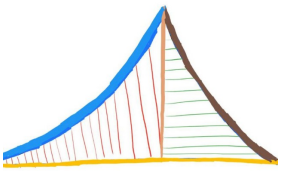
Embedd Systems Through Vaman



G. V. V. Sharma

First manual appeared in 2015

Contents

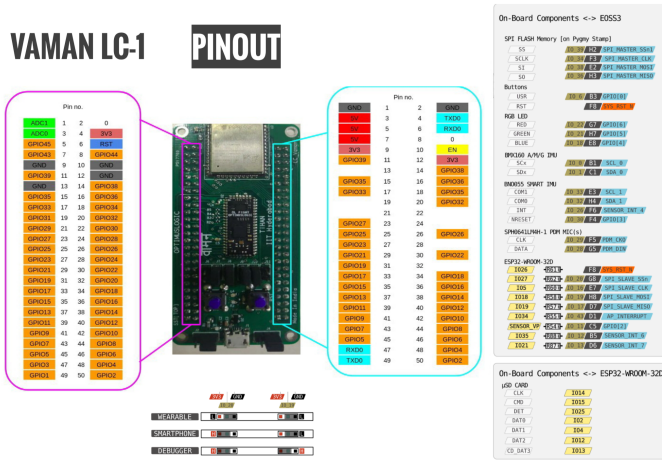


Fig. 1.1: Vaman pins. Right side reperesents ESP32 and left side M4-FPGA (Pygmy).

1 Blink

1.1 ESP

Here we show how to program the ESP32 on the Vaman using the Arduino framework.

1. Make sure that Vaman board does not power any devices.
2. Make connections as shown in Table 1.1.
3. The Vaman pin diagram is available in Figure 1.1.

VAMAN-ESP	UART PINS
5V	5V
GND	GND
TXD0	TXD
RXD0	RXD
0	GND

TABLE 1.1

4. Connect the UART to raspberry pi through USB.
5. Connect the pins between Vaman-ESP32 and Vaman-PYGMY as per Table 1.2

ESP32	**Vaman**
GPIO2	GPIO18

TABLE 1.2

6. On termux on your phone,

```
cd vaman/esp32/codes/ide/blink
pio run
```

7. Transfer the ini and bin files to the rpi

```
scp platformio.ini pi@192.168.50.252:./hi/platformio.ini

scp .pio/build/esp32doit-devkit-v1/firmware.bin
    pi@192.168.50.252:./hi/.pio/build/esp32doit-devkit
    -v1/firmware.bin
```

8. On rpi,

```
cd /home/pi/hi
pio run -t nobuild -t upload
```

You should see the blue led blinking.

9. Now disconnect pin 2 from pin 18 and connect to pygmy GPIO pin 21.
10. Repeat the above exercise using GPIO pin 22.
11. On your phone, open

```
src/main.cpp
```

and change the delay to

```
delay(100);
```

and execute the code by following the steps above.

12. Flash the following code.

```
vaman/esp32/codes/ide/ota/setup
```

after entering your wifi username and password (in quotes below)

```
#define STASSID "..." // Add your network credentials
#define STAPSK "..."
```

in src/main.cpp file

13. You should be able to find the ip address of your vaman-esp using

```
ifconfig
nmap -sn 192.168.231.1/24
```

where your computer's ip address is the output of ifconfig and given by 192.168.231.x

14. Assuming that the username is gvv and password is abcd, flash the following code wirelessly

```
vaman/esp32/codes/ide/ota/blink
```

through

```
pio run
pio run -t nobuild -t upload --upload-port
    192.168.231.245
```

where you may replace the above ip address with the ip address of your vaman-esp.

15. Flash the following code OTA

```
vaman/esp32/codes/ide/ota/blinkt
```

You should see the onboard LEDs blinking.

16. Change the blink duration to 100 ms.

1.2 FPGA

We show how to program the Vaman FPGA/microcontroller board.

1. Follow the instructions available in the video at

```
https://github.com/whyakari/TermuxDisableProcces?tab=
    readme-ov-file
```

to ensure that termux is not killed during the following installation process.

2. On termux-debian,

```
wget https://raw.githubusercontent.com/gadepall/fwc-1/
    main/scripts/setup.sh
bash setup.sh
```

3. Login to termux-debian on the android device and execute the following commands

```
cd vaman/fpga/setup/codes/blink
source ~/.vamenv/bin/activate
ql_symbiflow -compile -src vaman/fpga/setup/codes/
    blink -d ql-eos-s3 -P PU64 -v helloworldfpga.v
     -t helloworldfpga -p pygmy.pcf -dump binary
scp blink/helloworldfpga.bin pi@192.168.0.114:
```

Make sure that the appropriate IP address for the raspberry pi is given in the above command.

