (DREQ) and Pacing Timers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Interrupts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

CRC Calculation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

DMA Usage Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Collecting Data from the ADC using DMA

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Sending Data to a SPI LCD Display using DMA

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Generation, Timer, Watchdog and RTC

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Generation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Overview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

ROSC - Ring Oscillator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

XOSC - Crystal Oscillator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

External Clocks

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PLLs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Frequency Counter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Generator Multiplexers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Timer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

`pico_time` Selected Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Watchdog

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

RTC

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

GPIO, Pad and PWM

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

GPIO Overview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Function Select

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PADs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Digital Input and Output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Digital Output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Digital Input

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

GPIO Interrupts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Level Interrupts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Edge Interrupts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PWM

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PWM Slice

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Pins Assignment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Divider for PWM Generation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Basic PWM Generation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Some Fine Details of PWM Generation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Measuring Frequency and Duty Cycle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The Programmable I/O (PIO)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The PIO State Machines

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The FIFOs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Programmer's Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PIO Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

GPIO Pins Mapping

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

FIFOs Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Program Wrapping

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Interrupt (IRQ) Flags

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The Instructions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

JMP

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

WAIT

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

IN

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

OUT

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PUSH

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PULL

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

MOV

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

IRQ

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SET

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Flow Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Coding, Compiling and Running PIO Programs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

PIO Assembly Language

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Directives

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Values and Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Labels

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

State Machine Allocation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Program Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

FIFO Usage

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Miscellaneous Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

A simple square wave generator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Sending data serially

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Receiving clocked serial data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Interfacing a HC-SR04 Ultrasonic Sensor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Asynchronous Serial Communication: the UARTs

Asynchronous serial communication is one of the oldest form of serial communication. Bits are sent serially (one after the other) over a wire with no common clock signal to synchronize the receiver to the transmitter and determine where are the individual bits. A communication speed (called, not precisely, *baud rate*) must be previously agreed by the two sides.

The RP2040 has two UARTs, with the following features:

- 32 position queues (*FIFO*s - First In First Out) for transmission and reception
- programmable baud rate generator
- support for 5, 6, 7 and 8 bits of data, 1 or 2 stop bits, parity none, even or odd (see framing in the next section)
- break detection and generation
- support for hardware flow control
- interrupt and DMA support

The UARTs in the RP2040 are based on the PL011 (a standard UART design by ARM), but does not implements all its features.

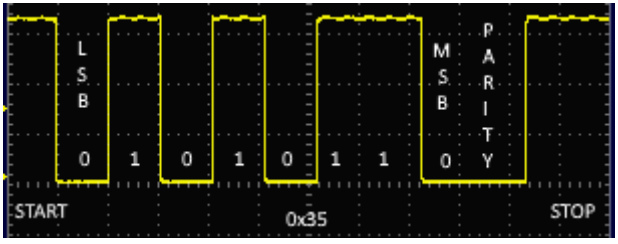
Framing

When no communication is taking place, the signal stays at a high ("1") level.

Individual words (typically a byte) are sent as a "frame" composed of:

- **start bit**: The signal goes to low level and stays there for a "bit time" (as defined by the baud rate). The change from high to low signals the receiver that a frame is starting and is used to determine where individual bits will be.
- **data bits**: The individual bits of the word. The least significant bit is send first and the most significant bit is send last.
- **parity bit**: optional bit used to detect communication errors. If using *even parity*, the total number of "1" bits (considering the data bits and the parity bit) is even. In *odd parity* the total number of "1" bits is odd.

- **stop bit(s)**: the signal is kept at high level to signal the end of the frame. Some really old equipment required two stop bits (that is, that the signal was kept high for at list two bit times before the next start bit), but nowadays one stop bit is standard. As the beginning of the next start bit is asynchronous to the stop bits, the line can be keep high for anytime after the minimum stop bit.



UART Framing - Transmission of 0x35 7 bits, even parity

A special condition (called *break*) is signaled by keeping the signal at low level for a whole frame (or more).

FIFOs

Each UART has two FIFOs, one for reception and one for transmission.

The transmission FIFO can store up to 32 8-bit words. Data written to this FIFO will be consumed by the transmitter. The Tx FIFO can be disabled to act like a single data-to-transmit register.

