u32BR

Fernando Pereira

fernando@dcc.ufmg.br

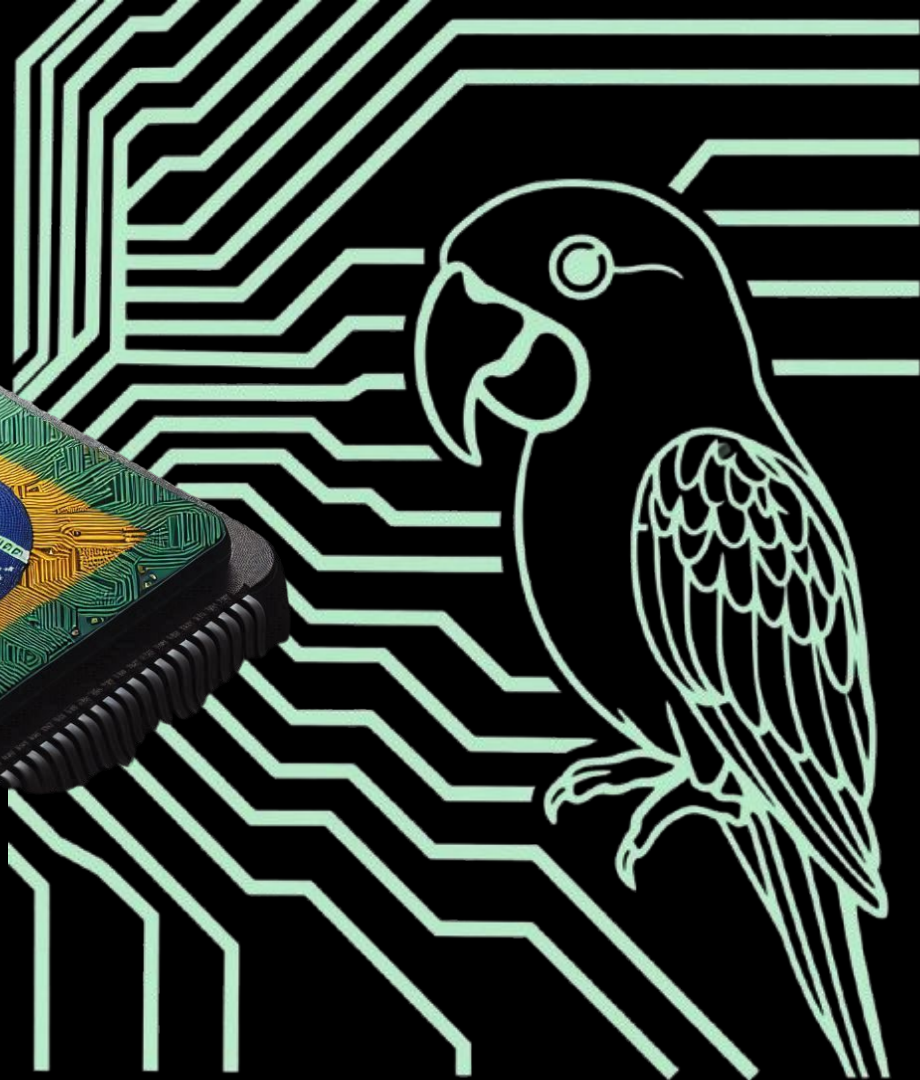
Softex

GOVERNO FEDERAL

BRASIL

UNIÃO E RECONSTRUÇÃO



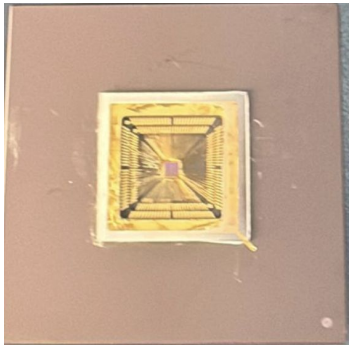


Este projeto é apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei 8.248, de 23 de outubro de 1991, conforme disposto no Art. 7.o da Portaria MCTI 5.275, de 5 de novembro de 2021.

Goal

Goal

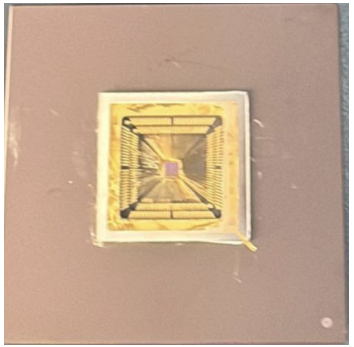
32-bit Microcontroller



Goal

32-bit Microcontroller

32KHz → ~200MHz

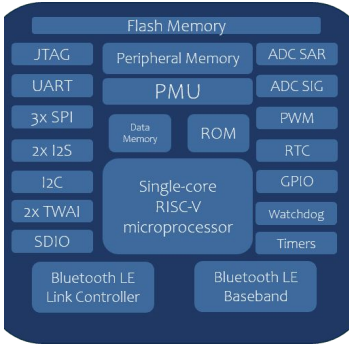


Goal

32-bit Microcontroller

32KHz → ~200MHz

Timer/PWM, 3x SPI, I2C, UART, GPIO, etc.



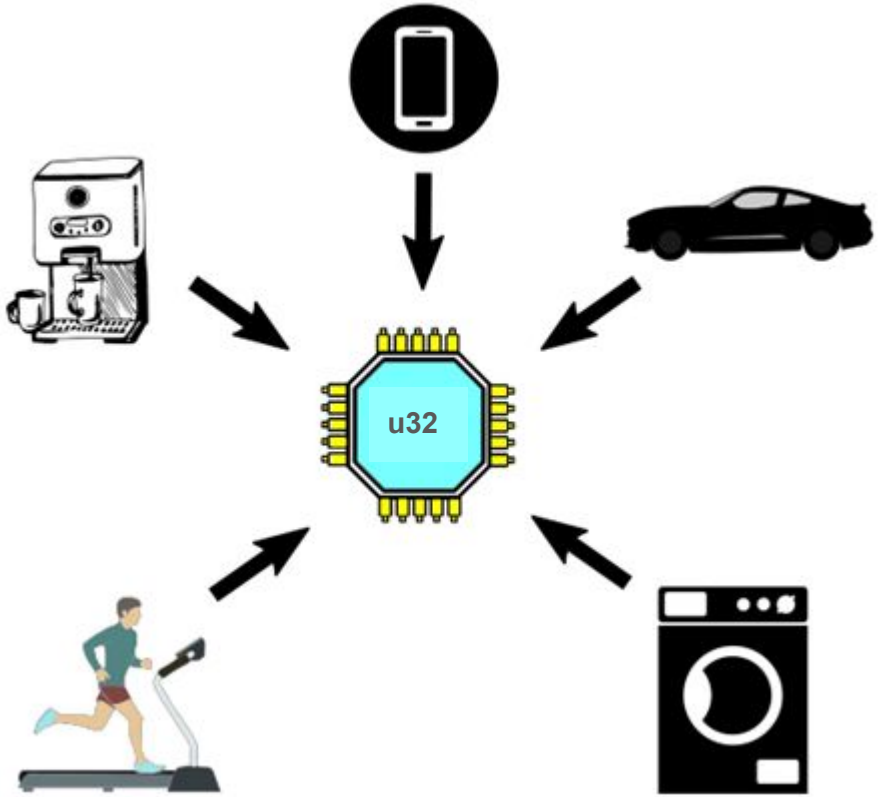
Goal

32-bit Microcontroller

32KHz → ~200MHz

Timer/PWM, 3x SPI, I2C, UART, etc.

Integrated bluetooth antenna (SiP)



Goal

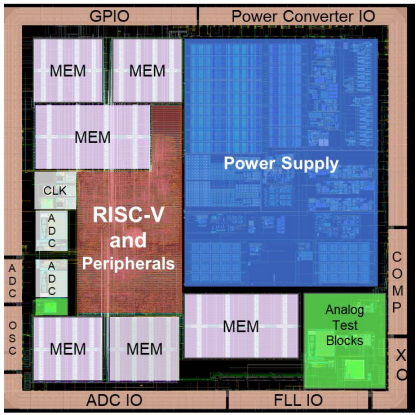
32-bit Microcontroller

32KHz → ~200MHz

Timer/PWM, 3x SPI, I2C, UART, etc.

Integrated bluetooth antenna (SiP)

RISC-V



Goal

32-bit Microcontroller

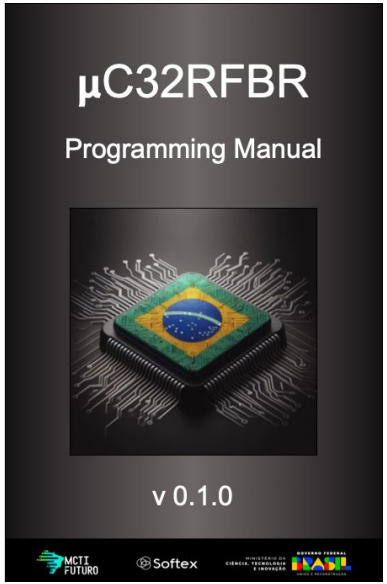
32KHz → ~200MHz

Timer/PWM, 3x SPI, I2C, UART, GPIO, etc.

