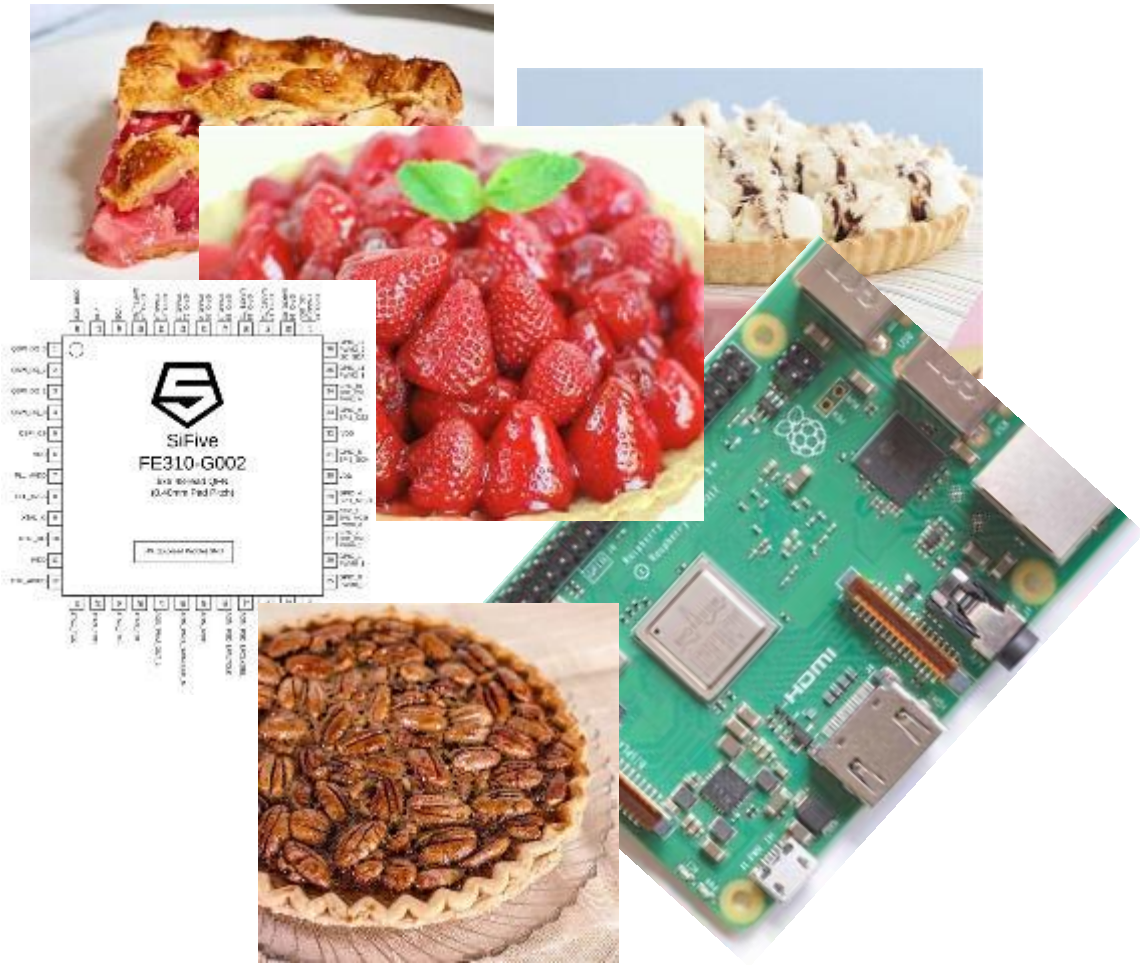
Flash Memory Summit

RISC FIVE AS EASY AS PI

Paul Sherman



Overview

- Installing the O/S
- Building the “Tool Chain”
- Configuring the Hardware
- Wiring the Hardware
- Assembling, Compiling, Linking, and Loading
- What Can Go Wrong
- Simple Terminal & Logic Analyzer

First Things, First

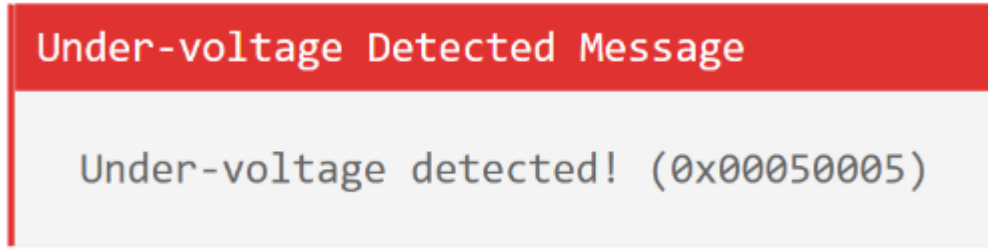
Flash Memory Summit

Low voltage supply (can) quickly kill an SD card, especially when it's used in a development system (assembler, compiler, linker, loader)

