

ESP32 PDP8 simulator

1 Introduction.

This document describes a pdp8 simulator that runs on an ESP32 SOC. The emulator simulates the pdp8 and some peripherals like an RK8E disk and a TU56 dectape drive capable of running the OS/8 operating system. The pdp8 is a minicomputer from the sixties. OS/8 is developed in the seventies. On the internet there is plenty of information available about this remarkable machine. The software is developed using Eclipse IDE and the Espressif esp-idf toolchain. The speed of the simulated pdp8 is about as fast as an original pdp8/e machine, but disk/tape I/O is much faster. The simulator will emulate a pdp8 like this:

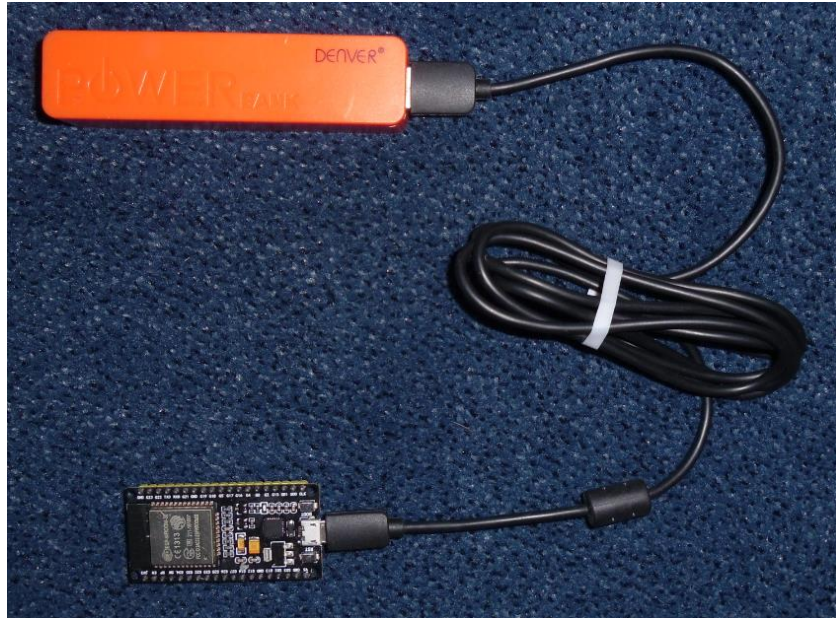


An RX01 floppy disk drive is also simulated.

2 Hardware.

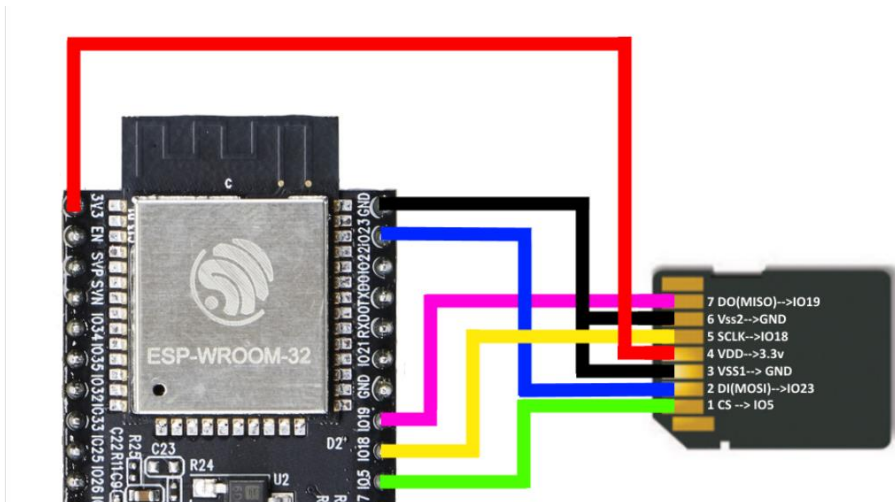
2.1 Minimal version.

The hardware used for the simulator is an ESP32 development board. An SD-card can be connected to the VSPI bus of the ESP32 (pins 18,19,23) for storage of several OS/8 compatible images. The CS pin used for the SD card can be defined with “make menuconfig” in the esp-idf. The simulator can be used without the SD card. The console terminal is connected through a TELNET connecting using the WiFi capabilities of the ESP32. A minimal configuration could look like this:



2.2 SD Card (optional).

This is how an SD card can be connected:



2.3 To Do.

In the coming months I will build a suitable case for the simulator including the famous pdp8 blinking lights.

