Booting linux on ESP32s3

Sources:

- https://github.com/jcmvbkbc/xtensa-dynconfig/tree/original
- https://github.com/jcmvbkbc/config-esp32s3
- https://github.com/jcmvbkbc/esp-idf/tree/linux-5.0.1
- https://github.com/jcmvbkbc/linux-xtensa/tree/xtensa-6.6-rc7-esp32
- https://github.com/jcmvbkbc/binutils-gdb-xtensa/tree/xtensa-2.40-fdpic
- https://github.com/jcmvbkbc/gcc-xtensa/tree/xtensa-14-4992-fdpic
- https://github.com/jcmvbkbc/uclibc-ng-xtensa/tree/xtensa-1.0.44-fdpic
- https://github.com/jcmvbkbc/buildroot/tree/xtensa-2023.08-fdpic
- https://github.com/jcmvbkbc/crosstool-NG/tree/xtensa-fdpic

Scripts with all steps below: https://github.com/jcmvbkbc/esp32-linux-build

Things that work

- WiFi. Use the script that builds firmware based on esp-hosted. It runs on core 0, linux runs on core 1, special linux IPC is used for communication. WiFi transport that uses linux IPC is added both to the firmware and to the linux kernel wifi driver.
- Writing to FLASH and using ESP FLASH partition table. Driver based on linux IPC sends FLASH-related requests to the firmware. Default configuration has an etc partition that is flashed with /etc file system and mounted at boot time. The file system is writable and it can be used to store things like wpa_supplicant.conf, /etc/passwd, /etc/shadow, …
- USB serial. It is visible as the /dev/ttyACM1 (in kernels based on v6.5) or the /dev/ttyGS3 (in kernels based on v6.6-rc and newer) device inside the linux environment. (The change of name was requested by the linux TTY subsystem maintainer, the change of number is to keep UART ports numbering coherent with esp32s3 TRM).
- GPIO, including interrupts. Software I2C over GPIO and software SPI over GPIO.
- hardware SPI. Tested clock speed up to 20MHz with SD card.
- clock frequency detection. CPU, XTAL and APB clocks may be preset by the bootloader, the kernel will understand and use preset frequencies without additional configuration.
- ssh server and ssh client. There's an issue using scp from openssh version 9 because that version switched from the original scp protocol to sftp. The option -O makes it use the now legacy scp.
- mounting NFS shares.
- running executable code from outside the rootfs, e.g. from /etc or from NFS mounts. Code in the linux bootloader maps PSRAM to the IRAM address range and a special hack in the kernel code does address remapping, so that when an executable memory mapping is created, the mapping address points to the PSRAM alias in the IRAM address range.
- passing command line from bootloader to the kernel. Bootloader reads the file /etc/cmdline if it exists and passes its contents to the kernel as the command line. The bootloader does JFFS2 file system parsing for that.
- strace (a one-line fix is needed for the mainline strace to correctly handle the initial exec).
- perf stat (heavy patching is needed to build it for nommu). -D1 is needed to properly enable events (perf relies on ability to run code after the fork but before the exec in the child process to manage events on systems with mmu, -D1 looks like a good workaround for nommu case).

- c++ exceptions, c cleanup routines, forced stack unwinding, unwinding over signal frames.

Things that don't work

- about 290 failing tests in the gcc testsuite, most related to TLS and linuxthreads limitations. About 300 failing tests in the g++ testsuite.
- NPTL. Not yet. Need to define relevant TLS relocations, teach the compiler to use them, the assembler to understand them and the linker to relax them.
- mmap with MAP_FIXED flag. By design of the nommu linux, but it seems to me that it doesn't have to be like that. It's usually not a big deal, but that's the reason there's no module information in the /proc/*/maps other than for the ld.so and the executable.
- tcpdump and libpcap in general. It tries to mmap the packet socket and it's missing a few things (mm/nommu.c doesn't know what capabilities to assign to S_IFSOCK files, the socket file needs to have get_unmapped_area callback defined), but most importantly the packet_mmap code heavily relies on the presence of mmu. Although it seems that it could be worked around.
- bluetooth. It requires heavy userspace support that includes dbus which wants fork. Looks like it can be worked around, but the amount of bloat is discouraging.
- perf record. It requires mmapping perf events and the kernel code somewhat relies on the presence of mmu. I've tried to add a workaround, but something is still missing.