The reception FIFO holds 32 12-bit words. The Rx FIFO can be disabled to act like a single data-received register. The received data is in the lower 8 bits, the upper 4 bits contains the following flags:

- bit 11: **OE** (Overrun Error). This bit will be one if data is received when the FIFO is full, indicating that data was lost. This bit is not associated with the received data, it will only return to zero when a new frame is received and there is space for it in the FIFO.
- bit 10: **BE** (Break Error). If a break condition is detected, a word with this bit set and data equals zero will be put in the FIFO. A new word will be generated only after the line goes back to high level (ending the break condition).
- bit 9: **PE** (Parity Error). This bit will be one if data is received with the wrong parity;
- bit 8: **FE** (Framing Error). This bit will be one if a valid stop bit is not detected. This can be caused by a communication error, wrong frame format or wrong baud rate.

Control Signals and Hardware Flow Control

PC users may recall the use of serial communications with modems and phone lines to connect to the Internet. Standards define a number of control signals between a computer (or *DTE* - Data Transmission Equipment) and a modem (or *DCE* - Data Communication Equipment).

While the registers for the UARTs refer to many of these signals (manly for compatibility with the chips used in old IBM PCs), the RP2040 actually supports only two of them:

- **RTS** (Request To Send) this is an output signal that goes (in the standard) to high level to indicate that the DTE wants to transmit
- **CTS** (Clear To Send) this an input signal, high level means that the DCE can accept data from the DTE

The use of these signals is called **hardware flow control** as opposed to **software flow control** (where special characters or messages in the data signal when transmission must stop).

The RP2040 UARTs support the use of RTS and CTS in a non-standard way to control reception and/or transmission:

- In *RTS flow control*, the RTS signal is used to inform the other side when to transmit. It will be high as long as there is a configurable space in the reception queue. When the reception queue fills up, RTS goes down to inform the other side to stop transmission.
- In *CTS flow control*, transmission of each word only starts when CTS is high.

To use the hardware flow control, RTS and CTS pins must be configured and flow control enabled.

In a typical hardware flow control configuration, you will enable both options and cross the RTS and CTS signals of the two sides.

Baud Rate Generation

Strictly speaking, baud rate is the number of *symbols* transmitted per unit of time while bits per second (bps) is the number of bits transmitted per second. In the UART a symbol is one bit, so it is common to use the term baud rate when bits per second would be more appropriate.

The UART will generate the baud rate from its clock `clk_peri` (FUARTCLK in the docs) using a fractional divider, as long as

- `clk_peri` is at least 16 times the baud rate

- `clk_peri` is at most 16*65535 times the baud rate
- `clk_peri` is at most 5/3 of the processor clock `clk_sys`(FPCLK in the docs)

The baud rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. It divides the UART clock to generate an internal Baud16 clock that is 16 times the baud rate.

For example, suppose we are using the standard 125MHz for `clk_sys` and `clk_peri`. To generate 9600 bps we need to use a divisor of 125000000/(16*9600) = 813.802. 813 is the integer part, the 6-bit fractional part is integer (0.802*64+0.5) = 51.

Note that the resulting baud is not exact, as 51/64 = 0.796875, The actual baud rate (not taking in account the clock precision) is 125000000/(16*813.796875) = 9600.06 (a very low error).

These calculations can be done by the `uart_set_baudrate()` function in the SDK.

UART Status and Interrupts

In the RP2040, the UART uses a single combined interrupt in the processors (UARTIMTR). There are eleven possible reasons for this interrupt and they can be independently masked:

