Caravel M.V.PCB: TinyTapeout Variant

Sample of a minimum viable PCB for TinyTapeout ASICs on QFN.

This version has a symbol and net names to match TinyTapeout function assignments. Two options here:
1) TinyTapeout with Caravel CPU available
2) TinyTapeout stand-alone (no CPU)

If using the CPU, the orange boxes are needed (namely external flash memory) and a way to access the HK SPI to program it.
It is possible to go stand-alone, in which case some passives and configuration choices are in order (see green box).

Always REQUIRED elements are:
* power: regulated 3v3 and 1v8
* osc: a CMOS clock signal

REQUIRED	Always needed
TTSA REQ	Needed when using TT stand-alone
CPU REQ	Needed to use Caravel CPU
OPTIONAL	Customize as required

Power Supply

J1 — USB4105-GF-A-12

VBUS, CC1, CC2, D-, D-, D+, D+, SBU1, SBU2, SHIELD, GND

A4 VBUS, A5 CC1, B5 CC2, A7 D-, B7 D-, A6 D+, B6 D+, A8 SBU1, B8 SBU2, A1 GND

R6 5.1k, R7 5.1k
R1 1M, C3 6800pF
S1, J2

Populate R1/C3 according to your shielding preference/religion

Vin may be provided via USB or basic JST or 100mil header.

Any voltage suitable for top level LDO should do, say 3v3 <= Vin < 15V here.
Lower is more efficient, gens less heat.

Power_Regulation

VIN VDDIOD, VCCDD
+3V3, +1V8

Voltage regulators provide vdd* (3v3) and vcc* 1v8. These are staggered to allow for more efficiency on 1v8 regulation but should also allow backpowering via vdd/3v3

Debounced Switches

R2 10k, R3 5.1k, C4 1uF, SW1, B3U-1000P
SoC reset, active LOW
SOC_RST

SW2, B3U-1000P, C5 1uF, R8 10k, R11 5.1k
prj_io37
Sample active HIGH switch sent to GPIO37

Flash Memory

C1 100nF, R4, R5, 10k, 10k
U2 — W25Q32JVSS
CS, VCC, DI(IO0), DO(IO1), CLK, GND, IO2, IO3
flash_csb, flash_clk, flash_io0, flash_io1
R9 10k, R10 10k, R12 10k, R13 10k

Caravel code resides here. Flash may be access via the HK SPI in pass-through.

CMOS Clock Osc

C2 100nF, X1 Tri-State, GND VDD, OUT
osc_out, 10MHz
Std 4-pin 2.5x2.0mm osc
External clock needed to run CPU, and scan chain driver

U1 — Caravel_QFN_TT3

TinyTapeout 1,2,3 Caravel in QFN

Left side pins:
32 crvio[1]_SDO — HK_SDO
31 crvio[0]_JTAG — HK_JTAG
30 vdda
29 vssio
28 gpio
27 flash_io1
26 flash_io0
25 flash_clk
24 flash_csb
23 vssd
22 clock — osc_out
21 resetb — SOC_RST
20 vssa
19 N/C
18 vccd
17 vddio

Top pins:
33 crvio[2]_SDI — HK_SDI
34 crvio[3]_CSB — HK_CSB
35 crvio[4]_SCK — HK_SCK
36 crvio[5]_ser_rx — ser_rx
37 crvio[6]_ser_tx — ser_tx
38 vssa1
39 vdda1
40 unused
41 crvio[7]
42 driver_sel0/crvio[8] — SCS.DRVo
43 driver_sel1/crvio[9] — SCS.DRV1
44 slow_clk/crvio[10] — slowclk
45 set_clk_div/crvio[11] — setdiv
46 sel[0]/crvio[12] — sel0
47 sel[1]/crvio[13] — sel1
48 +3V3