Integrated bluetooth antenna (SiP)

RISC-V

Software Development Kit (SDK)



Goal

32-bit Microcontroller

32KHz → ~200MHz

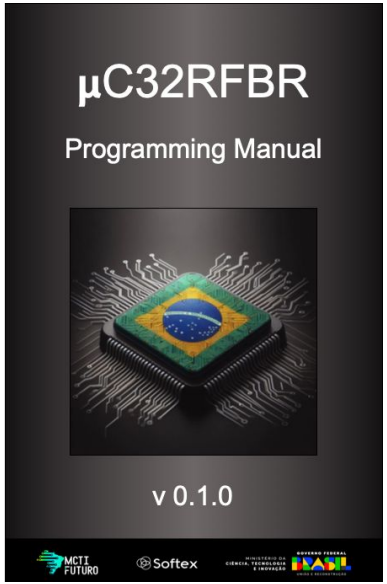
Timer/PWM, 3x SPI, I2C, UART, GPIO, etc.

Integrated bluetooth antenna (SiP)

RISC-V

Software Development Kit (SDK)





What?

30-month Project

R$ ~10M

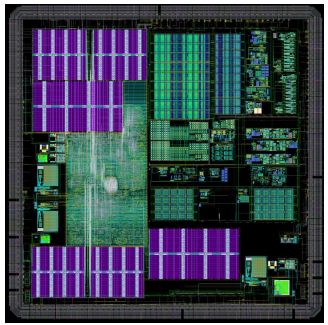


What?

30-month Project

R$ ~10M

Design:



Software:

u32BR SDK

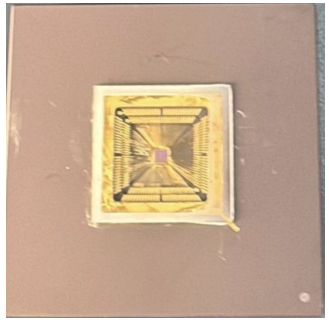
Generated by
Doxygen 1.8.5

250 pages

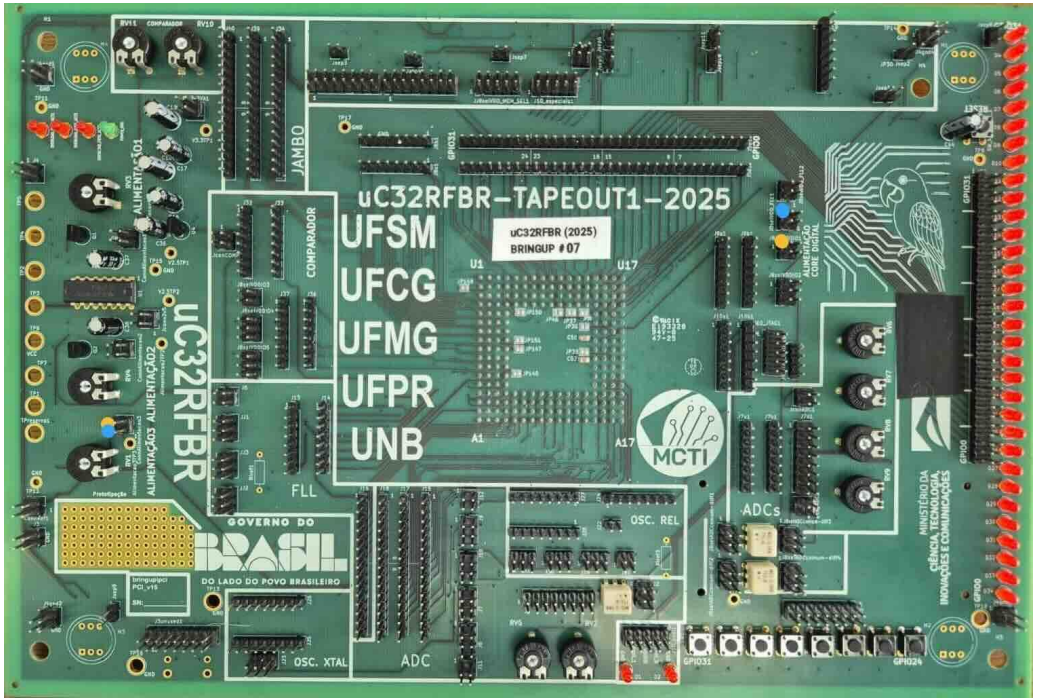
Fri May 15 2026

doxygen

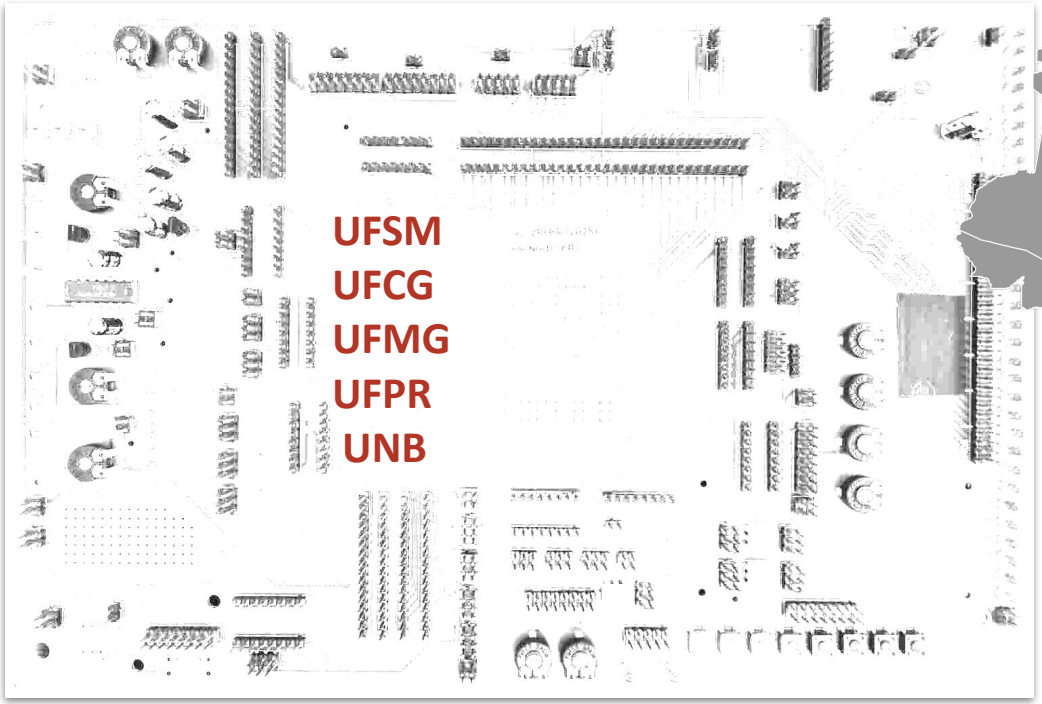
Prototype:

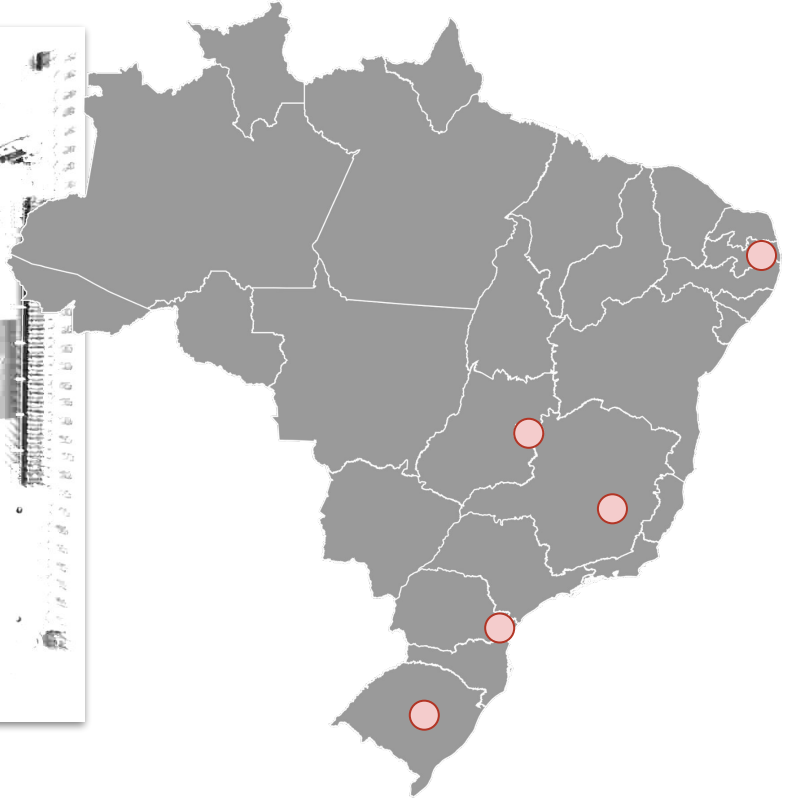


National Effort

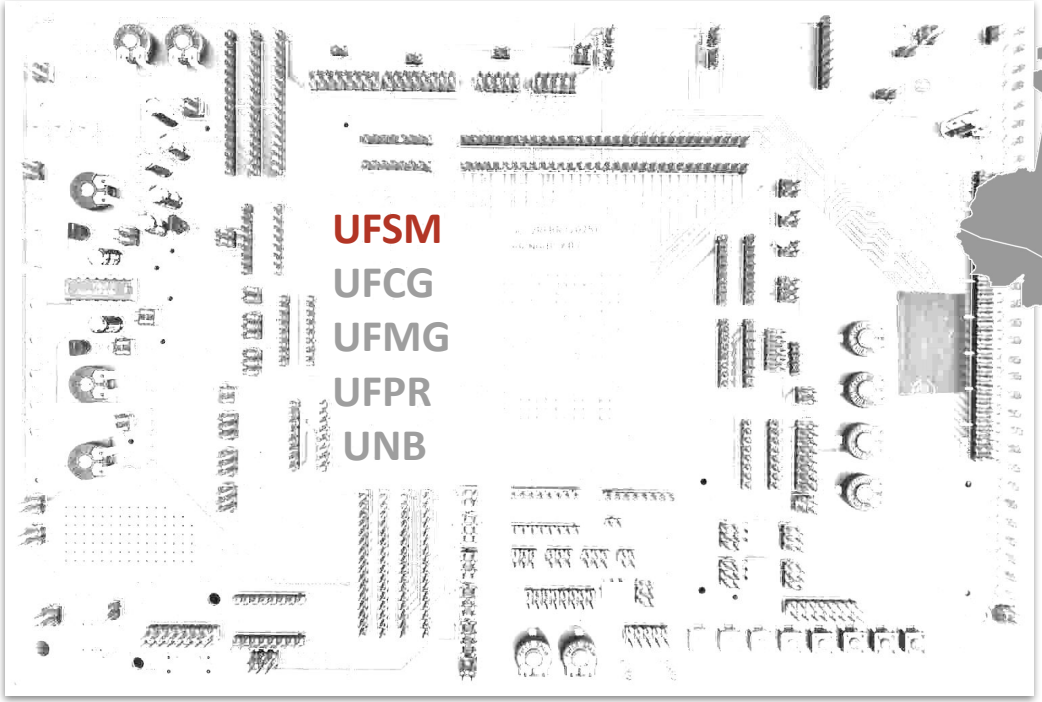


National Effort





National Effort





Universidade Federal de Santa Maria

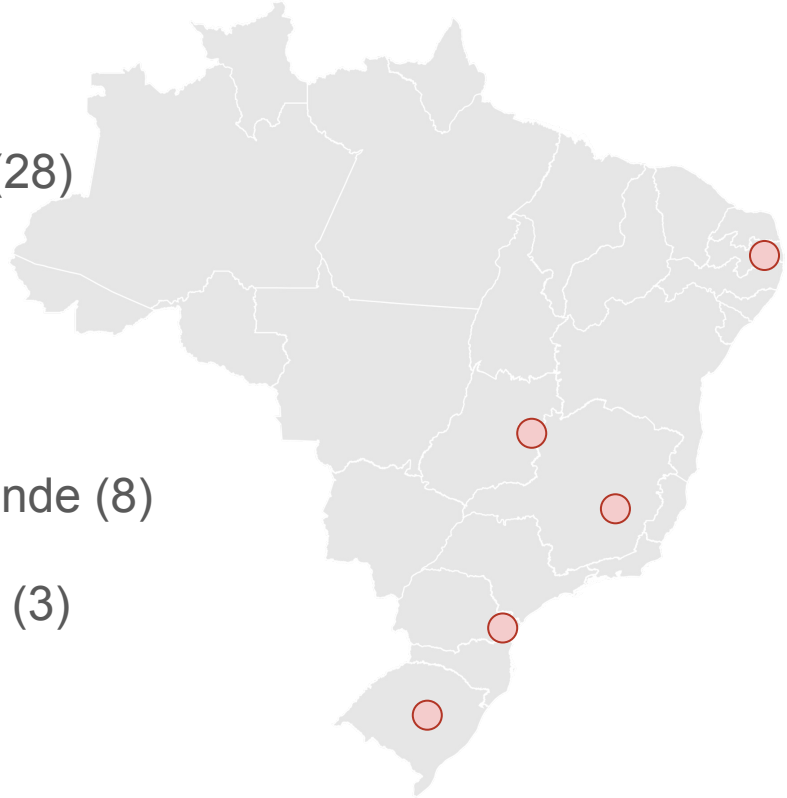
SEDES SAPIENTIÆ

1960

National Effort

Academic Partners

- Universidade Federal de Santa Maria (28)
- Universidade Federal do Paraná (6)
- Universidade Federal de Brasília (9)
- Universidade Federal de Campina Grande (8)
- Universidade Federal de Minas Gerais (3)



National Effort

Academic Partners

- Universidade Federal de Santa Maria (28)
- Universidade Federal do Paraná (6)
- Universidade Federal de Brasília (9)
- Universidade Federal de Campina Grande (8)
- Universidade Federal de Minas Gerais (3)

National Effort

Academic Partners

Universidade Federal de Santa Maria (28)

Universidade Federal do Paraná (6)

Universidade Federal de Brasília (9)

Universidade Federal de Campina Grande (8)

Universidade Federal de Minas Gerais (3)

Softex

Industry Partners

- Exatron
- Intral
- Novus
- FKS
- TecnoFlex

Why?

"We already have so many microcontrollers..."

Why?

"Between 2020 and 2023, there was a worldwide chip shortage affecting more than 169 industries."



https://en.wikipedia.org/wiki/2020%E2%80%932023_global_chip_shortage

"~~We already have so many microcontrollers...~~"

Why?

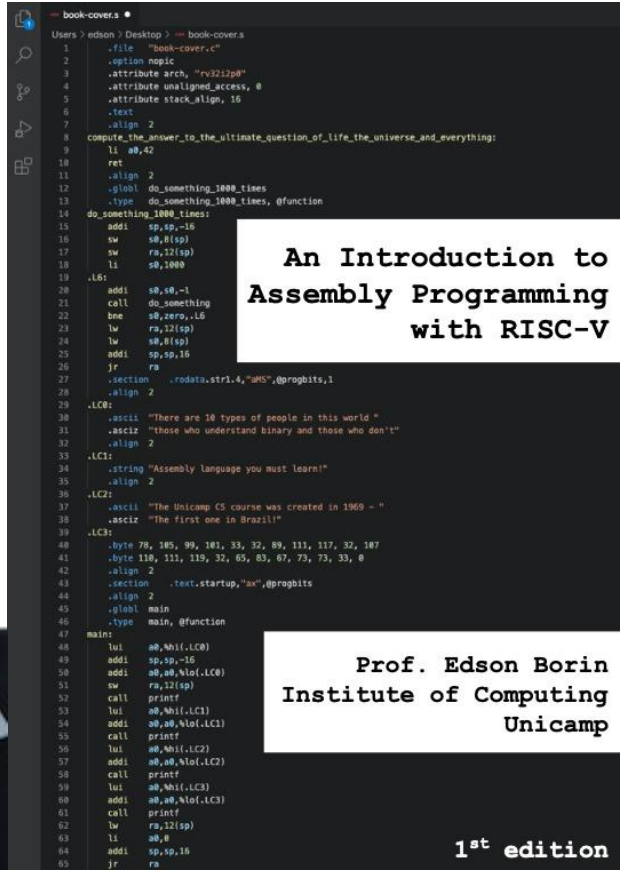
Know how

Why?