Building/flashing/running

```
$ git clone https://github.com/jcmvbkbc/esp32-linux-build
$ cd esp32-linux-build
$ ./rebuild-esp32s3-linux-wifi.sh
```

It will run the following steps to build the toolchain, the kernel, the root and etc filesystem images and the firmware and it will flash it.

Build toolchain dynconfig library and export XTENSA_GNU_CONFIG for use by the toolchain

```
$ git clone https://github.com/jcmvbkbc/xtensa-dynconfig -b original
$ git clone https://github.com/jcmvbkbc/config-esp32s3 esp32s3
$ make -C xtensa-dynconfig ORIG=1 CONF_DIR=`pwd` esp32s3.so
$ export XTENSA_GNU_CONFIG=`pwd`/xtensa-dynconfig/esp32s3.so # make sure
this environment variable is set for all commands involving building or
using the toolchain
```

Build the toolchain

```
$ git clone https://github.com/jcmvbkbc/crosstool-NG.git -b xtensa-fdpic
$ pushd crosstool-NG
```

```
$ ./bootstrap && ./configure --enable-local && make
$ ./ct-ng xtensa-esp32s3-linux-uclibcfdpic
$ CT_PREFIX=`pwd`/builds nice ./ct-ng build
$ popd
```

It doesn't complete successfully ATM, failing with the following message, but it builds and installs everything that's important.

```
...
[INFO ]  Installing final gcc compiler: done in 572.47s (at 29:47)
[INFO ]  ================================================================
[INFO ]  Checking dynamic linker symlinks
[EXTRA]    Checking dynamic linker for multilib ''
[ERROR]    collect2: error: ld returned 1 exit status
[ERROR]
[ERROR]  >>
[ERROR]  >>  Build failed in step 'Checking dynamic linker symlinks'
[ERROR]  >>        called in step '(top-level)'
[ERROR]  >>
[ERROR]  >>  Error happened in: CT_DoExecLog[scripts/functions@377]
[ERROR]  >>        called from: CT__FixupLDSO[scripts/functions@1695]
[ERROR]  >>        called from: CT_IterateMultilibs[scripts/functions@1608]
[ERROR]  >>        called from: CT_MultilibFixupLDSO[scripts/functions@1761]
[ERROR]  >>        called from: uClibc_ng_post_cc[scripts/build/libc/uClibc-
ng.sh@335]
[ERROR]  >>        called from: do_libc_post_cc[scripts/build/libc.sh@38]
[ERROR]  >>        called from: main[scripts/crosstool-NG.sh@697]
[ERROR]  >>
...
```

Build the rootfs and kernel image

```
$ git clone https://github.com/jcmvbkbc/buildroot -b xtensa-2023.08-fdpic
$ nice make -C buildroot O=`pwd`/build-buildroot-esp32s3 esp32s3_defconfig
$ buildroot/utils/config --file build-buildroot-esp32s3/.config --set-str
TOOLCHAIN_EXTERNAL_PATH `pwd`/crosstool-NG/builds/xtensa-esp32s3-linux-
uclibcfdpic
$ buildroot/utils/config --file build-buildroot-esp32s3/.config --set-str
TOOLCHAIN_EXTERNAL_PREFIX '$(ARCH)-esp32s3-linux-uclibcfdpic'
$ buildroot/utils/config --file build-buildroot-esp32s3/.config --set-str
TOOLCHAIN_EXTERNAL_CUSTOM_PREFIX '$(ARCH)-esp32s3-linux-uclibcfdpic'
$ nice make -C buildroot O=`pwd`/build-buildroot-esp32s3
```