- UARTMSINTR: modem status interrupt, generated when a modem status changes (in the RP2040 this can only be the CTS signal, the PL011 has provision for DCD, DSR and RI, totaling four possible sources). It is cleared by writing a 1 to the corresponding bit(s) in the Interrupt Clear Register (UARTICR).
- UARTRXINTR: receive interrupt. When the FIFO is enabled, this interrupt will be asserted when the FIFO reaches a programmable level. It will be cleared when the FIFO level drops bellow another programmable level (by reading the received data) or by writing a 1 to the corresponding bit(s) in the Interrupt Clear Register. If the FIFO is not enabled, both levels are one (that is, the interrupt will be assert when a byte is received and cleared when this byte is read - or the interrupt is cleared in the UARTICR).
- UARTTXINTR: transmit interrupt. When the FIFO is enabled, this interrupt will be asserted when the FIFO level is equal or lower a programmable level. It will be cleared when the FIFO level get above another programmable level (by transmitting the data in the FIFO) or by writing a 1 to the corresponding bit(s) in the Interrupt Clear Register. If the FIFO is not enabled, both levels are one (that is, the interrupt will be assert when the transmitter is free and cleared when the transmitter is busy - or the interrupt is cleared in the UARTICR).
- UARTRTINTR: receive timeout interrupt. This interrupt is assert when no data is received in 32 bit time and there is data in the FIFO. It is cleared when the FIFO is emptied (by reading the data) or by writing a 1 to the corresponding bit in the Interrupt Clear Register (UARTICR). This interrupt is normally used when the FIFO level for UARTRXINTR is greater than one, so data is not “forgotten” in the FIFO.

- UARTEINTR: error interrupt. Here we have the four errors we seem before: framing, parity, break detect and overrun. It can be cleared by writing to the relevant bits of the Interrupt Clear Register.

The programmable levels in the FIFO for the transmit and receive exists to reduce the number (and overhead) of interrupts.

Using the transmit interrupt requires a little logic:

- Start with the transmit interrupt disabled
- When you have something for transmission, first check if you can just put it in the UART FIFO. If so, there is no need for an interrupt. If the UART FIFO is full, you store the data in a FIFO of your own and enable the transmit interrupt.
- In the transmit interrupt, check if there is data in your FIFO. If so, move it to the UART FIFO and keep the interrupt enabled. If there is nothing to put in the FIFO, disable the transmit interrupt.

Pins Options

The RP2040 has a somewhat flexible mapping of pins for the serial interfaces (UART, SPI and I2C).

The options for UART0 are:

Function	GPIOs
Tx	0, 12, 16, 28
Rx	1, 13, 17, 29
CTS	2, 14, 18
RTS	3, 15, 19

The options for UART1 are:

Function	GPIOs
Tx	4, 8, 20, 24
Rx	5, 9, 21, 25
CTS	6, 10, 22, 26
RTS	7, 11, 23, 27

Selected SDK Functions

These functions are the library `hardware_uart`.

In this functions, uart should be `uart0` or `uart 1`.

```
uint uart_init (uart_inst_t *uart, uint baudrate)
```

Initialize a UART, baudrate is in bps. Must be called before the other functions. Returns the actual baudrate programmed.

```
uint uart_set_baudrate (uart_inst_t *uart, uint baudrate)
```

Change the baudrate of a UART. Returns the actual baudrate programmed.

```
static void uart_set_hw_flow (uart_inst_t *uart, bool cts, bool rts)
```

Turns on or off the hardware flow control options.

```
static void uart_set_format (uart_inst_t *uart, uint data_bits, uint
stop_bits, uart_parity_t parity)
```

Set the format of the data sent and received:

- `data_bits` must be between 5 and 8
- `stop_bits` must be 1 or 2
- `parity` must be one of the following: `UART_PARITY_NONE, UART_PARITY_EVEN, UART_-PARITY_ODD`

```
static void uart_set_irq_enables (uart_inst_t *uart, bool rx_has_data, bool
tx_needs_data)
```

Controls the use of the UART interrupts. If `rx_has_data`is true, enables the receive interrupt (there is data in the RX FIFO). If `tx_needs_data`is true, enables the transmit interrupt (the TX FIFO needs data).

Notice that there is no control (in the SDK) over the thresholds of the FIFO. Enabling the receive interrupt will also enable the receive timeout interrupt.

```
static void uart_set_fifo_enabled (uart_inst_t *uart, bool enabled)
```

Enables or disables the FIFOs in the UART. The RP2040 does not allow independent control over RX and TX FIFOs, you can have both or none.

```
static bool uart_is_readable (uart_inst_t *uart)
```

Return true if there is data in the receive FIFO.

```
bool uart_is_readable_within_us (uart_inst_t *uart, uint32_t us)
```

Wait at most us microseconds for data to be available in the receive FIFO. Return true if data is available or false if the time expired with no data available.