3 Partitioning.

The OS/8 block devices like RKA0:, RKB0:, DTA0: are emulated as partitions in the 4 MB flash memory of the ESP32. The current layout (partitions.cvs) is:

#	Name,	Type,	SubType,	Offset,	Size,	Flags
nvs,	data,	nvs,		0x009000,	0x006000,	
phy_init,	data,	phy,		0x00F000,	0x001000,	
factory,	app,	factory,		0x010000,	0x094000,	
RKA0,	data,	0x80,		0x0A4000,	0x140000,	
RKB0,	data,	0x80,		0x1E4000,	0x140000,	
DTA0,	data,	0x81,		0x324000,	0x04C000,	
DTA1,	data,	0x81,		0x370000,	0x04C000,	
RXA0,	data,	0x82,		0x3BC000,	0x034000,	
TMP0,	data,	0x83,		0x3EC000,	0x010000,	

The last partition is reserved for a scratch disk of just 168 blocks.

A sector (1000 hexadecimal bytes = 4kB) in the flash memory can hold 21 pdp8 pages of 128 packed words. A sector in flash can only be written to as a whole block of 4 kB. To minimize the number of writes (and reads), a cache of at least 16 sectors are buffered in the simulator. The dirty cache buffers (caused by write operations) will be saved in flash every 5 minutes or by using the “PO” command in the console.

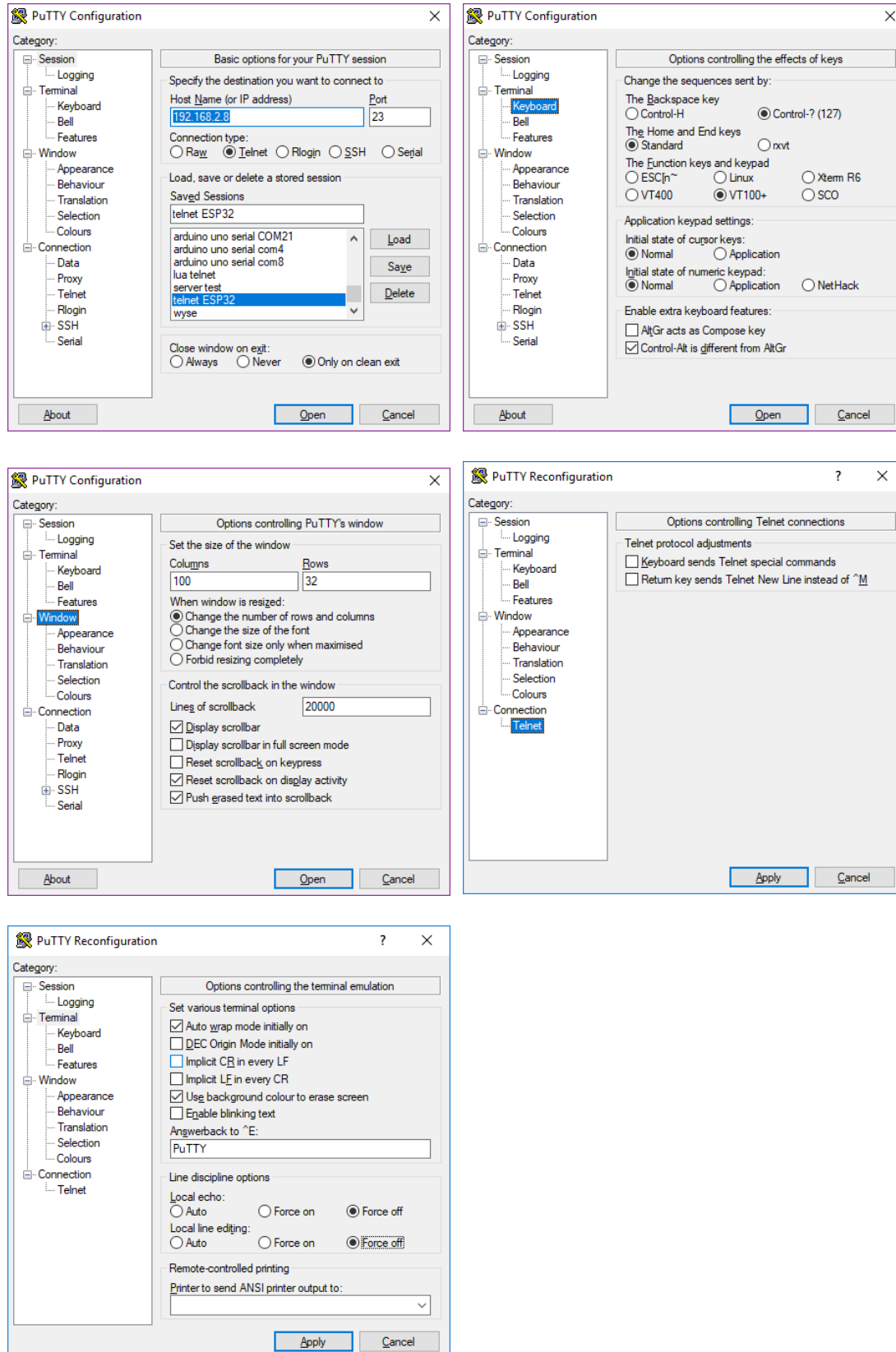
At least the RKA0 partition must be filled by a proper image in order to run OS/8 on the simulator. The image can be downloaded from the Internet through the build-in http client or loaded from an SD card.