Build and flash the bootloader, flash kernel and rootfs images

```
$ git clone https://github.com/jcmvbkbc/esp-idf -b linux-5.0.1
$ pushd esp-idf
$ . export.sh
```

```
$ cd examples/get-started/linux_boot
$ idf.py set-target esp32s3
$ cp sdkconfig.defaults.esp32s3 sdkconfig
$ idf.py build
$ idf.py flash
$ popd
$ parttool.py write_partition --partition-name linux  --input build-
buildroot-esp32s3/images/xipImage
$ parttool.py write_partition --partition-name rootfs --input build-
buildroot-esp32s3/images/rootfs.cramfs
$ parttool.py write_partition --partition-name etc    --input build-
buildroot-esp32s3/images/etc.jffs2
```

The result

```
ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3810,len:0x10a0
load:0x403c9700,len:0xa24
load:0x403cc700,len:0x2d04
entry 0x403c988c
I (73) octal_psram: vendor id    : 0x0d (AP)
I (73) octal_psram: dev id       : 0x02 (generation 3)
I (74) octal_psram: density      : 0x03 (64 Mbit)
I (78) octal_psram: good-die     : 0x01 (Pass)
I (83) octal_psram: Latency      : 0x01 (Fixed)
I (89) octal_psram: VCC          : 0x01 (3V)
I (93) octal_psram: SRF          : 0x01 (Fast Refresh)
I (99) octal_psram: BurstType    : 0x01 (Hybrid Wrap)
I (105) octal_psram: BurstLen     : 0x01 (32 Byte)
I (110) octal_psram: Readlatency  : 0x02 (10 cycles@Fixed)
I (117) octal_psram: DriveStrength: 0x00 (1/1)
I (122) esp_psram: Found 8MB PSRAM device
I (126) esp_psram: Speed: 80MHz
I (130) cpu_start: Pro cpu up.
I (134) cpu_start: Starting app cpu, entry point is 0x40375344
I (0) cpu_start: App cpu up.
I (593) esp_psram: SPI SRAM memory test OK
I (602) cpu_start: Pro cpu start user code
I (602) cpu_start: cpu freq: 160000000 Hz
I (602) cpu_start: Application information:
I (605) cpu_start: Project name:     linux_boot
I (610) cpu_start: App version:      v5.0.1-4-g680509ab40d1
I (617) cpu_start: Compile time:     May  7 2023 16:29:12
I (623) cpu_start: ELF file SHA256:  a110e4309915b853...
```

```
I (629) cpu_start: ESP-IDF:          v5.0.1-4-g680509ab40d1
I (635) cpu_start: Min chip rev:     v0.0
I (640) cpu_start: Max chip rev:     v0.99
I (644) cpu_start: Chip rev:         v0.1
I (649) heap_init: Initializing. RAM available for dynamic allocation:
I (656) heap_init: At 3FC958C0 len 00053E50 (335 KiB): D/IRAM
I (663) heap_init: At 3FCE9710 len 00005724 (21 KiB): STACK/DRAM
I (669) heap_init: At 3FCF0000 len 00008000 (32 KiB): DRAM
I (676) heap_init: At 600FE010 len 00001FF0 (7 KiB): RTCRAM
I (682) esp_psram: Adding pool of 8192K of PSRAM memory to heap allocator
I (690) spi_flash: detected chip: generic
I (694) spi_flash: flash io: dio
I (699) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
I (719) esp_psram: Reserving pool of 32K of internal memory for DMA/internal
allocations
ptr = 0x42830000
ptr = 0x42b30000
[    0.000000] Ignoring boot parameters at (ptrval)
[    0.000000] Linux version 6.3.0-00022-g5d8354462a70 (jcmvbkbc@octofox)
(xtensa-dc233c-elf-gcc (GCC) 13.1.0, GNU ld (GNU Binutils) 2.40) #39 PREEMPT
Sun May  7 16:35:44 PDT 2023
[    0.000000] config ID: c2f0fffe:23090f1f
[    0.000000] earlycon: esp32uart0 at MMIO32 0x60000000 (options
'115200n8')
[    0.000000] printk: bootconsole [esp32uart0] enabled
[    0.000000] **********************************************************
[    0.000000] **   NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE   **
[    0.000000] **                                                      **
[    0.000000] ** This system shows unhashed kernel memory addresses   **