`static bool uart_is_writable (uart_inst_t *uart)`

Return true if there is space available in the TX FIFO.

`static void uart_tx_wait_blocking (uart_inst_t *uart)`

Blocks until the TX FIFO and the transmit shift register are empty.

`static void uart_putc_raw (uart_inst_t *uart, char c)`

Waits for space in the TX FIFO and puts a character in it.

Notes:

- Does not perform CR/LF conversion.
- The function return when the character is put in the FIFO, not when it is sent (this can take same time if there are more characters in the FIFO and/or hardware flow control is used).

`static void uart_putc (uart_inst_t *uart, char c)`

Waits for space in the TX FIFO and puts a character in it.

Notes:

- If CR/LF conversion is active (see `uart_set_translate_crlf`) and c is a line feed (0x0A), this function will put two characters in the TX FIFO, 0x0D (carriage return) and 0x0A. It will wait for space in the FIFO before putting each one.
- The function return when the character is put in the FIFO, not when it is sent (this can take same time if there are more characters in the FIFO and/or hardware flow control is used).

`static void uart_puts (uart_inst_t *uart, const char *s)`

Sends a null terminate string. The logic for each character in the string is similar to the one in `uart_putc`, except that if CR/LF conversion is active it will not insert a carriage return if the line feed is already preceded by a carriage return in the string. The ending null is not sent.

The function returns when the last character is put in the FIFO.

`static void uart_write_blocking (uart_inst_t *uart, const uint8_t *src, size_t len)`

Sends len characters starting form src. Does not perform CR/LF conversion. The function returns when the last character is put in the FIFO, blocking for space as necessary.

`static char uart_getc (uart_inst_t *uart)`

Read a character from the UART, will block until one is available in the RX FIFO.

`static void uart_read_blocking (uart_inst_t *uart, uint8_t *dst, size_t len)`

Reads `len` characters into `dst`, blocking as necessary for the characters to be received.

`static void uart_set_break (uart_inst_t *uart, bool en)`

Turns on or off the transmission of a break condition.

`void uart_set_translate_crlf (uart_inst_t *uart, bool translate)`

If `translate` is true, a line feed (0x0A) will be translate to carriage return (0x0D) + line feed in `uart_putc` and `uart_puts`.

`static uint uart_get_dreq (uart_inst_t *uart, bool is_tx)`

Return the DREQ (DMA Request) for transmitting (`is_tx = true`) or receiving (`is_tx = false`).

Using the UART Registers

The functions available in the SDK does not support all the functionality provided by the UARTs. If you want more control you will have to access the UART Registers.

The complete documentation of the registers available is in the RP2040 datasheet. Here I will just give a general idea of how this is done.

All registers are mapped into memory. The UART0 and UART1 registers start at base addresses of 0x40034000 and 0x40038000 respectively (defined as `UART0_BASE` and `UART1_BASE` in the SDK). Each register is at an offset of these base addresses.

The SDK defines routines, structures and constants that simplify accessing the UART registers, as exemplified bellow:

```
1  // Tests if Overrun Error is set in the Receive Status Register
2  bool uart_overrun_error(uart_inst_t *uart) {
3      // uart_get_hw(uart) returns the base address of the uart
4      // uart_get_hw(uart)->rsr returns the contents of the recieve status register
5      return !!(uart_get_hw(uart)->rsr & UART_UARTRSR_OE_BITS);
6  }
7
8  // Clear errors in  the Receive Status Register
9  void uart_clear_errors(uart_inst_t *uart) {
10     // uart_get_hw(uart) returns the base address
```