A fully patch RKA0/RKB0 image will be available in the Github data directory, ready to be transferred to an SD card.

4 Managing the simulator.

4.1 TELNET client.

On (re)start of the ESP32, the simulator does not start automatically. The user must make a TELNET connection to the ESP32. A good TELNET client for this purpose is “PuTTY” (see <http://putty.org>). Assuming an IP address of 192.168.2.8, the settings for a connection are:

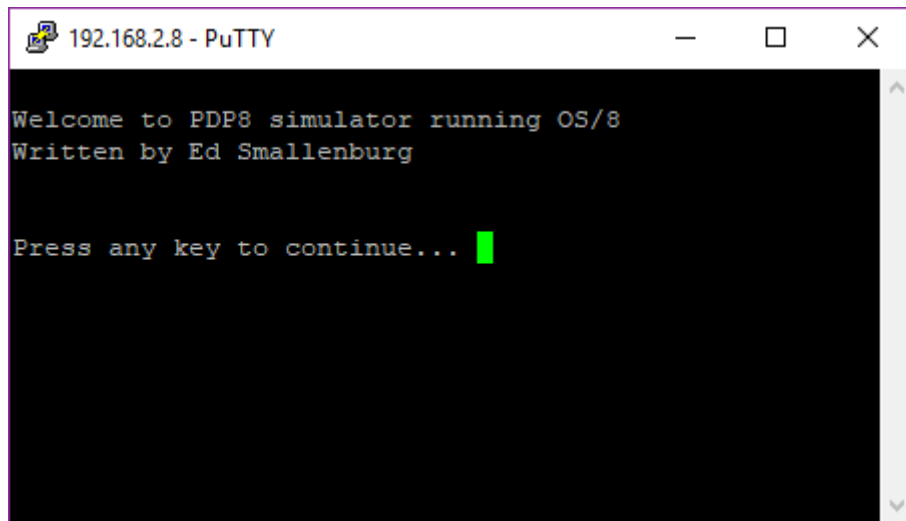


You may save the settings for convenience.

4.2 Control Console.

Connect to the simulator through the build-in Control Console.

Once connected, you will see a screen like this:



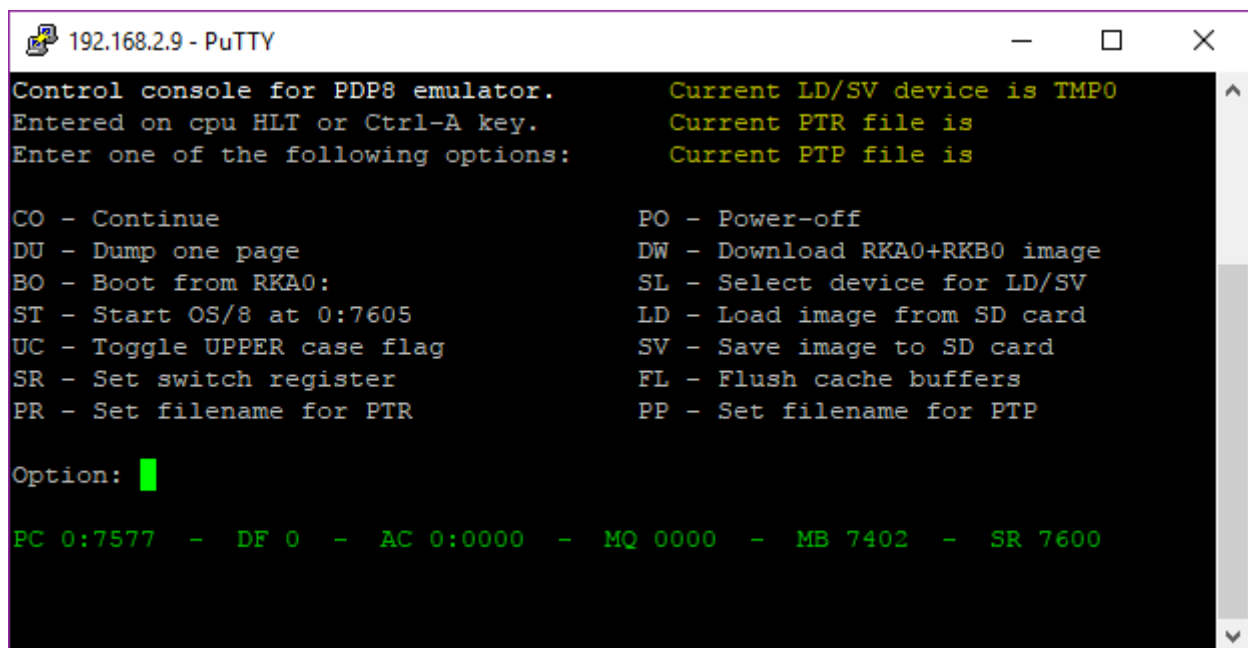
```
192.168.2.8 - PuTTY

Welcome to PDP8 simulator running OS/8
Written by Ed Smallenburg

Press any key to continue... █
```