4. Put the Vaman board in download mode. For this, you need to first press the button to the right of the usb port and immediately press the button to the left. The green led should now flash and you can go to the next step.
5. Now execute the following commands on the raspberry pi.

```
python3 -m venv ~/.vamenv
source ~/.vamenv/bin/activate
git clone --recursive https://github.com/QuickLogic-
    Corp/TinyFPGA-Programmer-Application.git
pip3 install tinyfpgab
deactivate
sudo reboot
source ~/.vamenv/bin/activate
python3 TinyFPGA-Programmer-Application/tinyfpga
    -programmer-gui.py --port /dev/ttyACM0 --
    appfpga /home/pi/helloworldfpga.bin --mode fpga
     --reset
```

6. Make sure that the correct USB port address is given in the above command. After some time, the LED will start blinking red.
7. Replace the following line in the code in instruciton 10

```
assign redled = led; //If you want to change led colour
    to red,
```

with

```
assign blueled = led;
```

and execute the code.

8. In the following .pcf file,

```
codes/blink/pygmy.pcf
```

the pin numbers for the 3 colour-leds are defined. See Table 1.2 and Figure 1.2. The IO locations in Figure 1.2 can be found in pygmy.pcf while the aliases (GPIO) are printed on the board.

9. Now modify the helloworldfpga.v file to get the green led blinking.
10. In the following verilog program,

```
codes/blink/helloworldfpga.v
```

pay attention to the following lines

```
delay = delay+1;
if(delay > 20000000)
begin
delay=25'b0;
led=!led;
end
```

It may be deduced from the above that the blink frequency is 20 MHz.

11. In instruction 10, replace

```
if(delay > 20000000)
```

with

```
if(delay==25'b1001100010010110100000000)
```

and execute the verilog code.

12. Since the delay is 20 MHz, the blink period is 1 second. Modify the verilog code so that the blink period becomes 0.5s.
13. Find the bit length of 20 MHz.
Solution:

$$\log_2(20000000) \approx 25 \tag{1.1}$$

14. Obtain the above answer using a Python code.
Solution: Execute the following code and compare with instruction 11.

```
codes/blink/freq_count.py
```

15. Ensure that the LED stays on in green colour.
Solution: Execute the following code

```
vaman/setup/codes/blink/onoff.v
```

16. Execute the following code and make pin connections as per Table 1.3. Take out the input pin connect to 3.3V. Plug it again. Do this repeatedly.

Type	Vaman Pin	Connection
Input	IO_28	3.3V

TABLE 1.3: Vaman Input/Output.

Type	Pin	Destination
Input	IO_5	5V

TABLE 1.4: Vaman control through external input.

```
vaman/setup/codes/input/blink_ip.v
vaman/setup/codes/input/pygmy.pcf
```

17. Connect an external LED and repeat.

1.3 ARM

We show how to control an LED using the M4 on Vaman.

1. Check your path

```
cd vaman/arm/setup/blink/GCC_Project
nvim config.mk
```

and

2. and modify so that you have the following lines

```
#export PROJ_ROOT=/data/data/com.termux/files/home
    /pygmy-dev/pygmy-sdk
export PROJ_ROOT=/root/pygmy-dev/pygmy-sdk
```

3. Now execute

```
cd vaman/arm/setup/blink/GCC_Project
make -j4
scp output/bin/blink.bin pi@192.168.0.114:
```

Appropriately modify the above ip address before sending blink.bin to the pi.

4. Now log on to the RPi and execute the following

```
sudo python3 /home/pi/Vaman-dev/Vaman-sdk/
    TinyFPGA-Programmer-Application/tinyfpga-
    programmer-gui.py --port /dev/ttyACM0 --
    m4app blink.bin --mode m4-fpga
```

5. Enter the appropriate USB device port above while executing. Press the button to the right after the above command is successfully executed. The LED will start blinking.
6. See the following lines of the code below

```
codes/setup/blink/src/main.c
```

```
    PyHal_Set_GPIO(18,1);//blue
    PyHal_Set_GPIO(21,1);//green
    PyHal_Set_GPIO(22,1);//red
        HAL_DelayUSec(2000000);
    PyHal_Set_GPIO(18,0);
    PyHal_Set_GPIO(21,0);
    PyHal_Set_GPIO(22,0);
```

We may conclude that the blink delay is 2000 000us = 2 s.

7. Replace the following line in 6

```
        HAL_DelayUSec(2000000);
```

with

```
        HAL_DelayUSec(1000000);
```

and execute. Can you see any difference in the blink period?

8. To obtain red colour, execute the following code.

```
vaman/arm/codes/setup/red/src/main.c
```

Now obtain blue colour.