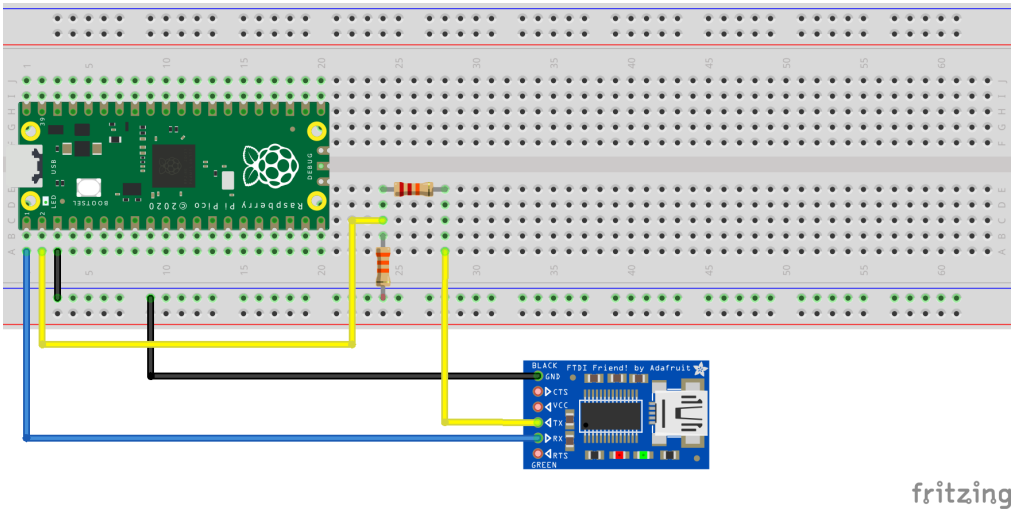
```
11      // uart_get_hw(uart)->rsr access the recieve status register
12      uart_get_hw(uart)->rsr = 0;     // doesn't matter what is written
13  }
14
15  // Set stick one parity
16  // (send and check parity bit as 1)
17  void uart_clear_overrun(uart_inst_t *uart) {
18      // uart_get_hw(uart) returns the base address
19      // &uart_get_hw(uart)->rsr returns the address of the recieve status register
20      // hw_write_masked(&uart_get_hw(uart)->lcr_h, data, mask) write data to
21      //   the bits selected by mask
22      hw_write_masked(&uart_get_hw(uart)->lcr_h,
23                      UART_UARTLCR_H_PEN_BITS |
24                      UART_UARTLCR_H_EPS_BITS |
25                      UART_UARTLCR_H_SPS_BITS,
26                      UART_UARTLCR_H_PEN_BITS |
27                      UART_UARTLCR_H_EPS_BITS |
28                      UART_UARTLCR_H_SPS_BITS);
29  }
```

The definition of the structures and constants can be found at `pico-sdk\src\rp2040`.

Example

To use this example you need to connect a Pi Pico to a PC.

As modern PCs do not have serial interfaces, you will need a “TTL” serial to USB adapter (sometimes called an “FDTI adapter”, FDTI is a company that makes serial to USB chips). Care must be taken in that the the RP2040 works at 3.3V and will be damaged if 5V (the standard voltage for TTL chips) is applied at any pin. Very few adapters have the option to use 3.3V in the Rx and Tx pin. You can use a resistive divisor to reduce to 3.3V the voltage inputed in the Rx pin in the Pi Pico:



Connecting a Serial to USB Adapter

You can also use a Pi Pico as a simple serial to USB converter, see the examples in the USB chapter.

On the software side in the PC, depending on the adapter and OS you may need a driver; this should be provided by the vendor of the adapter. You will also need a communication program that sends the characters you type and shows on the screen the received characters. A simple option is the Serial Monitor in the Arduino IDE. For Windows, PuTTY (availible at putty.org) is a popular option. For Linux, you may use minicom.

In this example the communication parameters are 300bps (very slow, so you can see the characters arriving), 8 data bits, no parity and 1 stop bit ("8N1").

The program will sum decimal numbers:

- The receive interrupt will be use to input decimal numbers, a carriage return (Enter) will signal the end of the number input. The digits will be echoed (sent back) in the interrupt. When Enter is received, the interrupt is disabled, so received characters will be stored in the FIFO while the sum is updated.
- In the main program we will wait for the input of a number, update the sum, transmit it and re-enable the receive interrupt.