[    0.000000] ** via the console, logs, and other interfaces. This    **
[    0.000000] ** might reduce the security of your system.           **
[    0.000000] **                                                      **
[    0.000000] ** If you see this message and you are not debugging    **
[    0.000000] ** the kernel, report this immediately to your system   **
[    0.000000] ** administrator!                                       **
[    0.000000] **                                                      **
[    0.000000] **   NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE   **
[    0.000000] **********************************************************
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x000000003c030000-0x000000003c82ffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x000000003c030000-0x000000003c82ffff]
[    0.000000] Initmem setup node 0 [mem
0x000000003c030000-0x000000003c82ffff]
[    0.000000] pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
[    0.000000] pcpu-alloc: [0] 0
[    0.000000] Built 1 zonelists, mobility grouping off.  Total pages: 2032
[    0.000000] Kernel command line:
earlycon=esp32uart,mmio32,0x60000000,115200n8 console=ttyS0,115200n8 debug
```

```
rw root=mtd:data no_hash_pointers
[    0.000000] Dentry cache hash table entries: 1024 (order: 0, 4096 bytes,
linear)
[    0.000000] Inode-cache hash table entries: 1024 (order: 0, 4096 bytes,
linear)
[    0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[    0.000000] virtual kernel memory layout:
[    0.000000]     lowmem  : 0x3c030000 - 0x3c830000  (    8 MB)
[    0.000000]     .text   : 0x42830000 - 0x429e0d28  ( 1731 kB)
[    0.000000]     .rodata : 0x429e1000 - 0x42a1f000  (  248 kB)
[    0.000000]     .data   : 0x3c030000 - 0x3c0a9420  (  485 kB)
[    0.000000]     .init   : 0x3c0a9420 - 0x3c0adf00  (   18 kB)
[    0.000000]     .bss    : 0x3c0adf00 - 0x3c0e1988  (  206 kB)
[    0.000000] Memory: 7332K/8192K available (1731K kernel code, 485K
rwdata, 248K rodata, 88K init, 206K bss, 860K reserved, 0K cma-reserved)
[    0.000000] SLUB: HWalign=16, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] rcu: Preemptible hierarchical RCU implementation.
[    0.000000] rcu: RCU calculated value of scheduler-enlistment delay is 10
jiffies.
[    0.000000] NR_IRQS: 33
[    0.000000] rcu: srcu_init: Setting srcu_struct sizes based on
contention.
[    0.000000] clocksource: ccount: mask: 0xffffffff max_cycles: 0xffffffff,
max_idle_ns: 11945377789 ns
[    0.000086] sched_clock: 32 bits at 160MHz, resolution 6ns, wraps every
13421772796ns
[    0.008110] Calibrating delay loop (skipped)... 160.00 BogoMIPS preset
[    0.014370] pid_max: default: 4096 minimum: 301
[    0.021337] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes,
linear)
[    0.026541] Mountpoint-cache hash table entries: 1024 (order: 0, 4096
bytes, linear)
[    0.077044] rcu: Hierarchical SRCU implementation.
[    0.077673] rcu:     Max phase no-delay instances is 1000.
[    0.091972] devtmpfs: initialized
[    0.115533] clocksource: jiffies: mask: 0xffffffff max_cycles:
0xffffffff, max_idle_ns: 19112604462750000 ns
[    0.116804] futex hash table entries: 16 (order: -5, 192 bytes, linear)
[    0.139676] NET: Registered PF_NETLINK/PF_ROUTE protocol family
[    0.156790] platform soc: Fixed dependency cycle(s) with
/soc/intc@600c2000
[    0.219233] clocksource: Switched to clocksource ccount
[    0.266507] NET: Registered PF_INET protocol family
[    0.274408] IP idents hash table entries: 2048 (order: 2, 16384 bytes,