9. Now obtain green colour without blink.
Solution: Execute the following code.

```
vaman/arm/codes/setup/onoff/src/main.c
```

10. Using Table 1.4 and Fig. 1.1, use an input pin to control the onboard LED.
Solution: Execute the following code. You should see the green LED on. Connecting IO_5 to GND will turn the green LED off.

```
vaman/arm/codes/setup/gpio/src/main.c
```

Arduino	2	3	4	5	6	7	8
Display	a	b	c	d	e	f	g

TABLE 2.1

2 Seven Segment Display

2.1 ESP

1. Make connections according to Table 2.1
2. Execute the following code

```
vaman/esp32/sevenseg/codes/sevenseg/sevenseg.cpp
```

3. Now generate the numbers 0-9 by modifying the above program.

PD64		
IO Locatic	Alias	IO Type
B1	IO_0	BIDIR
C1	IO_1	BIDIR
A1	IO_2	BIDIR
A2	IO_3	BIDIR
B2	IO_4	BIDIR
C3	IO_5	BIDIR
B3	IO_6	BIDIR
A3	IO_7	BIDIR/CLOCK
C4	IO_8	BIDIR/CLOCK
B4	IO_9	BIDIR
A4	IO_10	BIDIR
C5	IO_11	BIDIR
B5	IO_12	BIDIR
D6	IO_13	BIDIR
A5	IO_14	BIDIR
C6	IO_15	BIDIR
E7	IO_16	BIDIR
D7	IO_17	BIDIR
E8	IO_18	BIDIR
H8	IO_19	BIDIR
G8	IO_20	BIDIR
H7	IO_21	BIDIR
G7	IO_22	BIDIR/CLOCK
H6	IO_23	BIDIR/CLOCK
G6	IO_24	BIDIR/CLOCK
F7	IO_25	BIDIR
F6	IO_26	BIDIR
H5	IO_27	BIDIR
G5	IO_28	BIDIR
F5	IO_29	BIDIR
F4	IO_30	BIDIR
G4	IO_31	BIDIR
H4	IO_32	SDIOMUX
E3	IO_33	SDIOMUX
F3	IO_34	SDIOMUX
F2	IO_35	SDIOMUX
H3	IO_36	SDIOMUX
G2	IO_37	SDIOMUX
E2	IO_38	SDIOMUX
H2	IO_39	SDIOMUX
D2	IO_40	SDIOMUX
F1	IO_41	SDIOMUX
H1	IO_42	SDIOMUX
D1	IO_43	SDIOMUX
E1	IO_44	SDIOMUX
G1	IO_45	SDIOMUX

PU64		
IO Locatic	Alias	IO type
4	IO_0	BIDIR
5	IO_1	BIDIR
6	IO_2	BIDIR
2	IO_3	BIDIR
3	IO_4	BIDIR
64	IO_5	BIDIR
62	IO_6	BIDIR
63	IO_7	BIDIR/CLOCK
61	IO_8	BIDIR/CLOCK
60	IO_9	BIDIR
59	IO_10	BIDIR
57	IO_11	BIDIR
56	IO_12	BIDIR
55	IO_13	BIDIR
54	IO_14	BIDIR
53	IO_15	BIDIR
40	IO_16	BIDIR
42	IO_17	BIDIR
38	IO_18	BIDIR
36	IO_19	BIDIR
37	IO_20	BIDIR
39	IO_21	BIDIR
34	IO_22	BIDIR/CLOCK
33	IO_23	BIDIR/CLOCK
32	IO_24	BIDIR/CLOCK
31	IO_25	BIDIR
30	IO_26	BIDIR
28	IO_27	BIDIR
27	IO_28	BIDIR
26	IO_29	BIDIR
25	IO_30	BIDIR
23	IO_31	BIDIR
22	IO_32	SDIOMUX
21	IO_33	SDIOMUX
20	IO_34	SDIOMUX
18	IO_35	SDIOMUX
17	IO_36	SDIOMUX
15	IO_37	SDIOMUX
16	IO_38	SDIOMUX
11	IO_39	SDIOMUX
13	IO_40	SDIOMUX
14	IO_41	SDIOMUX
10	IO_42	SDIOMUX
7	IO_43	SDIOMUX
8	IO_44	SDIOMUX
9	IO_45	SDIOMUX

WR42		
IO Locatic	Alias	IO Type
A7	IO_0	BIDIR
B7	IO_1	BIDIR
C7	IO_3	BIDIR
A6	IO_6	BIDIR
B6	IO_8	BIDIR/CLOCK
A5	IO_9	BIDIR
B5	IO_10	BIDIR
A4	IO_14	BIDIR
B4	IO_15	BIDIR
E1	IO_16	BIDIR
D1	IO_17	BIDIR
C1	IO_19	BIDIR
F2	IO_20	BIDIR
E2	IO_23	BIDIR/CLOCK
D2	IO_24	BIDIR/CLOCK
D3	IO_25	BIDIR
F3	IO_28	BIDIR
E3	IO_29	BIDIR
F4	IO_30	BIDIR
E4	IO_31	BIDIR
D5	IO_34	SDIOMUX
F5	IO_36	SDIOMUX
E6	IO_38	SDIOMUX
F6	IO_39	SDIOMUX
D7	IO_43	SDIOMUX
E7	IO_44	SDIOMUX
F7	IO_45	SDIOMUX

Fig. 1.2: Pin Definitions