Use ~5.25 V, 2.5A supply with good, thick 20 AWG cables, such as:

Prevents this

and this



www.adafruit.com/product/1995
5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable, **$8.25**

Installing the O/S

raspberrypi.com/software – Raspberry PI Imager

Choose OS – Raspberry PI OS (Other) – Raspberry PI OS Lite (32-bit)

Choose STORAGE – Generic STORAGE DEVICE USB DEVICE

(gear) set hostname, uid, pwd, wifi, locale as desired

WRITE

To Clean an older SD card, if needed:

Run – diskpart – List Disk – Select disk x

- List Partition – Select partition x – Delete partition
- Create Partition Primary – Format fs=fat32

```
sudo rasp-config – Localization [*] en_US UTF-8

/boot/cmdline.txt : console=tty1 root=... rootfstype=ext4 fsck.repair=yes
                    quiet loglevel=3 logo.nologo rootwait

/boot/config.txt : disable_splash=1              dtparam=audio=off    camera-auto-detect=0
                   dtoverlay=[pi3-]disable-bt                        enable_uart=1
                   dtoverlay=[pi3-]disable-wifi

sudo sed –i ‘2i\ \ \ \ \ \ \ \ exit 0’ /etc/profile.d/wifi-check.sh

sudo apt-get update
sudo apt-get install autoconf automake autotools-dev curl python3 git
                     libmpc-dev libmpfr-dev libgmp-dev
                     gawk build-essential bison flex texinfo gperf
                     libtool patchutils bc zlib1g-dev libexpat-dev
                     libfdt-dev libisl-dev

sudo apt clean
sudo apt autoremove
```

Prevents start-up message “WiFi is currently blocked by rfkill”

for best Linux health:
DON’T! pull the plug before you
`sudo shutdown now`

Building the "Tool Chain"

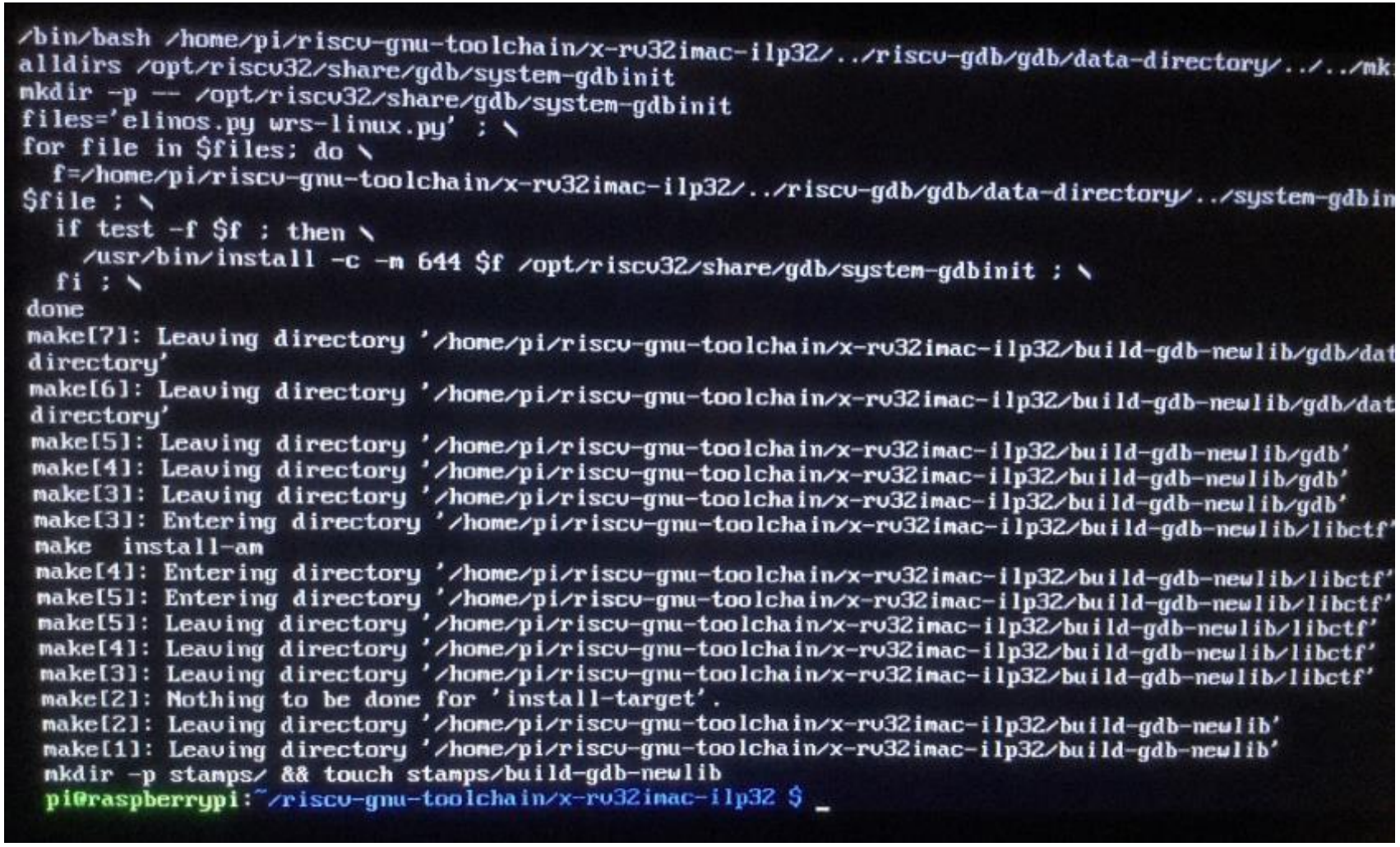

```
riscv32-unknown-elf-gcc --version - 11.1.0
                        -as  --version - 2.38
                        -ld  --version - 2.38
                        -gdb --version - 10.1
openocd --version - 0.11.0
```

```
sudo rm -fr /opt/riscv32
sudo rm -fr ./riscv-gnu-toolchain
git clone https://github.com/riscv/riscv-gnu-toolchain
cd riscv-gnu-toolchain
mkdir x-rv32imac-ilp32
cd x-rv32imac-ilp32
../configure -prefix=/opt/riscv32 --enable-languages=c,c++
                                  --with-arch=rv32imac
                                  --with-abi=ilp32
sudo make
export RISCV=/opt/riscv32
export PATH=$PATH:$RISCV/bin
```