UART Example

```c
/**
 * @file uartsum.c
 * @author Daniel Quadros
 * @brief Example of using the UART
 * @version 0.1
 * @date 2022-06-17
 *
 * @copyright Copyright (c) 2022, Daniel Quadros
 *
 */

#include "stdio.h"
#include "pico/stdlib.h"
#include "hardware/uart.h"
#include "hardware/irq.h"

// Select UART and Pins
#define UART_ID uart0
#define UART_TX_PIN   0
#define UART_RX_PIN   1

// UART Configuration
#define BAUD_RATE 300
#define DATA_BITS 8
#define STOP_BITS 1
#define PARITY    UART_PARITY_NONE

// UART interrupt requuest
int UART_IRQ;

// Current number and sum
volatile int number;
volatile bool number_received = false;
volatile int sum = 0;

// Rx interrupt handler
void on_uart_rx() {
    // There can be multiple chars in the FIFO
```

```
39      while (uart_is_readable(UART_ID)) {
40          uint8_t ch = uart_getc(UART_ID);
41
42          if (ch == 0x0D) {
43              // A number was entered
44              // disable interrupt and signal number received
45              irq_set_enabled(UART_IRQ, false);
46              number_received = true;
47              break;
48          } else if ((ch >= '0') && (ch <= '9')) {
49              // Update number, limit to 4 digits
50              number = (number*10  + ch - '0') % 10000;
51              // Echo the digit
52              if (uart_is_writable(UART_ID)) {
53                  uart_putc(UART_ID, ch);
54              }
55          }
56      }
57  }
58
59  // Main Program
60  int main() {
61      char msg[30];  // Buffer for sum message
62
63      // Set up UART
64      uart_init(UART_ID, BAUD_RATE);
65      uart_set_hw_flow(UART_ID, false, false);
66      uart_set_format(UART_ID, DATA_BITS, STOP_BITS, PARITY);
67      uart_set_fifo_enabled(UART_ID, true);
68
69      // Set the TX and RX pins
70      gpio_set_function(UART_TX_PIN, GPIO_FUNC_UART);
71      gpio_set_function(UART_RX_PIN, GPIO_FUNC_UART);
72
73      // Set up and enable receive interrupt
74      UART_IRQ = UART_ID == uart0 ? UART0_IRQ : UART1_IRQ;
75      irq_set_exclusive_handler(UART_IRQ, on_uart_rx);
76      irq_set_enabled(UART_IRQ, true);
77      uart_set_irq_enables(UART_ID, true, false);
```

```
78
79      // Main loop
80      while (1) {
81          if (number_received) {
82              // update sum
83              sum = (sum + number) % 1000000; // limit to 6 digits
84
85              // set up the sum message
86              sprintf (msg, " Sum:%d\r\n", sum);
87
88              // send sum
89              uart_puts(UART_ID, msg);
90
91              // wait for space in the Tx FIFO, so we can echo received chars
92              while (!uart_is_writable(UART_ID)) {
93              }
94
95              // get ready to receive another number
96              number = 0;
97              number_received = false;
98              irq_set_enabled(UART_IRQ, true);
99          }
100     }
101
102 }
```

Communication Using I²C

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

I²C Basics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

I²C Topology

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Electrical Interface

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Start and Stop Conditions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Read Operation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Write Operation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Combined Write/Read Operation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

I²C in the RP2040

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Generation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Pins Options

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Slave Library

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

I²C Scanner

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Using a 24C32 EEPROM

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

I²C Slave Device

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Communication Using SPI

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SPI Basics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SPI Signals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SPI Modes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

SPI in the RP2040

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Clock Generation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Pins Options

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Analog Input: the ADC

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Overview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Modes of Operation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Accuracy of the ADC

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Temperature Sensor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected SDK Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

A Brief Introduction to the USB Controller

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

USB Basics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Hardware

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Device Classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

TinyUSB

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Using the USB

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The HID Device Class

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example - Emulating a PC Keyboard

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example - Connecting a PC Keyboard to the Pi Pico

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Example - Serial USB Adapter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Appendix A - CMake Files for RP2040 Programs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Appendix B - Using stdio

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected pico_stdio Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected pico_stdio_uart Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Selected pico_stdio_usb Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

The printf Function

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Appendix C - Debugging Using the SWD Port

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Picoprobe Connections

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Software Installation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Debugging from the Command Line

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Debugging from inside Visual Code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Appendix D - Accessing the RP2040 Registers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Registers Addresses and Basic Access

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Special Write Operations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.

Using the SDK Functions for GPIO Output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/rp2040.