Right side pins:
49 vccd1 — +1V8
50 sel[2]/crvio[14] — sel2
51 sel[3]/crvio[15] — sel3
52 vssa1 — GND
53 sel[4]/crvio[16] — sel4
54 sel[5]/crvio[17] — sel5
55 sel[6]/crvio[18] — sel6
56 vssio — GND
57 sel[7]/crvio[19] — sel7
58 sel[8]/crvio[20] — sel8
59 in0
60 in[1]/crvio[22] — in1
61 in[2]/crvio[23] — in2
62 in[3]/crvio[24] — in3
63 vccd2 — +1V8
64 vddio — +3V3

Bottom pins:
16 ready/crvio[37] — ready
15 out[7]/crvio[36] — out7
14 out[6]/crvio[35] — out6
13 out[5]/crvio[34] — out5
12 out[4]/crvio[33] — out4
11 out[3]/crvio[32] — out3
10 vssd2
9 vdda2 — +3V3
8 out[2]/crvio[31] — out2
7 out[1]/crvio[30] — out1
6 out[0]/crvio[29] — out0
5 in[7]/crvio[28] — in7
4 in[6]/crvio[27] — in6
3 in[5]/crvio[26] — in5
2 in[4]/crvio[25] — in4
1 vssa2
65 ThermalPad

* I/O pin names: TT / caravel

out bus: 8 outputs from project
in bus: 8 inputs sent to project
sel bus: active project selection

out[0..7]
in[0..7]
sel[0..8]
SCS{DRV[0..1] slowclk setdiv}

I/O headers

J5 — 613012243121 (PMOD Host)
1 IO1 out0, 2 IO5 out4, 3 IO2 out1, 4 IO6 out5, 5 IO3 out2, 7 IO4 out3, 8 IO8 out7, 9 IO7 out6, GND, VCC
+3V3, GND
out[0..7]

J6
sel0, sel1, sel2, sel3, sel4, sel5, sel6, sel7, sel8
in0, in1, in2, in3, in4, in5, in6, in7
ser_tx, ser_rx, ready
SCS.DRVo, SCS.DRV1, SCS.slowclk, SCS.setdiv
GND, +3V3

Sample I/O headers give access to all gpio pins from mprj_io[7] to mprj_io[37], on either the PMOD or the 2x18. Modify as required.

Housekeeping SPI

J3
HK_SDI 8, HK_SCK 6, HK_CSB 4, HK_JTAG 2
HK_SDO 7
+3V3
prj_io 0 to 4, special purpose, use to program flash
GND

TT SA Config

Stand-alone Support

in[0..7]
sel[0..8]
SCS{DRV[0..1] slowclk setdiv}
File: ttsa_config.kicad_sch

Power Good and GPIO LED

+3V3
D1 LED, R15 510
gpio, D2 LED, R14 510
GND

Caravel Serial

J4
SOC_RST, ser_tx, ser_rx, +3V3
C15 100nF
prj_io 5 and 6 uart rx/tx on standardish FTDI adapter 6-pin header

Bypass Caps

+1V8
C6 100nF, C7 100nF, C8 100nF
Layout near vccd* on ASIC QFN

+3V3
C9 100nF, C10 100nF, C11 100nF, C12 100nF, C13 100nF, C14 100nF
Layout near vdd* on ASIC QFN
GND

Fids for PnP
FID1 Fiducial
FID2 Fiducial
FID3 Fiducial

(C) 2023 Pat Deegan
Psychogenic Technologies INC
Sheet: /
File: caravel-mvp.kicad_sch

Title: Caravel Minimum Viable PCB Example: TinyTapeout Version

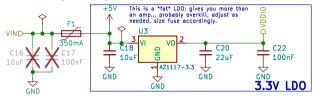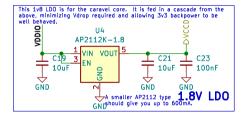| Size: A4	Date: 2023-10-24	Rev: 1.1
KiCad E.D.A. kicad 7.0.8-7.0.8~ubuntu22.04.1		Id: 1/3

Voltage Regulators

Simple voltage regulation for logic and core. In a distinct sheet to allow you to easily do fancy stuff, like use switchers or whatever is needed.