Know how

Prof. Edson Borin, UNICAMP





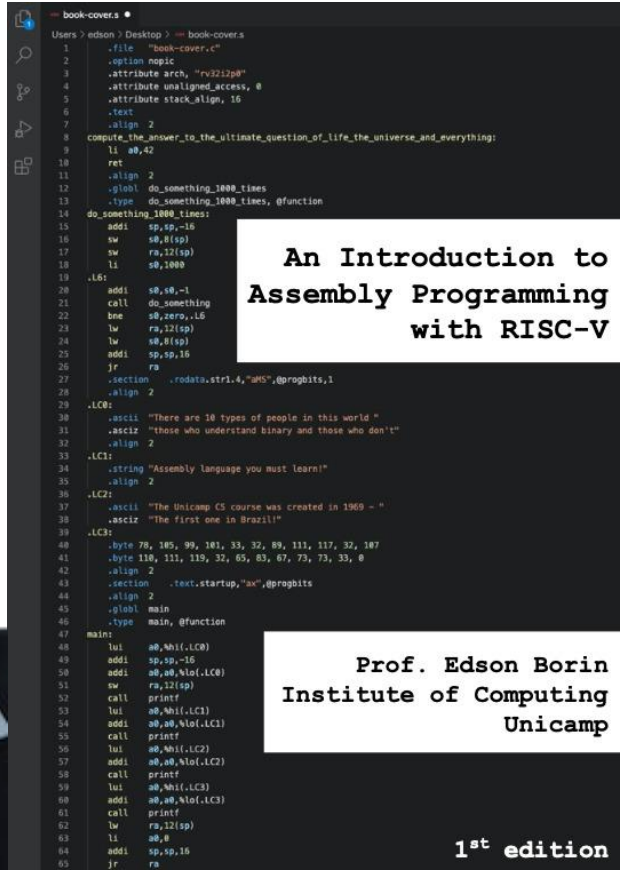
Why?

Know how

Prof. Edson Borin, UNICAMP

The Ministry of Education recognizes 662 courses of Computer Engineering in Brazil





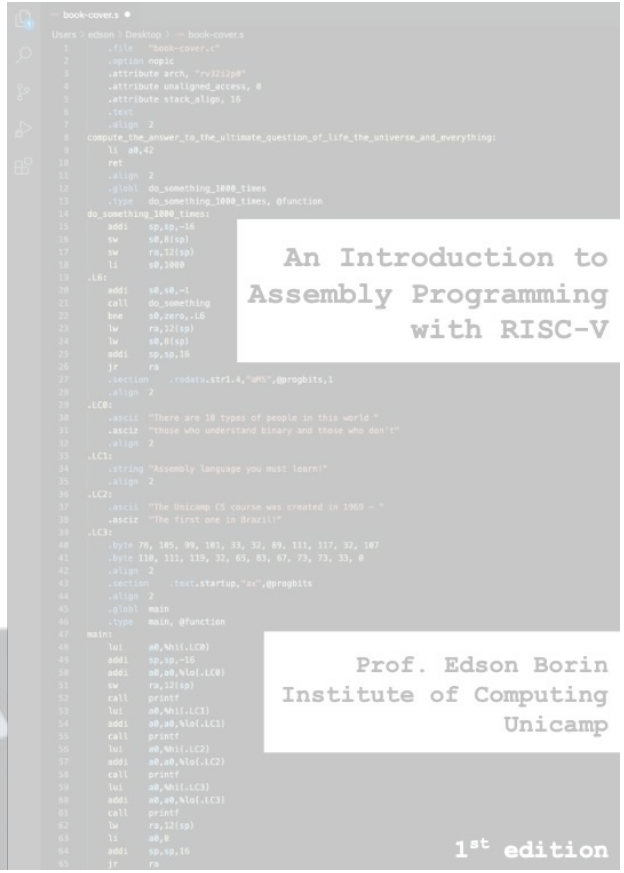
Why?

Know how

Prof. Edson Borin, UNICAMP

The Ministry of Education recognizes 662 courses of Computer Engineering in Brazil





GDP: R$ 12.7 T

Population: 213 M

Why?

Know how

Trust

Why?

Know how

Trust

> *"Since the design of modern microprocessors is usually kept as a trade secret, there is no efficient method for the user to verify that a single multiplication bug does not exist."*
>
> David Wagner, CRYPTO 2008, Advances in Cryptology

Why?

Know how

Trust

ZDNET tomorrow belongs to those who embrace it today

Home / Tech / Security

Researchers find backdoor on ZTE Android phones

Security researchers have reported a hidden backdoor in the ZTE Score M and Skate Android phones, which could give a hacker root access



Written by **Michael Lee,** Contributor

May 15, 2012 at 2:45 a.m. PT

https://www.zdnet.com/article/researchers-find-backdoor-on-zte-android-phones/

Why?

Know how

Trust

- The Shenzhen 2018 Supermicro Incident
- The Tailored Access Operations Unit (2013)
- The 2012 ZTE Phones (confirmed by D. Alperovitch)
- The Cisco 2008 FBI Report
- Jonathan Brossard Rakshasa Backdoor (2011)
- ...

Why?

Know how

Trust



Why?

Know how

Trust





Why?

Know how

Trust

Partnerships

UFSM



UnB



etc, etc...

Why?

Know how

Trust

Partnerships



	Professors	Engineers	Students
Digital	7	9	6
Analog	5	9	8
Radio	3	3	0
Software	2	0	1

The Software Stack



The Software Development Kit (SDK)

Tools

Libraries

Documentation

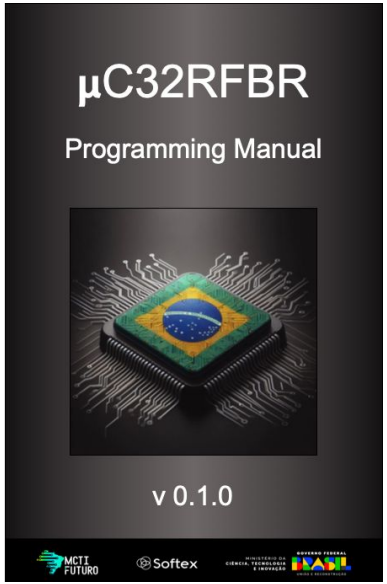
u32BR SDK

Generated by
Doxygen 1.8.5

250 pages

Fri May 15 2026

doxygen



The Software Development Kit (SDK)

Tools

Libraries

Documentation

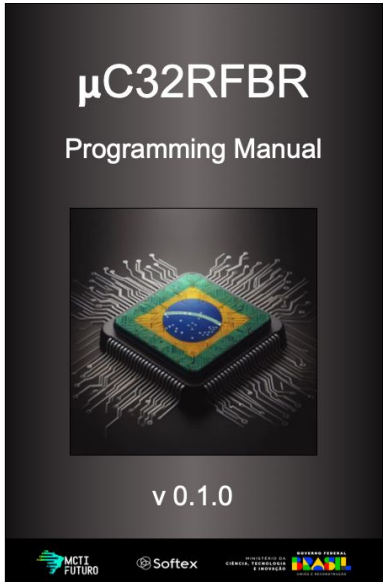
u32BR SDK

Generated by
Doxygen 1.8.5

250 pages

Fri May 15 2026

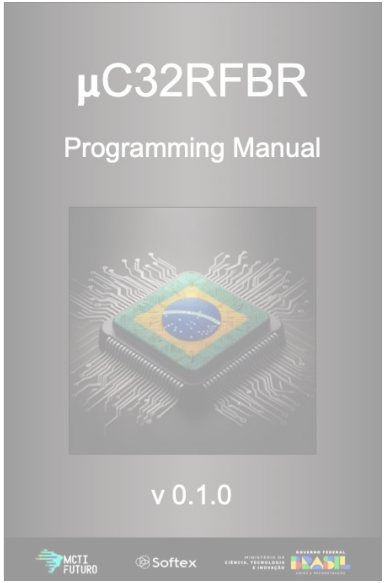
doxygen



What is this good for?

Workflow using the SDK (for u32BR clients)

1. Setup the development environment



Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE (Integrated Development Environment)

Workflow using the SDK (for u32BR clients)

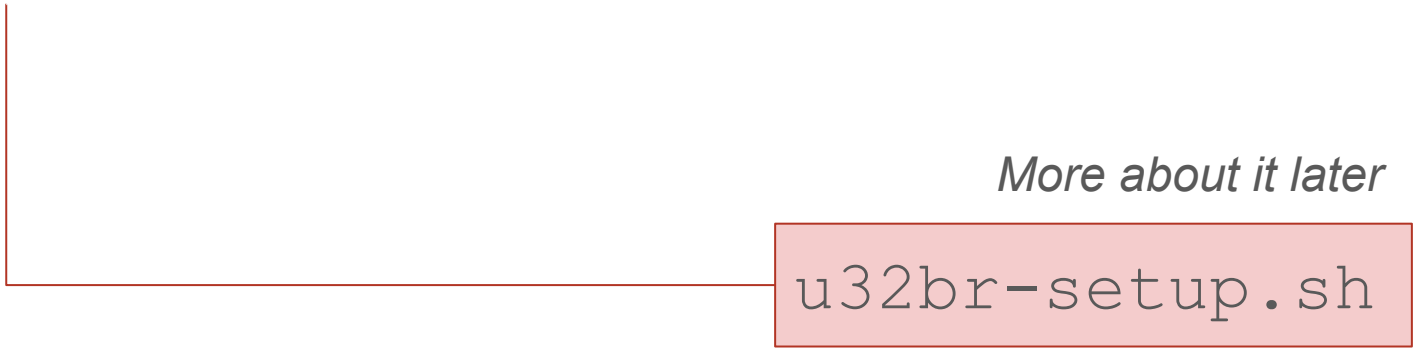
1. Setup the development environment
   a. Install SDK
   b. Install IDE (Integrated Development Environment)
2. Create a new project