An extra line is added if an SD card is connected.

After pressing a key, the simulator console will be shown:



```
192.168.2.9 - PuTTY

Control console for PDP8 emulator.      Current LD/SV device is TMP0
Entered on cpu HLT or Ctrl-A key.      Current PTR file is
Enter one of the following options:     Current PTP file is

CO - Continue                          PO - Power-off
DU - Dump one page                     DW - Download RKA0+RKBO image
BO - Boot from RKA0:                   SL - Select device for LD/SV
ST - Start OS/8 at 0:7605               LD - Load image from SD card
UC - Toggle UPPER case flag            SV - Save image to SD card
SR - Set switch register                FL - Flush cache buffers
PR - Set filename for PTR               PP - Set filename for PTP

Option: █

PC 0:7577 - DF 0 - AC 0:0000 - MQ 0000 - MB 7402 - SR 7600
```

This screen can be activated any time by pressing Ctrl-A on the keyboard. A HLT instruction or an illegal IOT will also bring up this screen.

You may adjust the size of the console screen if filenames are too long to fit the screen.

The most obvious command is “BO”, but for the first time you have to configure your PDP8. Possible commands are listed below. Command can have one or two parameters.

- CO – This will continue the PDP8 after a Ctrl-A interrupt.
- DU – This will dump the contents of one PDP8 page on the screen. Parameters may be the data field and the start address of the dump. If no parameters are supplied, the next page will be dumped. Examples: “DU 3 200”, “DU 1000”, “DU”.
- BO – This will boot the RKA0: disk and start OS/8.
- ST – This will start OS/8 at 07605 after a Ctrl-A interrupt. Parameters may be field and address of a different start point. Examples: “ST”, “ST 0 7600”, “ST 0200”.

- UC – This will toggle the Uppercase flag. Some versions of OS/8 expect only uppercase characters. This function may be handy for this situation.
- SR – Will set the pdp8 switch register according the parameter. The “LAS” instruction will read this number in the accumulator.
Example: “SR 7600”
- PR – Set the input file (on SD card) for the simulated paper tape reader. The current input file will be visible at the top of the screen.
- PO – Power off. This will write cached data to the simulated drives and the ESP32 is set to sleep. Wake up by the RESET or BOOT button of the development module. Not that disconnecting the power without this command does not write the cache to the simulated disks/tapes, and may therefore cause file inconsistency.
- DW – This will download a more or less working OS/8 system from the internet at **www.pdp8online.com/ftp/images/os8/diag-games-kermit.rk05**. You may also supply a different URL as a parameter for this command. Other images are available if an SD card (with the right data on it) is connected.
- SL – This will select a pdp8 device for the LD and SV commands. If no parameter is supplied, a list of the available devices will be shown.
- LD – This will copy an image from the SD-card to the selected pdp8 device. For RKA0/RKB0, the image is usually the size of one or two disks. The reason is that the disk was divided into 2 “sides” as the whole disk was too big to be addressed as one device (more than 4096 blocks). Without a parameter, the console will show all the matching (see SL command) images on the SD card like this:
os8.rk05
diag-games-kermit.rk05
To load an image, select the right input file according to the directory listing.
Example: “LD os8.rk05”. The extension of the file will be forced to the standard extension, like “.rk05” for a RKA0/RKB0 image.
Note: a load from existing .rx01 files does not work yet. However, you may save an rx01 image and read it back using LD.
- SV – Save the image of the selected device to the SD card. This is the reverse of “LD”. You have to specify a filename. The extension will be forced to a standard extension.
- FL – This will flush the cached data to the pdp8 devices. Never turn the power off without a flush! You may also use the “PO” command.
- PP – Set the output file (on SD card) for the simulated paper tape punch. Specify the filename in the parameter. The current output file is visible at the top of the screen.