DO NOT use many thread `-j` option, too hard on SD card

```
sudo apt-get install libusb-1.0-0  libusb-1.0-0-dev
sudo rm -fr ./openocd
git clone git://git.code.sf.net/p/openocd/code openocd
cd openocd
./bootstrap
./configure -prefix=/opt/openocd --enable-bcm2835gpio --enable-sysfsgpio
make
sudo make install
```


Toolchain Build Success



TOOLCHAIN BUILDING SEQUENCE

`binutils`

↓

`gcc`

↓

`newlib`

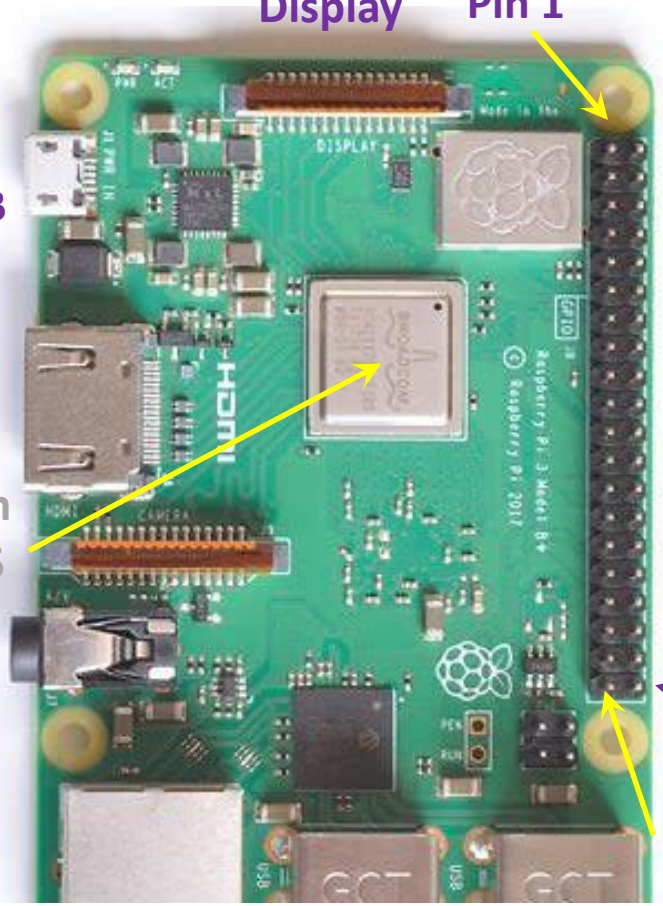
↓

`gdb`

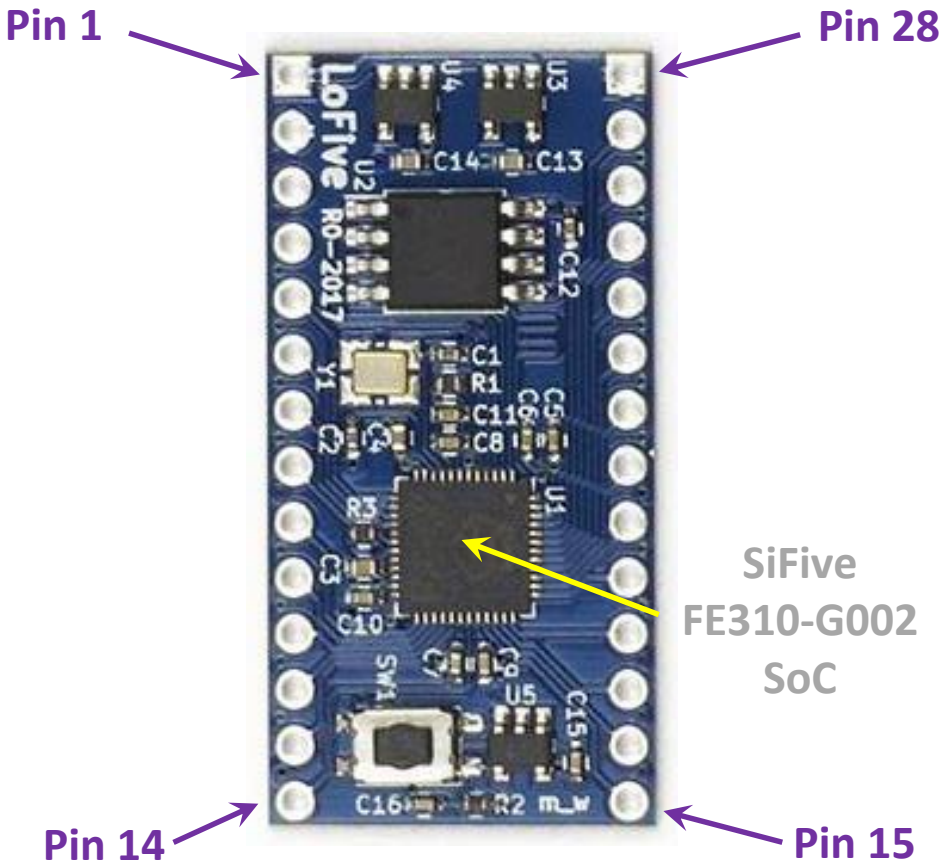
DONE!