This is a *fat* LDO: gives you more than an amp... probably overkill, adjust as needed, size fuse accordingly.

+5V

VIND

F1
350mA

C16 10uF
C17 100nF

GND

U3
VI 3
VO 2
GND
AZ1117-3.3

C18 10uF
GND

C20 22uF
GND

C22 100nF
GND

VDDIO

3.3V LDO

GND

This 1v8 LDO is for the caravel core. It is fed in a cascade from the above, minimizing Vdrop required and allowing 3v3 backpower to be well behaved.

VDDIO

U4
AP2112K-1.8
VIN 1
VOUT 5
EN 3
GND 2

C19 10uF
GND

C21 10uF
GND

C23 100nF
GND

VCCD

1.8V LDO

A smaller AP2112 type should give you up to 600mA.

GND

(C) 2023 Pat Deegan
Psychogenic Technologies INC

Sheet: /Power_Regulation/
File: power_reg.kicad_sch

Title: Voltage Regulation

Size: User	Date: 2023-09-30	**Rev: 1.1**
KiCad E.D.A. kicad 7.0.8-7.0.8~ubuntu22.04.1		Id: 2/3

Stand—Alone TinyTapeout Reqs and Knobs
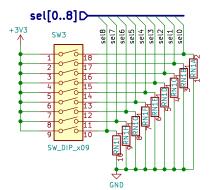
If not using the Caravel CPU, you can save space and cost by dumping its support components but that means you need to handle a bit of its work externally. Here are the related components and configs.

Project Selection

Projects are selected using the 9 active_select bits (sel* here).
If you're only interested in a single specific project, you can hardwire it here as shown in this image.
For example, I want project #267, which in binary (MSB) is:
1 0000 1011
Bringing every '1' bit high and tying ever '0' to ground, will select this value.

Or we can make things more configurable, by using switches or other means. The main thing to consider is always ensuring the signals are in a defined state (pulled high or low, not floating).

Example

sel[0..8]

Rather than hard—wiring a selection, here is one scheme to allow any of the projects to be active using a DIP switch and resistor network.
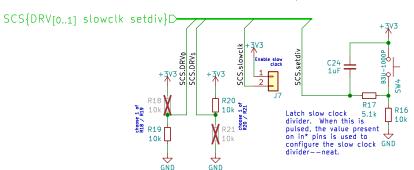
When a switch is open, the line is pulled low. When it is closed, that bit goes high.

The pull down resistors on each sel* isn't required when you have everything you need to run the Caravel CPU because, as part of the running firmware, it is trivial to enable pull—up or —down on any of the I/O pins.

SW3
SW_DIP_x09

Scan Chain Driver And Clocking

The scan chain stuff is a little involved... best to refer to the TinyTapeout documentation.
Still, notes here should be somewhat helpful.

SCS{DRV[0..1] slowclk setdiv}

SCS.DRV0
SCS.DRV1
SCS.slowclk
SCS.setdiv

Enable slow clock
J7

R18 10k — choose 1 of R18 / R19
R19 10k

R20 10k — choose 1 of R20 / R21
R21 10k

C24 1uF
B3U—1000P
SW4
R17 5.1k
R16 10k

Latch slow clock divider. When this is pulsed, the value present on in* pins is used to configure the slow clock divider——neat.

The scan chain driver selection is done through these two pins.

DRV_1	DRV_0	setting
0	0	external
0	1	logic an.
1	X	internal

You mostly likely want internal, here, so populating as shown using DNPs.

Or you can use jumpers, such as in the TT demo board here: TinyTapeout 123 Demo PCB schematic

Note: none of these components are in the sample layout. This allows you to delete this sheet from the top level and not muck about, if you aren't going stand—alone.

Input State

in[0..7]

Depending on your use case, it might be worth ensuring the state of the input pins is always valid——i.e. using pull—ups or pull—downs. This is highly application specific but digital doesn't like floaties so much... worst case maybe just sit all the in* on some weak (high value) resistors to ground.