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE (Integrated Development Environment)
2. Create a new project
   a. Use a project template from the SDK
   b. Configure settings (frequency, peripherals, etc)

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE (Integrated Development Environment)
2. Create a new project
   a. Use a project template from the SDK
   b. Configure settings (frequency, peripherals, etc)



Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE (Integrated Development Environment)
2. Create a new project
   a. Use a project template from the SDK
   b. Configure settings (frequency, peripherals, etc)
3. Develop code

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE
2. Create a new project
   a. Use a project template from the SDK
   b. Configure settings
3. Develop code
   a. Use u32BR libraries
   b. Use test cases as examples

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
    a. Install SDK
    b. Install IDE
2. Create a new project
    a. Use a project template from the SDK
    b. Configure settings
3. Develop code
    a. Use u32BR libraries
    b. Use test cases as examples
4. Build the project
    a. Use provided makefiles

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE
2. Create a new project
   a. Use a project template from the SDK
   b. Configure settings
3. Develop code
   a. Use u32BR libraries
   b. Use test cases as examples
4. Build the project
   a. Use provided makefiles
5. Debug and Test

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE
2. Create a new project
   a. Use a project template from the SDK
   b. Configure settings
3. Develop code
   a. Use u32BR libraries
   b. Use test cases as examples
4. Build the project
   a. Use provided makefiles
5. Debug and Test
   a. Load binary via bootloader
   b. Set breakpoints and step through
   c. Inspect variables

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE
2. Create a new project
   a. Use a project template
   b. Configure settings
3. Develop code
   a. Use u32BR libraries
   b. Use test cases as examples
4. Build the project
   a. Use provided makefiles
5. Debug and Test
   a. Load binary via bootloader
   b. Set breakpoints and step through
   c. Inspect variables
6. Optimize
   a. Use profiling tools
   b. Play with compiler options

Workflow using the SDK (for u32BR clients)

1. Setup the development environment
   a. Install SDK
   b. Install IDE
2. Create a new project
   a. Use a project template
   b. Configure settings
3. Develop code
   a. Use u32BR libraries
   b. Use test cases as examples
4. Build the project
   a. Use provided makefiles
5. Debug and Test
   a. Load binary via bootloader
   b. Set breakpoints and step through
   c. Inspect variables
6. Optimize
   a. Use profiling tools
   b. Play with compiler options
7. **Deploy**
   a. **Simulate**
   b. **Test on the microcontroller**

Reference Implementation

Reference Implementation

Pulp Platform (Parallel Ultra-Low Power)

https://github.com/pulp-platform/pulp-sdk

Reference Implementation

Pulp Platform

https://github.com/pulp-platform/pulp-sdk

Integrated Systems Laboratory (*ETH Zurich*)

Energy-Efficient Embedded Systems (*University of Bologna*)

Bruschi, Nazareno and Haugou, Germain and Tagliavini, Giuseppe and Conti, Francesco and Benini, Luca and Rossi, Davide **GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors.** ICCD 2021

Reference Implementation

Pulp Platform

https://github.com/pulp-platform/pulp-sdk

Apache 2.0 License

Reference Implementation

Pulp Platform

- Toolchain (GCC-based)
- Runtime Libraries
- RTOS Support (FreeRTOS)
- Examples and Templates

Reference Implementation

Pulp Platform

- Toolchain (GCC-based)
  - Compiler (gcc)
  - Assembler
  - Linker
- ~~Runtime Libraries~~
- ~~RTOS Support (FreeRTOS)~~
- ~~Examples and Templates~~

Reference Implementation

Pulp Platform

- ~~Toolchain (GCC based)~~ gcc 7 (custom) → gcc 15 (general)
- ~~Runtime Libraries~~
- ~~RTOS Support (FreeRTOS)~~
- ~~Examples and Templates~~

- Compiler (gcc)
- Assembler
- Linker

Reference Implementation

Pulp Platform

- ~~Toolchain (GCC-based)~~
- ~~Runtime Libraries~~
- ~~RTOS Support (FreeRTOS)~~
- ~~Examples and Templates~~

Removed lots of unnecessary code

Modified lots of old code

Added lots of new code

Reference Implementation

Runtime Libraries

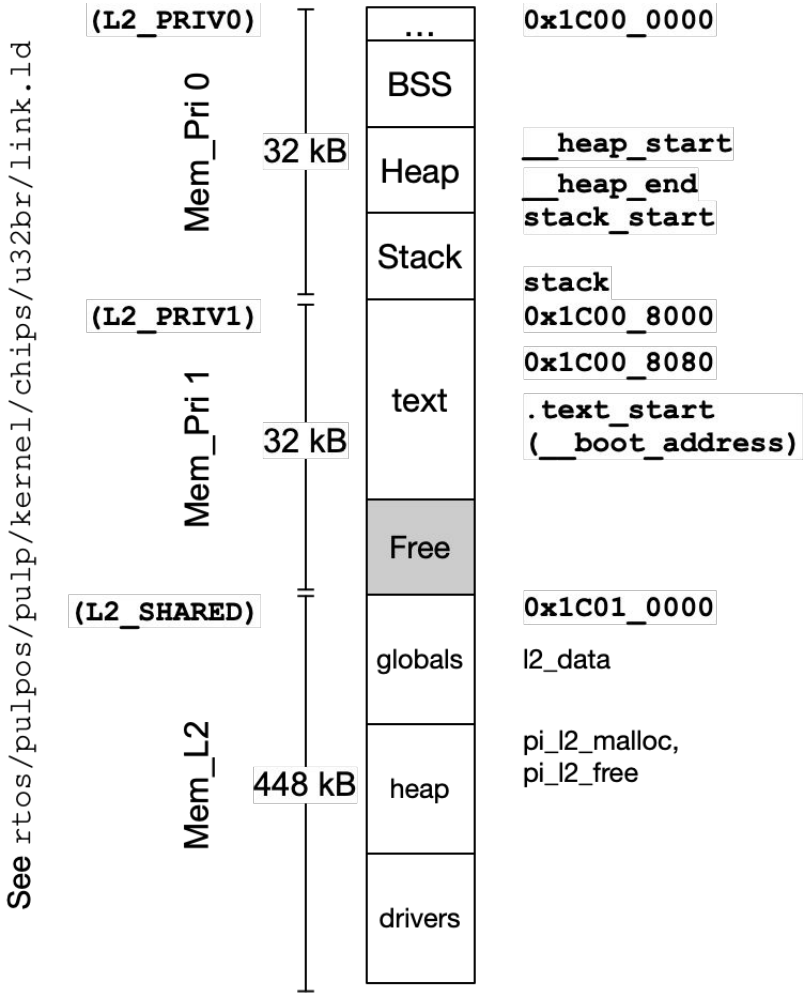
→ Memory allocation system

Reference Implementation

Runtime Libraries

→ Memory allocation system

malloc/free



Simulation

- Xcelium
- FPGA

Simulation

- Xcelium → UFMG/CEFET-MG
- FPGA → UFSM

Simulation

- Xcelium → UFMG/CEFET-MG
- FPGA → UFMS

- RTL Simulation
- Signal verification
- Coverage analysis

Simulation

- Xcelium → UFMG/CEFET-MG
- FPGA → UFSM



- Artix-7 FPGA (Xilinx)
- Nexys 4DDR Board (Digilent)