What is the Hardware

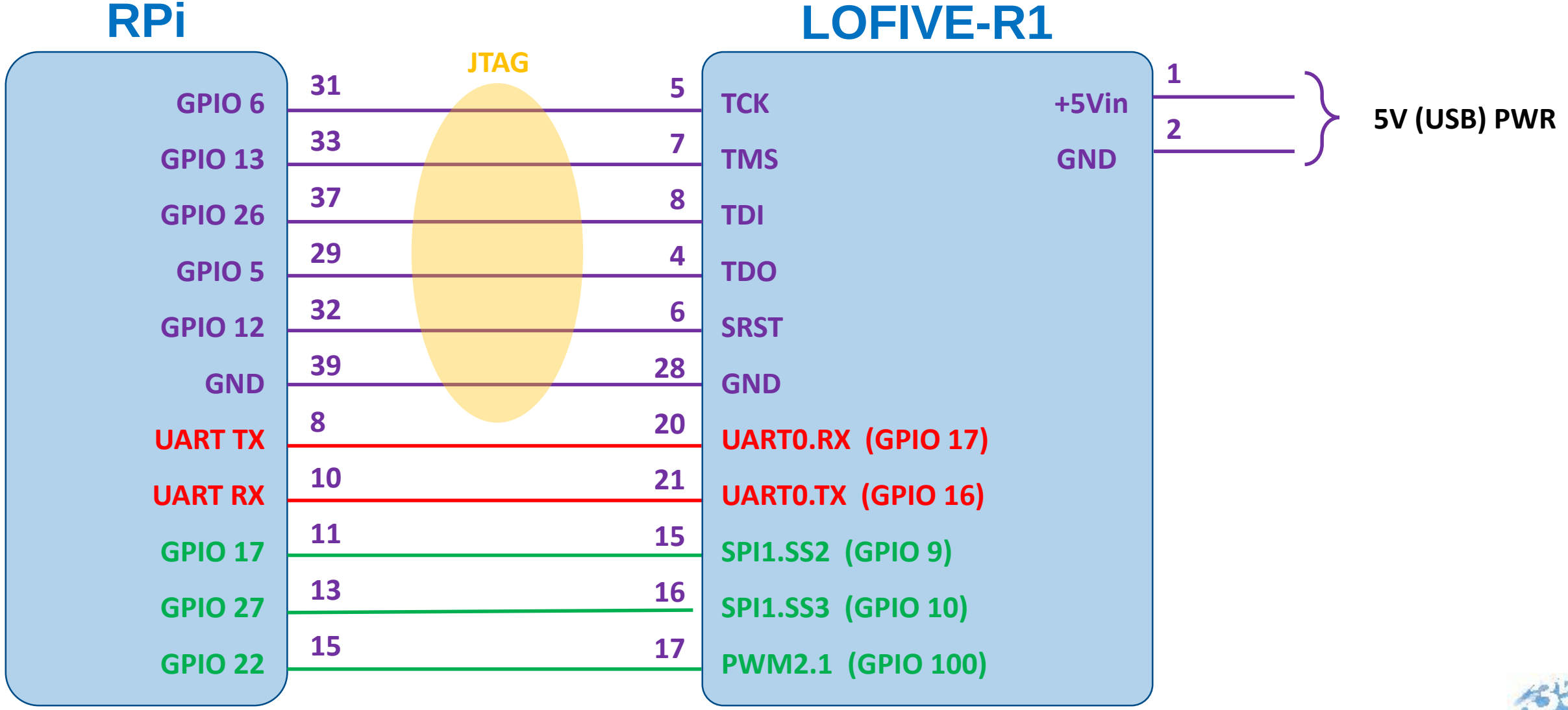
RPi



LOFIVE-R1



Wiring the Hardware



Assembling, Compiling, Linking – the Makefile

The MAKEFILE script does not need to be changed when switching between Flash and RAM boot or code execution

Notice the two places where the linker script file "*foo.lds*" gets used in the build process.

```
RISCVGNU ?= riscv32-unknown-elf
AOPS = -march=riscv32imac -mabi=ilp32
COPS = -march=riscv32imac -mabi=ilp32 -Wall -O2 -nostdlib -nostartfiles -ffreestanding

start.o : start.s
        $(RISCVGNU)-as $(AOPS) start.s -o start.o

... all other ASM and C source files go here ...

main.o : main.c
        $(RISCVGNU)-gcc $(COPS) -c main.c -o main.o

foo.bin : foo.lds start.o ... main.o
        $(RISCVGNU)-ld start.o ... main.o -T foo.lds -o foo.elf -Map foo.map
        $(RISCVGNU)-objdump -D foo.elf > foo.lst
        $(RISCVGNU)-objcopy foo.elf -O ihex foo.hex
        $(RISCVGNU)-objcopy foo.elf -O binary foo.bin
```

```
clean:
        rm -f *.o
        rm -f *.elf
        rm -f *.bin
        rm -f *.lst
        rm -f *.hex
        rm -f *.map
```


Linker Script

This is how to selectively load and/or boot from Flash (ROM) or RAM.
It is a bit bare but should be easy to see all of the moving parts.

There are only two places to change when making the choice between Flash (ROM) or RAM:
The linker script file "*foo.lds*" shown here,
and the Loading & Running command lines, shown next.

```
OUTPUT_ARCH("riscv")
ENTRY( _start_ )
MEMORY
{
    rom           : ORIGIN = 0x20000000, LENGTH = 0x2000