5 Running OS/8.

5.1 Unpatched OS/8 images.

An unpatched RK8E OS/8 image will run on the simulator. But only RKA0 is available in this case. You can make RKB0 accessible by making a simple patch. This is the procedure:

1. Find out what the device number is for RKB0. Use RESORC.SV for this purpose:

```
.RESORC /E
```

This will show a table like:

#	NAME	TYPE	MODE	SIZE	BLK	KIND	U	V	ENT	USER
01	SYS	RK8E	RWF	3248	SYS		0	C	07	
02	DSK	RK8E	RWF	3248	SYS		0	C	07	
03	RKA0	RK8E	RWF	3248	16	RK05	0	A	20	
04	RKB0	RK8E	RWF	3248	16	RK05	0	A	21	
05	RKA1	RK8E	RWF	3248	16	RK05	1	A	22	
06	RKB1	RK8E	RWF	3248	16	RK05	1	A	23	
07	RXA0	RX8E	RWF	494	17			E	30	

2. Look at the device number (column 1) for device RKB0. In this example it is 4.
3. Start ODT and open location 17646 + DEVNR, in this example 17652 and change its contents to 7613. Exit ODT afterwards by pressing Ctrl-C:

```
. ODT
17652/0000 7613
.^C
.
```

4. Now RKB0 is accessible.
5. Optional, you can make DTA0 accessible by putting the number 7617 into 17646+(devnr of DTA0).

5.2 Patched OS/8 images.

In the data directory is a fully patched OS/8 image available (os8.rk05).

5.3 Some handy commands:

- SET TTY WIDTH 80
This prevents extra new lines to be printed if lines are longer than 72 characters.
- SET TTY NO PAUSE
This will prevent pauses if text scrolls over the screen. Will not work with all OS/8 versions.
- DIR
This will give you a directory listing. Also “DIR DTA0:” or “DIR RKB0:” Note that device names end with a semicolon to distinguish if there would be confused with a filename.
- HELP
This will give you some help texts.

5.4 OS/8 programs incompatibility.

Some programs will not run on this simulator. Specially programs that uses I/O directly. Direct I/O for extended memory handling and teletype I/O are emulated, but if a program tries to execute IOTs for disk or dectape access directly, the simulator will stop with information about the failed instruction.

So for example DTCOPY.SV will not work. Programs that use the interrupt facility will also not work.

A PATCH.PA program is supplied in the tools directory that will patch CCL, DIRECT, RESORC and PIP for this simulator.

5.5 CCL.SV.

The OS/8 date was originally designed to run from 1-JAN-70 to 31-DEC-77. This limit was caused by having just 3 bits to store the year. Here the year offset was 1970. Later on, the limit of the year was extended to 1999 by using 2 additional bits at location 07777, bits 3 and 4. The simulator tries to set the date automatically on boot, at location 17666 and 07777. For 2017, this will result in something like Sunday August 23, 1987.

So we have to change the offset for the year from 1970 to 2000. I patched CCL.SV for this purpose. Now, since a long time, it prints the correct date:

```
. DATE
Wednesday August 23, 2017
```

Note that setting the date is no longer necessary and maybe impossible.
I also patched DIRECT.SV so it will print the right date:

```
.DIR /C

23-Aug-17

A      .      1 23-Aug-17  PATCH .PA      1 23-Aug-17  PATCH .BN      2 23-Aug-17

      3 Files in      4 Blocks - 522 Free blocks
```

5.6 PIP.SV.

The /Z option of PIP.SV does not work for non-standard devices like TMP0. So we have to patch PIP.SV. The table resides at 13600. Patch the following location:

13640 / 0000 7530 (TMP0 has 168 blocks, 250 octal blocks, -250 = 7530)
13625 / 0000 7022 (RX01 has 756 octal blocks)

5.7 RESORC.SV.

The name and the length of TMP0 must be set in RESORC.SV. The table resides at 02000 and has 4 word entries. Patch the following locations:

02200 / 0000 2415 (First 2 characters of "TMP0")
02201 / 0000 2060 (Last 2 characters of "TMP0")
02202 / 0000 7602 (Minus size in blocks)

5.8 F4.SV / FRTS.SV.

The Fortran compiler seems to work. However I was not able to run the famous ADVENTURE program. Compiling the source code succeeded, but there was an error on running the program.