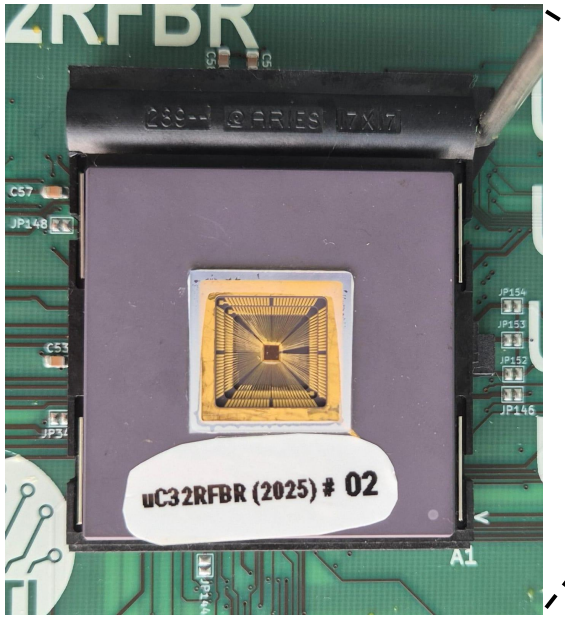
Actual Hardware

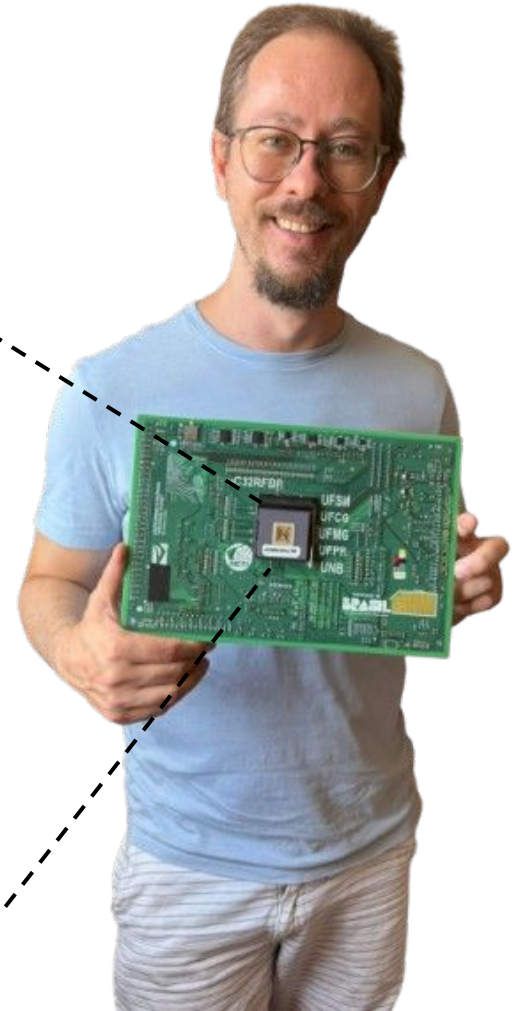
- Xcelium → UFMG/CEFET-MG
- FPGA → UFSM
- Development board



Actual Hardware

- Xcelium
- FPGA
- Development board





u32br-setup

u32br-setup

```
Usage: ../u32br-setup.sh <options>

Options:
  -a|--app=[name]                    Use an application based on its name.
  -c|--clean                         Clean the project before building.
  -d|--debug=[none|rtl|openocd|gdb]  Debug none (default), rtl, openocd or gdb.
  -g|--gui                           Use GUI interface.
  -h|--help                          Show this usage.
  -o|--output=[stdout|uart]          Output mode to stdout (default) or uart.
  -p|--platform=[rtl|fpga|board]     Use rtl (default), fpga, or board as execution platform.
  -r|--run=[yes|no]                  Run the project yes (default) or no on the platform.
  -s|--setup=<all|projects>          Setup projects, all or a selection of the following
                                       dependencies,toolchain,openocd,pulpissimo,sdk
  -t|--toolchain=[prebuilt|compile]  Use toolchain prebuilt or compile it from source.
  -v|--verbose                       Verbose mode
```

u32br-setup

```
Usage: ../u32br-setup.sh <options>

Options:
  -a|--app=[name]                  Use an application based on its name.
  -c|--clean                       Clean the project before building.
  -d|--debug=[none|rtl|openocd|gdb] Debug none (default), rtl, openocd or gdb.
  -g|--gui                         Use GUI interface.
  -h|--help                        Show this usage.
  -o|--output=[stdout|uart]        Output mode to stdout (default) or uart.
  -p|--platform=[rtl|fpga|board]   Use rtl (default), fpga, or board as execution platform.
  -r|--run=[yes|no]                Run the project yes (default) or no on the platform.
  -s|--setup=<all|projects>        Setup projects, all or a selection of the following
                                     dependencies,toolchain,openocd,pulpissimo,sdk
  -t|--toolchain=[prebuilt|compile] Use toolchain prebuilt or compile it from source.
  -v|--verbose                     Verbose mode
```

u32br-setup

```
Usage: ../u32br-setup.sh <options>

Options:
  -a|--app=[name]                    Use an application based on its name.
  -c|--clean                         Clean the project before building.
  -d|--debug=[none|rtl|openocd|gdb]  Debug none (default), rtl, openocd or gdb.
  -g|--gui                           Use GUI interface.
  -h|--help                          Show this usage.
  -o|--output=[stdout|uart]          Output mode to stdout (default) or uart.
  -p|--platform=[rtl|fpga|board]     Use rtl (default), fpga, or board as execution platform.
  -r|--run=[yes|no]                  Run the project yes (default) or no on the platform.
  -s|--setup=<all|projects>          Setup projects, all or a selection of the following
                                       dependencies,toolchain,openocd,pulpissimo,sdk
  -t|--toolchain=[prebuilt|compile]  Use toolchain prebuilt or compile it from source.
  -v|--verbose                       Verbose mode
```

u32br-setup

```
Usage: ../u32br-setup.sh <options>

Options:
  -a|--app=[name]                    Use an application based on its name.
  -c|--clean                         Clean the project before building.
  -d|--debug=[none|rtl|openocd|gdb]  Debug none (default), rtl, openocd or gdb.
  -g|--gui                           Use GUI interface.
  -h|--help                          Show this usage.
  -o|--output=[stdout|uart]          Output mode to stdout (default) or uart.
  -p|--platform=[rtl|fpga|board]     Use rtl (default), fpga, or board as execution platform.
  -r|--run=[yes|no]                  Run the project yes (default) or no on the platform.
  -s|--setup=<all|projects>          Setup projects, all or a selection of the following
                                       dependencies,toolchain,openocd,pulpissimo,sdk
  -t|--toolchain=[prebuilt|compile]  Use toolchain prebuilt or compile it from source.
  -v|--verbose                       Verbose mode
```

u32br-setup

```
Usage: ../u32br-setup.sh <options>

Options:
  -a|--app=[name]                    Use an application based on its name.
  -c|--clean                         Clean the project before building.
  -d|--debug=[none|rtl|openocd|gdb]  Debug none (default), rtl, openocd or gdb.
  -g|--gui                           Use GUI interface.
  -h|--help                          Show this usage.
  -o|--output=[stdout|uart]          Output mode to stdout (default) or uart.
  -p|--platform=[rtl|fpga|board]     Use rtl (default), fpga, or board as execution platform.
  -r|--run=[yes|no]                  Run the project yes (default) or no on the platform.
  -s|--setup=<all|projects>          Setup projects, all or a selection of the following
                                       dependencies,toolchain,openocd,pulpissimo,sdk
  -t|--toolchain=[prebuilt|compile]  Use toolchain prebuilt or compile it from source.
  -v|--verbose                       Verbose mode
```

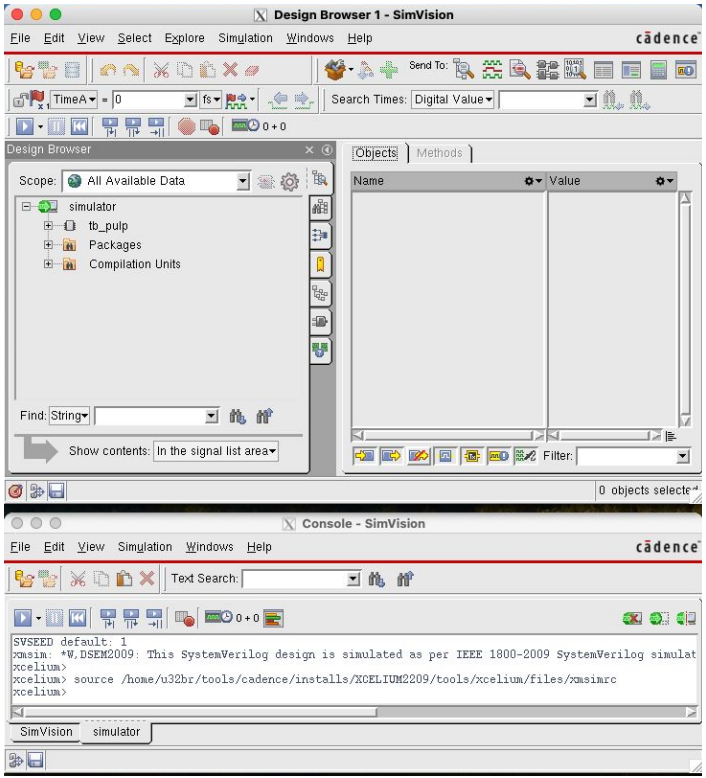
u32br-setup - Example

```
$ ./u32br-setup.sh -v --app=uart/loopback
...
```

u32br-setup - Example

```
$ ./u32br-setup.sh -v --app=uart/loopback
...
```

```
$ ./u32br-setup.sh -v \
--app=uart/loopback --gui
```