linear)
[    0.294920] tcp_listen_portaddr_hash hash table entries: 1024 (order: 0,
4096 bytes, linear)
[    0.296128] Table-perturb hash table entries: 65536 (order: 6, 262144
bytes, linear)
```

```
[    0.303770] TCP established hash table entries: 1024 (order: 0, 4096
bytes, linear)
[    0.309855] TCP bind hash table entries: 1024 (order: 1, 8192 bytes,
linear)
[    0.315629] TCP: Hash tables configured (established 1024 bind 1024)
[    0.326107] UDP hash table entries: 256 (order: 0, 4096 bytes, linear)
[    0.329877] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes,
linear)
[    0.340029] NET: Registered PF_UNIX/PF_LOCAL protocol family
[    0.361833] workingset: timestamp_bits=30 max_order=11 bucket_order=0
[    2.712542] 60000000.serial: ttyS0 at MMIO 0x60000000 (irq = 1, base_baud
= 0) is a ESP32 UART
[    2.714408] printk: console [ttyS0] enabled
[    2.714408] printk: console [ttyS0] enabled
[    2.720147] printk: bootconsole [esp32uart0] disabled
[    2.720147] printk: bootconsole [esp32uart0] disabled
[    2.759786] physmap-flash 42830000.flash: physmap platform flash device:
[mem 0x42830000-0x4302ffff]
[    2.761481] 2 fixed-partitions partitions found on MTD device
42830000.flash
[    2.765620] Creating 2 MTD partitions on "42830000.flash":
[    2.772746] 0x000000000000-0x000000300000 : "linux"
[    2.787049] 0x000000300000-0x000000880000 : "data"
[    2.787809] mtd: partition "data" extends beyond the end of device
"42830000.flash" -- size truncated to 0x500000
[    2.814687] NET: Registered PF_PACKET protocol family
[    3.044471] cramfs: checking physical address 0x42b30000 for linear
cramfs image
[    3.045351] cramfs: linear cramfs image on mtd:data appears to be 1808 KB
in size
[    3.052366] VFS: Mounted root (cramfs filesystem) readonly on device
31:1.
[    3.057890] devtmpfs: mounted
[    3.062212] Freeing unused kernel image (initmem) memory: 12K
[    3.064482] This architecture does not have kernel memory protection.
[    3.072536] Run /sbin/init as init process
[    3.074991]   with arguments:
[    3.077870]     /sbin/init
[    3.081602]   with environment:
[    3.083725]     HOME=/
[    3.086028]     TERM=linux
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
seedrng: can't create directory '/var/lib/seedrng': Read-only file system
Starting network: OK

Welcome to Buildroot
buildroot login: root
~ # cat /proc/cpuinfo
CPU count       : 1
```

```
CPU list        : 0
vendor_id       : Tensilica
model           : Xtensa LX7.0.12
core ID         : LX7_ESP32_S3_MP
build ID        : 0x90f1f
config ID       : c2f0fffe:23090f1f
byte order      : little
cpu MHz         : 160.00
bogomips        : 320.00
flags           : nmi debug ocd density boolean loop nsa minmax sext clamps
mac16 mul16 mul32 mul32h fpu s32c1i
physical aregs  : 64
misc regs       : 4
ibreak          : 2
dbreak          : 2
num ints        : 32
ext ints        : 26
int levels      : 6
timers          : 3
debug level     : 6
icache line size: 4
icache ways     : 1
icache size     : 0
icache flags    :
dcache line size: 16
dcache ways     : 1
dcache size     : 0
dcache flags    :
~ # free
              total        used        free      shared  buff/cache
available
Mem:           7344        3264        3444           0         636
3208

~ #
```

From:
http://wiki.osll.ru/ - **Open Source & Linux Lab**

Permanent link:
http://wiki.osll.ru/doku.php/etc:users:jcmvbkbc:linux-xtensa:esp32s3?rev=1701330592

Last update: **2023/11/30 10:49**