    ram (rxa!ri)  : ORIGIN = 0x80000000, LENGTH = 0x4000
}
SECTIONS
{
    .text   : { *(.text*) } > ram ... or ... rom
    .rodata : { *(.rodata*) } > ram ... or ... rom
    .bss    : { *(.bss*) } > ram
}
```

Table 4: *FE310-G002 Memory Map. Memory Attributes: **R** - Read, **W** - Write, **X** - Execute, **C** - Cacheable, **A** - Atomics*

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	

Loader Script

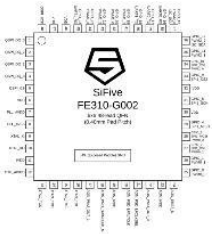
Interface specification – How to tell OpenOCD which pins and wires *of the host* system to use

```
rpi-3b.cfg    adapter driver bcm2835gpio
              bcm2835gpio peripheral_base 0x3f000000
              bcm2835gpio speed_coeffs 97469 24
              bcm2835gpio jtag_nums 6 13 26 5
              bcm2835gpio swd_nums  6 13
              bcm2835gpio srst_nums 12
              reset_config srst_only separate srst_nogate
```

This is where you define the connection signals TCK TMS TDI TDO in that order!

Target specification – How to tell OpenOCD what kind of chip to talk to

```
fe310-g002.cfg    transport select jtag
                  jtag newtap riscv cpu -irlen 5 -expected-id 0x20000913
                  target create riscv.cpu.0 riscv -chain-position riscv.cpu
                  riscv.cpu.0 configure -work-area-phys 0x80000000
                                        -work-area-size 0x100000
                                        -work-area-backup 0
```



Loading & Running

The Load & Run command lines need to change in two places when switching between RAM or Flash (ROM) boot, as shown by the highlighted statements.