Development Flow

Development Flow - Example GPIO

UFSM

Raw test

```
...
// gpio set mask followed by a clear mask
gpio_set_mask(0x555555FF);
gpio_clear_mask(0x11111100);

// negate all GPIOs
printf("main(): test gpio_write()\n");
gpio_values = ~(archi_read32(GPIO_ADDR_PAD_DOUT));
for (int i=0; i<GPIO_TOTAL; i++)
  gpio_write(i, (gpio_values >> i) & 0x01);

// print current GPIO input value one by one
printf("main(): test gpio_read()\n");
for (int i=0; i<GPIO_TOTAL; i++)
  printf("main(): gpio_read(%d)=%d\n", i, gpio_read(i));
...
```



Development Flow - Example GPIO

```
static void test_entry() {
  int status = 0;
  struct pi_device gpio;

  if (pi_gpio_open(&gpio)) {
     status = 1;
     pmsis_exit(status);
  }

  for (pi_gpio_e pin = PI_GPIO_0; pin <= PI_GPIO_31; pin += 2)
     status |= test_pin(gpio, pin);

  for (pi_gpio_e pin = PI_GPIO_1; pin <= PI_GPIO_31; pin += 2)
     status |= test_pin(gpio, pin);

  if (!pi_gpio_mask_set(0x555555FF))
     status |= 1;

  pmsis_exit(status);

}
```

UFMG

Unit test



Development Flow - Example GPIO

```
static void test_entry() {
  int status = 0;
  struct pi_device gpio;

  if (pi_gpio_open(&gpio)) {
    status = 1;
    pmsis_exit(status);
  }

  for (pi_gpio_e pin = PI_GPIO_0; pin <= PI_GPIO_31; pin += 2)
    status |= test_pin(gpio, pin);

  for (pi_gpio_e pin = PI_GPIO_1; pin <= PI_GPIO_31; pin += 2)
    status |= test_pin(gpio, pin);

  if (!pi_gpio_mask_set(0x555555FF))
    status |= 1;

  pmsis_exit(status)

}
```

UFMG

Unit test



Development Flow - Example GPIO

```
pi gpio open(&gpio)
```

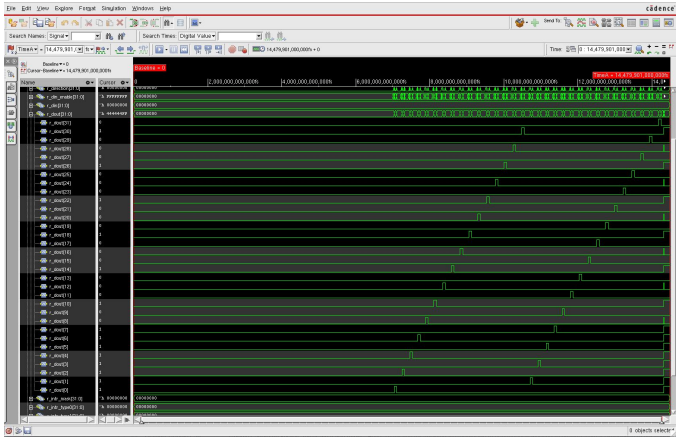
```
int pi_gpio_open(struct pi_device *device) {
  int irq = hal_irq_disable();

  pi_gpio_t *gpio = &pos_gpio;
  device->data = (void *)gpio;

  hal_irq_restore(irq);

  return 0;
}
```

UFMG

Implementation



Development Flow - Example GPIO



```
int pi_gpio_open(struct pi_device *device) {
  int irq = hal_irq_disable();

  pi_gpio_t *gpio = &pos_gpio;
  device->data = (void *)gpio;

  hal_irq_restore(irq);

  return 0;
}
```

UFMG

Validated on Xcelium



Development Flow - Example GPIO

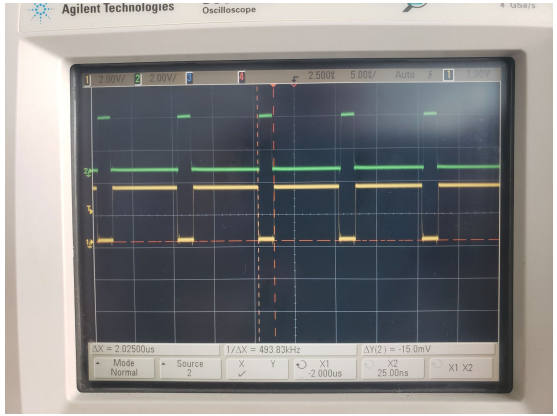
```
/**
 * @brief Opens a GPIO port.
 *
 * This function initializes and opens a GPIO port specified
 * by the input parameter.
 * It returns an integer indicating the status of the operation.
 *
 * @param[in] gpio A pointer to a `struct pi_device`
 * representing the GPIO device to be opened.
 * @return int Returns 0 if the GPIO port is successfully
 * opened, otherwise returns a non-zero error code.
 *
 * Example usage:
 * @code
 * struct pi_device gpio_device;
 * int status = pi_gpio_open(&gpio_device);
 * if (status != 0) {
 *   // Error handling
 * }
 * @endcode
 */
int pi_gpio_open(struct pi_device* gpio);
```

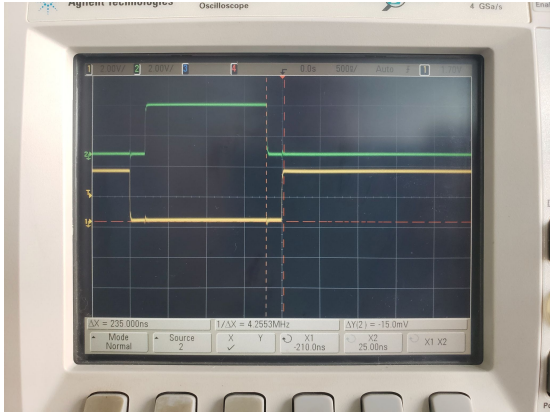
UFMG

Documentation



Development Flow - Example GPIO





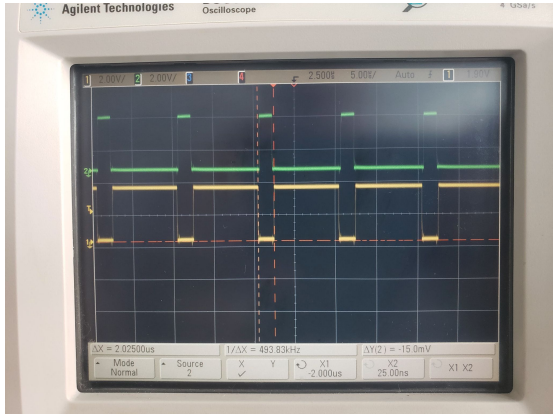


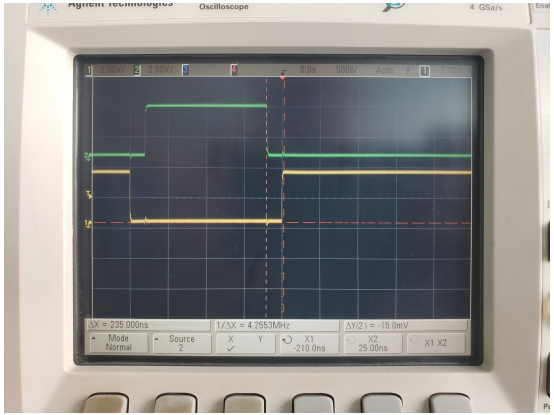
UFSM

Deployment on Artix 7



Development Flow - Example GPIO









UFSM

~~Deployment on Artix 7~~

Actual Hardware!