5.9 BASIC.SV

The RUN command in basic does not work. But you may run your basic programs by:

```
.COMPILE SPACWR.BA
```

5.10 Programs that run normally.

PAL8.SV	ABSLDR.SV	TECO.SV
EDIT.SV	FUTIL.SV	FORT.SV
SABR.SV	LOADER.SV	

6 Fake device handlers.

6.1 Block drivers.

The OS/8 block drivers for the simulated devices like RK8E, DTA0,... are not emulated by their IOTs. Instead their functions (read and write) are simulated. The IOTs 6770, 6771, 6772, 6773, 6774 and 6775 are used for this purpose. A special device handler has been made that can be used in “BUILD”. It forms a “fake” handler for 7 devices: SYS, RKA0, RKB0, DTA0, DTA1, RXA0 and TMP0. Below is the listing of the source code of this handler.

```
/ FAKE.PA
/
/ FAKE HANDLER FOR PDP8 SIMULATOR
/ ED SMALLENBURG, 29-AUG-2017
/
    VERSION="B&77
    *0

    DECIMAL;RKLEN=3248;OCTAL
    DECIMAL;DTALEN=737;OCTAL
    DECIMAL;FLPLEN=494;OCTAL
    DECIMAL;TMPLLEN=168;OCTAL

    -7          /7 HANDLERS: SYS,RKA0,RKB0,DTA0,DTA1,RXA0,TMP0
    DEVICE FAKE;DEVICE SYS ;4231;2007;0;RKLEN
    DEVICE FAKE;DEVICE RKA0;4231;1007;0;RKLEN
    DEVICE FAKE;DEVICE RKB0;4231;RKB0H&177+1000;0;RKLEN
    DEVICE FAKE;DEVICE DTA0;4161;DTA0H&177+1000;0;DTALEN
    DEVICE FAKE;DEVICE DTA1;4161;DTA1H&177+1000;0;DTALEN
    DEVICE FAKE;DEVICE RXA0;4251;RXA0H&177+1000;0;FLPLEN
    DEVICE FAKE;DEVICE TMP0;4401;TMP0H&177+1000;0;TMPLLEN
    BOOT-BLAST
    RELOC 0

BOOT,    TAD I BOOTX1
        DCA I BOOTX2
        TAD I BOOTX3
        CDF 10
        DCA I BOOTX4
        CDF 0
        TAD BOOTX2
        SZA CLA
        JMP BOOT
        JMP I B7605
BOOTX1,  177
BOOTX2,  7577
BOOTX3,  46
BOOTX4,  7646
        ZBLOCK 30-.
B7605,   7605
BLAST,   RELOC

        *200
        RELOC 7600

        ZBLOCK 7
SHNDLR,  VERSION          /SYSTEM AND RKA0: ENTRYPPOINT
        CLA CLL           /GUARD AGAINST NON-ZERO AC
        TAD SHNDLR        /POINTER TO PARAMETERS
        6770              /SIMULATES RKA0: READ/WRITE
/NO RETURN HERE, SIMULATOR WILL RETURN TO CALLERS RETURN ADDRESS

RKB0H,   VERSION          /RKB0: ENTRYPPOINT
        CLA CLL           /GUARD AGAINST NON-ZERO AC
        TAD RKB0H        /POINTER TO PARAMETERS
        6771              /SIMULATES RKB0: READ/WRITE
/NO RETURN HERE

DTA0H,   VERSION          /DTA0: ENTRYPPOINT
        CLA CLL           /GUARD AGAINST NON-ZERO AC
        TAD DTA0H        /POINTER TO PARAMETERS
        6772              /SIMULATES DTA0: READ/WRITE
/NO RETURN HERE

DTA1H,   VERSION          /DTA1: ENTRYPPOINT
        CLA CLL           /GUARD AGAINST NON-ZERO AC
        TAD DTA1H        /POINTER TO PARAMETERS
        6773              /SIMULATES DTA1: READ/WRITE
/NO RETURN HERE

RXA0H,   VERSION          /RXA0: ENTRYPPOINT
        CLA CLL           /GUARD AGAINST NON-ZERO AC
        TAD RXA0H        /POINTER TO PARAMETERS
        6774              /SIMULATES RXA0: READ/WRITE
/NO RETURN HERE

TMP0H,   VERSION          /TMP0: ENTRYPPOINT
        CLA CLL           /GUARD AGAINST NON-ZERO AC
        TAD TMP0H        /POINTER TO PARAMETERS
        6775              /SIMULATES TMP0: READ/WRITE
/NO RETURN HERE
    RELOC
    $
```