LOAD

RAM

```
sudo openocd -f rpi-3b.cfg -f fe310-g002.cfg -c "adapter speed 1000" -c init -c "reset init"
-c "sleep 25" -c "riscv set_reset_timeout_sec 25" -c "adapter speed 2500" -c "load_image
foo.bin 0x80000000 bin" -c "verify_image foo.bin 0x80000000 bin" -c shutdown -c exit
```

ROM

```
sudo openocd -f rpi-3b.cfg -f fe310-g002.cfg -c "flash bank spi0 fespi 0x20000000 0 0 0
riscv.cpu.0 0x10014000" -c "adapter speed 1000" -c init -c "reset init" -c "sleep 25" -c
"riscv set_reset_timeout_sec 25" -c "adapter speed 2500" -c "flash write_image erase unlock
foo.bin 0x20000000 bin" -c shutdown -c exit
```

RUN

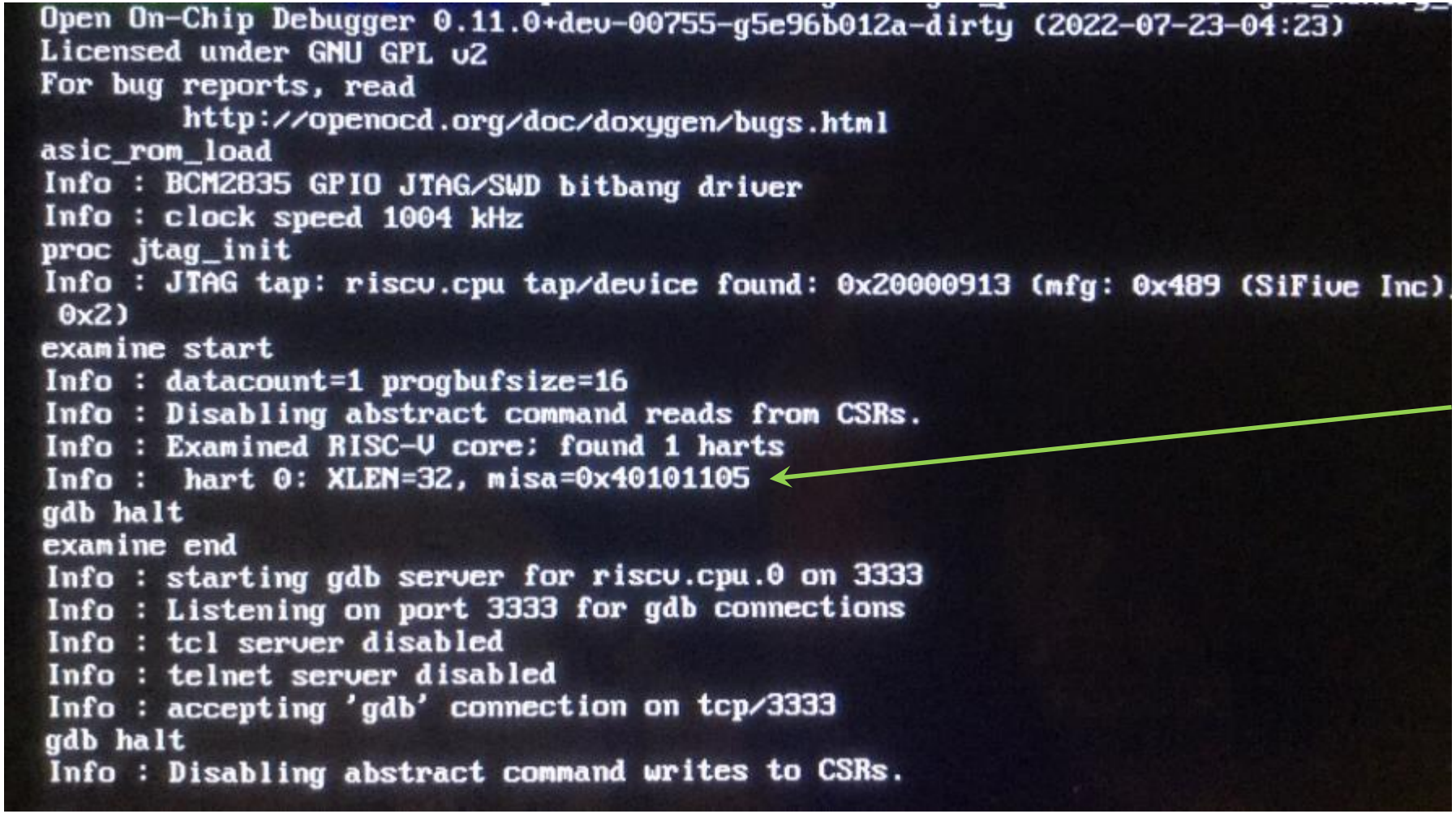
RAM

```
sudo openocd -f rpi-3b.cfg -f fe310-g002.cfg -c "adapter speed 1000" -c init -c "reset init"
-c "sleep 25" -c "adapter speed 2500" -c "resume 0x80000000" -c shutdown -c exit
```