GPIO (Doxygen)

u32BR SDK

Generated by
Doxygen 1.8.5

250 pages

Fri May 15 2026

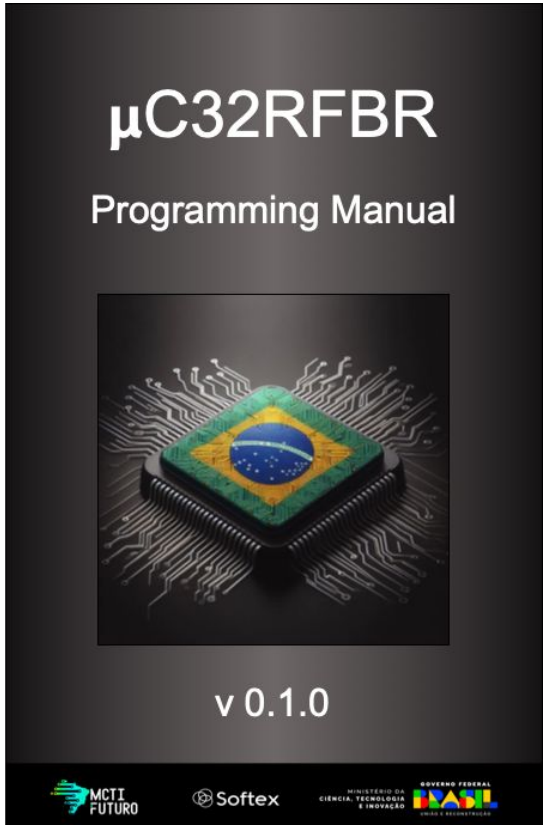
doxygen

UFMG

Programming manual



GPIO (Programmer Manual)



u32BR SDK

Generated by Doxygen 1.8.5

250 pages

Fri May 15 2026

doxygen

UFMG

Programming manual



Drivers

```
% cd ~/u32br-sdk/rtos/pmsis/pmsis_api/include/pmsis/drivers
% ls
adc.h          gpio.h         perf.h         spi.h
adcsd.h        i2c.h          pmu.h          timer.h
can.h          i2s.h          rtc.h          uart.h
fdcan.h        pad.h          sdio.h         watchdog.h
```

Drivers

```
% cd ~/u32br-sdk/rtos/pmsis/pmsis_api/include/pmsis/drivers
% ls
adc.h          gpio.h         perf.h         spi.h
adcsd.h        i2c.h          pmu.h          timer.h
can.h          i2s.h          rtc.h          uart.h
fdcan.h        pad.h          sdio.h         watchdog.h
```



Industrial Edge Gateway

Drivers

```
% cd ~/u32br-sdk/rtos/pmsis/pmsis_api/include/pmsis/drivers
% ls
adc.h          gpio.h         perf.h         spi.h
adcsd.h        i2c.h          pmu.h          timer.h
can.h          i2s.h          rtc.h          uart.h
fdcan.h        pad.h          sdio.h         watchdog.h
```



Industrial Edge Gateway



Automotive Control Unit

Drivers

```
% cd ~/u32br-sdk/rtos/pmsis/pmsis_api/include/pmsis/drivers
% ls
adc.h          gpio.h         perf.h         spi.h
adcsd.h        i2c.h          pmu.h          timer.h
can.h          i2s.h          rtc.h          uart.h
fdcan.h        pad.h          sdio.h         watchdog.h
```



Industrial Edge Gateway



Automotive Control Unit



Wearable Device

Drivers

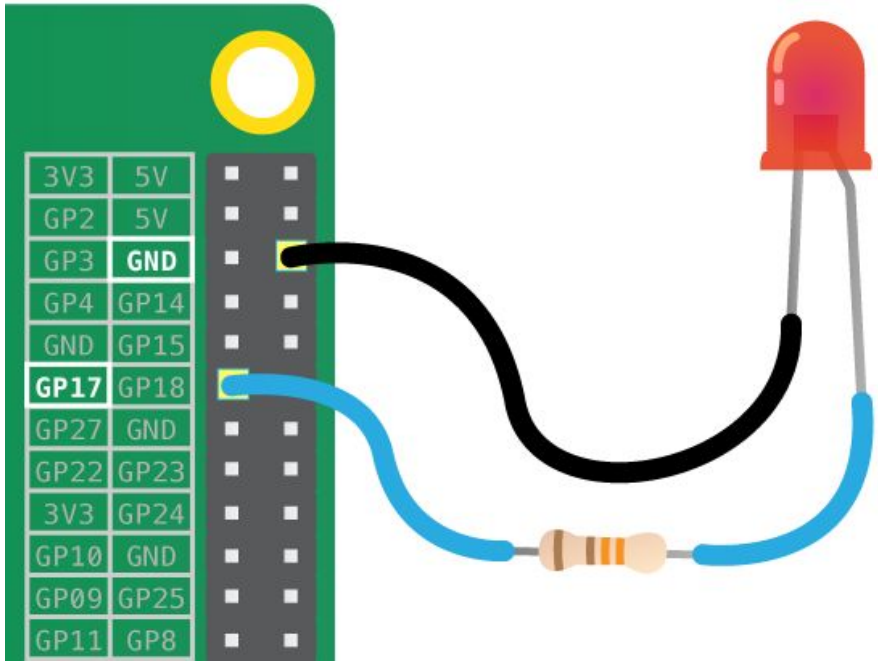
```
% cd ~/u32br-sdk/rtos/pmsis/pmsis_api/include/pmsis/drivers
% ls
adc.h          gpio.h         perf.h         spi.h
adcsd.h        i2c.h          pmu.h          timer.h
can.h          i2s.h          rtc.h          uart.h
fdcan.h        pad.h          sdio.h         watchdog.h
```

Testes

```
~/Programs/u32br-sdk/tests % ls
README.md      can            i2s            sdio           watchdog
adc            fdcan          perf           spi
adcsd          gpio           pmu            timer
apb_soc        hello          ptypes         uart
bootconfig     i2c            rtc            udma
```

Example: GPIO

```
static void test_entry() {
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
}
```



Example: GPIO

```
static void test_entry() {
    struct pi_gpio_conf conf;
    struct pi_device gpio;
    pi_gpio_e pin = PI_GPIO_0;
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
}
```



Declare data structures available in the standard library.

Example: GPIO

```
static void test_entry() {
    struct pi_gpio_conf conf;
    struct pi_device gpio;
    pi_gpio_e pin = PI_GPIO_0;

    printf("GPIO Hello World Example\n");
    printf("Configuring GPIO pin %d as output\n", pin);
    ...
    ...
    ...
    ...
    ...
    ...
}
```



Use standard input/output functions.

Example: GPIO

```
static void test_entry() {
    ...
    pi_gpio_conf_init(&conf);
    conf.gpio_id = 0;
    pi_setup_from_conf(&gpio, &conf);
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
}
```



Initialize the GPIO configuration structure

Example: GPIO

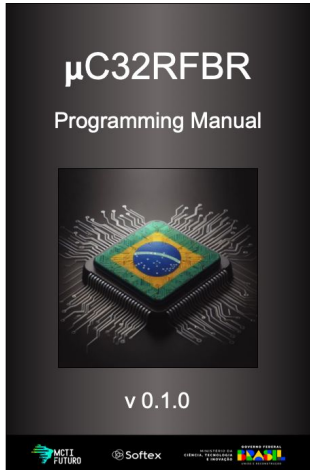
```
static void test_entry() {
    ...
    pi_gpio_conf_init(&conf);
    conf.gpio_id = 0;
    pi_setup_from_conf(&gpio, &conf);
    ...
    if (pi_gpio_open(&gpio)) {
        printf("Failed to open\n");
        pmsis_exit(-1);
    }
    ...
    ...
    ...
}
```



Open the GPIO device and check for errors.

Example: GPIO

```
static void test_entry() {
    ...
    pi_gpio_pin_configure(&gpio, pin,
        PI_GPIO_OUTPUT | PI_GPIO_ENABLE);
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
    ...
}
```



Configure the pin as output with enable.

Example: GPIO

```
static void test_entry() {
    ...
    for (int i = 0; i < 5; i++) {
        pi_gpio_pin_write(&gpio, pin, 1);
        printf("LED ON\n");

        pi_time_wait_us(500000);
        pi_gpio_pin_write(&gpio, pin, 0);
        printf("LED OFF\n");

        pi_time_wait_us(500000);
    }
    ...
}
```



Turn LED on.

Example: GPIO

```
static void test_entry() {
    ...
    for (int i = 0; i < 5; i++) {
        pi_gpio_pin_write(&gpio, pin, 1);
        printf("LED ON\n");

        pi_time_wait_us(500000);
        pi_gpio_pin_write(&gpio, pin, 0);
        printf("LED OFF\n");

        pi_time_wait_us(500000);
    }
    ...
}
```



Wait 500 ms.

Example: GPIO

```
static void test_entry() {
    ...
    for (int i = 0; i < 5; i++) {
        pi_gpio_pin_write(&gpio, pin, 1);
        printf("LED ON\n");

        pi_time_wait_us(500000);
        pi_gpio_pin_write(&gpio, pin, 0);
        printf("LED OFF\n");

        pi_time_wait_us(500000);
    }
    ...
}
```



Turn LED off.

Example: GPIO

```
static void test_entry() {
    ...
    ...
    ...
    ...
    printf("GPIO test completed successfully\n");
    ...
    ...
    ...
    ...
    pi_gpio_close(&gpio);
    pmsis_exit(0);
    ...
}
```



Close the GPIO device.

Final Thoughts



Learning Experience



UFMG's Compilers Lab

Learning Experience



UFMG's Compilers Lab





Learning Experience

Learning low-level programming





Learning Experience

Learning low-level programming

Computer Engineering at UFMG







Contact

Fernando Pereira

fernando@dcc.ufmg.br

http://lac.dcc.ufmg.br