The configuration after “BUILD” looks like this:

```
.R RESORC
*/E$
171 FILES IN 2680 BLOCKS USING 5 SEGMENTS
512 FREE BLOCKS (10 EMPTYIES)

#  NAME TYPE MODE SIZ BLK KIND  U V ENT USER
01 SYS  RK8E RWF 3248 SYS      0 B  07
02 DSK  RK8E RWF 3248 SYS      0 B  07
03 TTY   TTY  RW   16+ KL8E   E 176
04 RKA0 RK8E RWF 3248 SYS      0 B  07
05 RKB0 RK8E RWF 3248 SYS      1 B  13
06 DTA0 TC08 RWF  737 SYS      1 B  17
07 DTA1 TC08 RWF  737 SYS      1 B  23
10 RXA0 RX8E RWF  494 SYS      1 B  27
11 RL21  40  RWF      SYS      1 B  33

FREE DEVICE SLOTS: 06,  FREE BLOCK SLOTS: 07
OS/8 V3T
```

Note that the last entry (“RL21”) is in fact “TMP0”. OS/8 cannot properly decode device names that are packed into 12 bits. Therefore “RL21” is the same as “TMP0” and “RKB0” is the same as “QJC1”.

6.2 Fake PTR.

A paper tape reader (PTR) can be very handy to load source code to a pdp8 file. Also binary files (with the “.bn” extension) can be read from paper tape.

The simulator has the possibility to connect a file on SD card to the virtual paper tape reader. The IOTs for the paper tape reader read from this file. At end of file, a Ctrl-Z character will be the result of the RRB instruction. The filename for the simulated PTR can be supplied in the PR command. If the filename is omitted, the directory of the SD card is presented.

In the simulator a file can be read, for example, like:

```
.R PIP
*XXX.PA<PTR:
```

6.3 Fake PTP.

The PTP: handler is simulated to redirect the output to a SD card file. The name for the file must be specified in the control console. This makes it possible to get source files (or binaries) out of the simulator.

Output to paper can be initiated like this:

```
.R PIP
*PTP:<XXX.PA
```

7 More information.

7.1 The best website for software and manuals is:

<http://www.vandermark.ch/pdp8>

7.2 pdp8 online home page.

<http://www.pdp8online.com>

8 Logging start-up.

This will be logged on the serial output during start-up.

ets Jun 8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)

ets Jun 8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)

configsip: 0, SPIWP:0xee

clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00

mode:DIO, clock div:2

load:0x3fff0010,len:4

load:0x3fff0014,len:5424

ho 0 tail 12 room 4

load:0x40078000,len:0

load:0x40078000,len:12028

entry 0x40078f24

I (48) boot: ESP-IDF v3.0-dev-265-g969f1bb9 2nd stage bootloader

I (48) boot: compile time 14:06:11

I (48) boot: Enabling RNG early entropy source...

I (66) boot: SPI Speed : 40MHz

I (79) boot: SPI Mode : DIO

I (91) boot: SPI Flash Size : 4MB

I (104) boot: Partition Table:

I (115) boot: ##	Label	Usage	Type	ST	Offset	Length
I (138) boot: 0	nvs	WiFi data	01	02	00009000	00006000
I (161) boot: 1	phy_init	RF data	01	01	0000f000	00001000
I (184) boot: 2	factory	factory app	00	00	00010000	00090000
I (207) boot: 3	RKA0	Unknown data	01	80	000a0000	00140000
I (231) boot: 4	RKB0	Unknown data	01	80	001e0000	00140000
I (254) boot: 5	DTA0	Unknown data	01	81	00320000	0004c000
I (277) boot: 6	DTA1	Unknown data	01	81	0036c000	0004c000
I (300) boot: 7	RXA0	Unknown data	01	82	003b8000	00034000
I (324) boot: 8	TMP0	Unknown data	01	83	003ec000	00014000

I (347) boot: End of partition table

I (360) esp_image: segment 0: paddr=0x00010020 vaddr=0x3f400020 size=0x16be8 (93160) map

I (487) esp_image: segment 1: paddr=0x00026c10 vaddr=0x3ffb0000 size=0x02f98 (12184) load

I (503) esp_image: segment 2: paddr=0x00029bb0 vaddr=0x40080000 size=0x00400 (1024) load

0x40080000: _iram_start at ????

I (509) esp_image: segment 3: paddr=0x00029fb8 vaddr=0x40080400 size=0x06058 (24664) load

I (567) esp_image: segment 4: paddr=0x00030018 vaddr=0x400d0018 size=0x5f4a4 (390308) map

0x400d0018: _stext at ????

I (983) esp_image: segment 5: paddr=0x0008f4c4 vaddr=0x40086458 size=0x0df44 (57156) load

0x40086458: xQueueGiveFromISR at C:/msys32/home/ed/esp/esp-idf/components/freertos/queue.c:2034

I (1057) esp_image: segment 6: paddr=0x0009d410 vaddr=0x400c0000 size=0x00064 (100) load

I (1094) boot: Loaded app from partition at offset 0x10000

I (1094) boot: Disabling RNG early entropy source...

I (1097) cpu_start: Pro cpu up.