ROM

```
sudo openocd -f rpi-3b.cfg -f fe310-g002.cfg -c "adapter speed 1000" -c init -c "reset init"
-c "sleep 25" -c "adapter speed 2500" -c "resume 0x20000000" -c shutdown -c exit
```


Load & Run Success



Successful connection to target

Load & Run Unsuccessful


```
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : clock speed 100 kHz
proc jtag_init
Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive Inc), part: 0x0000, ver:
 0x2)
examine start
proc asic_reset
asic_reset: pulsing reset line
Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive Inc), part: 0x0000, ver:
 0x2)
asic_reset: wait for target get into reset state (prevent impatient scan retries)
Error executing event examine-start on target riscv.cpu.0:
foo.cfg:100: Error:
at file "foo.cfg", line 100
Error: DMI operation didn't complete in 2 seconds. The target is either really slow or broken. You c
ould increase the timeout with riscv set_command_timeout_sec.
Error: DMI operation didn't complete in 2 seconds. The target is either really slow or broken. You c
ould increase the timeout with riscv set_command_timeout_sec.
examine fail ... OOPS!
Warn : target riscv.cpu.0 examination failed
Info : starting gdb server for riscv.cpu.0 on 3333
Info : Listening on port 3333 for gdb connections
Info : tcl server disabled
Info : telnet server disabled
^Cshutdown command invoked

pi@raspberrypi:~/prj/foo $ _
```

Unsuccessful

Unsuccessful

```
Info : clock speed 1004 kHz
proc jtag_init
Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFiv
 0x2)
Info : datacount=1 progbufsize=16
Error: unable to halt hart 0
Error:   dmcontrol=0x80000001
Error:   dmstatus =0x00030c82
Error: Fatal: Hart 0 failed to halt during examine()
Warn : target riscv.cpu.0 examination failed
Info : gdb port disabled
shutdown command invoked
```

unsuccessful Connection jtag not reset?

Sample Program

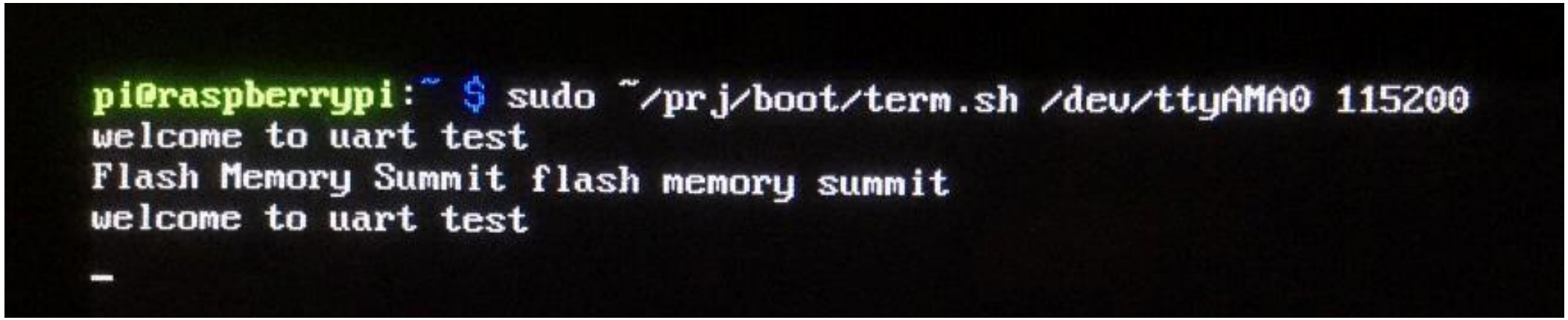

```
#include <stdint.h>  // for uint32_t
#include <stddef.h>  // for size_t
#include "clock.h>
#include "uart0.h"
#include "gpio.h"
unsigned int x;
uint32_t clk_hz;
```

```
void main() {
    clk_hz = clock_init( PRCI_EXT_DIR );
    gpio_init();
    uart0_init( clk_hz, 115200 );
    uart0_write_string( "welcome to uart test\r\n");
    gpio_dir( 9, GPIO_OUT ); gpio_dir( 10, GPIO_OUT ); gpio_dir( 11, GPIO_OUT );
    while(1) {
        x = uart_read();
        switch( x ) {
            case 'F': uart0_write_string("Flash "); gpio_high( 9 ); break;
            case 'f': uart0_write_string("flash "); gpio_low( 9 ); break;
            case 'M': uart0_write_string("Memory "); gpio_high( 10 ); break;
            case 'm': uart0_write_string("memory "); gpio_low( 10 ); break;
            case 'S': uart0_write_string("Summit "); gpio_high( 11 ); break;
            case 's': uart0_write_string("summit "); gpio_low( 11 ); break;
            case '\r': uart0_write_string("\r\n"); break;
            default: uart0_write( (uint8_t *) &x, 1); break;
        }
    }
}
```