I (1107) cpu_start: Starting app cpu, entry point is 0x40080f74

0x40080f74: call_start_cpu1 at C:/msys32/home/ed/esp/esp-idf/components/esp32/cpu_start.c:192

I (0) cpu_start: App cpu up.

I (1140) heap_init: Initializing. RAM available for dynamic allocation:

I (1161) heap_init: At 3FFAE2A0 len 00001D60 (7 KiB): DRAM

I (1180) heap_init: At 3FFC9A08 len 000165F8 (89 KiB): DRAM

I (1199) heap_init: At 3FFE0440 len 00003BC0 (14 KiB): D/IRAM

I (1219) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM

I (1239) heap_init: At 4009439C len 0000BC64 (47 KiB): IRAM

I (1258) cpu_start: Pro cpu start user code

I (1317) cpu_start: Starting scheduler on PRO CPU.

I (193) cpu_start: Starting scheduler on APP CPU.

I (193) gpio: GPIO[5] InputEn: 0 OutputEn: 1 OpenDrain: 0 Pullup: 0 Pulldown: 0 Intr:0

I (343) wifi: wifi firmware version: c1b8a2f

I (343) wifi: config NVS flash: enabled

I (343) wifi: config nano formatting: disabled

I (343) system_api: Base MAC address is not set, read default base MAC address from BLK0 of EFUSE

I (353) system_api: Base MAC address is not set, read default base MAC address from BLK0 of EFUSE

I (383) wifi: Init dynamic tx buffer num: 32

I (383) wifi: Init data frame dynamic rx buffer num: 32

I (383) wifi: Init management frame dynamic rx buffer num: 32

I (383) wifi: wifi driver task: 3fffd42f0, prio:23, stack:4096

I (393) wifi: Init static rx buffer num: 10

I (393) wifi: Init dynamic rx buffer num: 32

I (403) wifi: Init rx ampdu len mblock:7

I (403) wifi: Init lldesc rx ampdu entry mblock:4

I (403) wifi: power manager task: 0x3fffd9654 prio: 21 stack: 2560

I (413) [PDP8]: Setting WiFi configuration SSID NETGEAR-11...

I (423) wifi: wifi timer task: 3ffda6c0, prio:22, stack:3584

I (453) phy: phy_version: 357.0, a6bf95b, Jul 25 2017, 12:28:06, 0, 0

I (453) wifi: Init Ampdu: 1 tx baw=6 rx baw=6

```
I (453) wifi: mode : sta (24:0a:c4:82:ee:58)
I (1183) wifi: n:6 0, o:1 0, ap:255 255, sta:6 0, prof:1
I (2163) wifi: state: init -> auth (b0)
I (2163) wifi: state: auth -> assoc (0)
I (2173) wifi: state: assoc -> run (10)
I (2183) wifi: connected with NETGEAR-11, channel 6
I (4733) event: ip: 192.168.2.9, mask: 255.255.255.0, gw: 192.168.2.254
I (4733) [PDP8]: Connected to AP
I (4743) [PDP8]: 0 - RKA0 mounted, size 13A000 bytes (3297 OS/8 blocks) 3248 blocks used
I (4773) [PDP8]: 1 - RKB0 mounted, size 13A000 bytes (3297 OS/8 blocks) 3248 blocks used
I (4783) [PDP8]: 2 - DTA0 mounted, size 048000 bytes ( 756 OS/8 blocks) 737 blocks used
I (4783) [PDP8]: 3 - DTA1 mounted, size 048000 bytes ( 756 OS/8 blocks) 737 blocks used
I (4793) [PDP8]: 4 - RXA0 mounted, size 030000 bytes ( 504 OS/8 blocks) 494 blocks used
I (4793) [PDP8]: 5 - TMP0 mounted, size 010000 bytes ( 168 OS/8 blocks) 168 blocks used
I (4803) [PDP8]: SD card found, ready for use
I (4813) [PDP8]: Starting PDP8 Emulator task and telnet server
I (6813) [PDP8]: Initializing SNTP
I (6813) [PDP8]: Waiting for system time to be set... (1/10)
I (9813) [PDP8]: Waiting for system time to be set... (2/10)
I (12173) wifi: pm start, type:0

I (12173) wifi: active cnt: 19
I (12813) [PDP8]: Waiting for system time to be set... (3/10)
I (14813) [PDP8]: Free stack space telnet task is 2200 words
I (14813) [PDP8]: Free stack space simulator task is 1388 words
I (14813) [PDP8]: Free stack space main task is 3148 words
I (14813) [PDP8]: Free heap space (in bytes)is 53144
I (15813) [PDP8]: Time is set to 11-10-2017 - 14:23:33
```