available at https://github.com/psherman42

Simple Terminal


```
sudo ~/prj/boot/term.sh /dev/serial0 115200
```




```
#!/bin/bash
set -e
bak="$(sty -g)"
trap 'set +e; kill "$bgPid"; sty "$bak"' EXIT
port="$1"; shift
stty -F "$port" raw -echo "$@"
sty raw -echo isig intr ^Q quit undef susp undef
cat "$port" & bgPid=$!
cat >"$port"
```

available at https://github.com/psherman42

Linux Logic Analyzer


```
c1, c2, c2 - channel GPIO pin(s)
tc - trigger channel GPIO pin
tp - trigger polarity (+ or -)
tm - trigger mode (auto or norm)
cl1, cl2, cl3 - channel label(s)
```

```
sudo ~/prj/boot/sense.sh --c1 17 --c2 27 --c3 22
                         --tc 17 --tp + --tm norm
                         --cl1 GPIO17 --cl2 GPIO-27 --cl3 GPIO-22
```

available at https://github.com/psherman42

References

SiFive Docs – https://www.sifive.com/documentation

E31 Core Complex Manual, Freedom E310 Datasheet & Manual

https://forums.sifive.com

https://github.com/sifive/sifive-blocks

Good technical discussions.
See HiFive1 Rev B, user: **pds**

LoFive R1 – https://github.com/mwelling/lofive

RPi – https://pinout.xyz

https://www.raspberrypi.com/software

USB Adapters: Olimex, FTDI FT-2232, etc.

Availability: digikey, mouser, adafruit

Acknowledgments

Erich Haratsch & Tom Coughlin – this great opportunity

Engling Yeo – Professional & Technical challenges to overcome

Many colleagues, friends, family – being there, always

All of you – this morning, bright eyed & bushy tailed

Is RISC Five as easy as Mac or PC?

Flash Memory Summit

It sure is! Use the FT(2)232 chip with any USB port.

- Mac – drivers already supported
- PC – may need to disable the UEFI driver security check

Reset line glitches at startup, so revise a little bit as below

`./openocd/share/openocd/scripts/interface/ftdi/olimex-...`

```
ftdi.cfg adapter driver ftdi
         ftdi device_desc "Olimex OpenOCD JTAG ARM-USB-TINY-H"
         ftdi vid_pid 0x15ba 0x002a

        #----------------- P/U --- DIR --
        #ftdi layout_init 0x0808  0x0a1b
         ftdi layout_init 0x0b08  0x0b1b

         ftdi layout_signal nSRST -oe 0x0200
         ftdi layout_signal nTRST -data 0x0100 -oe 0x0100
         ftdi layout_signal LED -data 0x0800
```

BUG!!! tiny 10us glitch on nTRST at startup

BUG FIX: rst lines (out) and (push-pull)

ftdi layout init

Sig	MPSSE	PIN	BIT	P/U	DIR
TCK	TCK/SK	ADBUS0	0	0	1
TDI	TDI/DO	ADBUS1	1	0	1
TDO	TDO/DI	ADBUS2	2	0	0
TMS	TMS/CS	ADBUS3	3	1	1
???	GPIOL0	ADBUS4	4	0	1
.	GPIOL1	ADBUS5	5	0	0
.	GPIOL2	ADBUS6	6	0	0
.	GPIOL3	ADBUS7	7	0	0
TRST	GPIOH0	ACBUS0	8	1	1
SRST	GPIOH1	ACBUS1	9	1	1
.	GPIOH2	ACBUS2	a	0	0
LED	GPIOH3	ACBUS3	b	1	1
.	GPIOH4	ACBUS4	c	0	0
.	GPIOH5	ACBUS5	d	0	0
.	GPIOH6	ACBUS6	e	0	0
.	GPIOH7	ACBUS7	f	0	0

docs at https://ftdichip.com/products/ft2232hq
and https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY-H

Can I do it all with one click (or key press)?

Yes!

```
make -f foo.mk ram -tgt=LOAD
```

Link step

```
ram : foo.lds start.o ... main.o
        $(RISCVGNU)-ld start.o ... main.o -T foo.lds -o foo.elf -Map foo.map
        $(RISCVGNU)-objdump -D foo.elf > foo.lst
        $(RISCVGNU)-objcopy foo.elf -O ihex foo.hex
        $(RISCVGNU)-objcopy foo.elf -O binary foo.bin
ifeq ($(tgt), LOAD)
        @openocd -f interface/ftdi/olimex-arm-usb-tiny-h.cfg -f foo.cfg
                                                -c init -c "asic_ram_load foo"
                                                -c shutdown -c exit
else
        @echo "target not changed"
endif
```

note the @ symbol to run a shell command

Optional Load step

see https://github.com/psherman/Demystifying-OpenOCD

Contact

Paul Sherman

Email: pauldylansherman@icloud.com

(510) 229-6249

Github: psherman42

