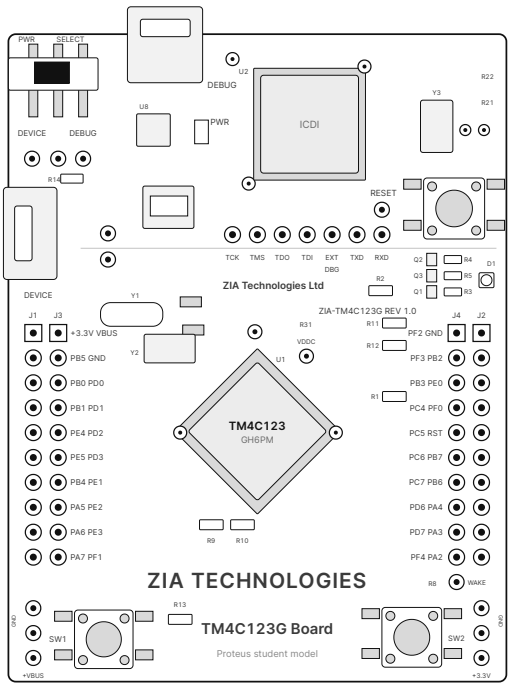
ZIA TM4C123G Board for Proteus

A model of the EK-TM4C123GXL LaunchPad for teaching embedded systems, with the same layout as the board students use in the lab.



Version
1.0 Preliminary

Published
November 2026

Document
ZIA-DOC-TM4C-001

Contents

1	Introduction	3
2	Board layout	5
3	Pin reference	6
4	On-board components	8
5	Installing the model in Proteus	9
5.1	Import the pin file	9
5.2	Draw the board symbol	12
5.3	Set the device properties	12
5.4	Create the PCB footprint	13
5.5	Share the library	13
6	Running programs	14
6.1	Run a Keil program	14
6.2	ZIA Board Viewer	15
6.3	What the simulator covers	15
6.4	Warnings	16
7	Exercises for students	16
8	Testing	17
9	Known limitations	17
10	Roadmap	18
Appendix A	Footprint coordinates	19

Title	ZIA TM4C123G Board for Proteus
Document	ZIA-DOC-TM4C-001
Version	1.0, preliminary, November 2026
Applies to	Labcenter Proteus 8.8 or later
Product page	ziafoundry.com/products/tm4c123g-board-for-proteus
Contact	info@ziafoundry.com

ZIA Foundry is the product house of ZIA TECHNOLOGIES LTD, registered in England and Wales, company number 17349446. © 2026 ZIA TECHNOLOGIES LTD. All rights reserved. This model is compatible with the pin-out of the Texas Instruments EK-TM4C123GXL and is not a Texas Instruments product. Tiva and LaunchPad are trademarks of Texas Instruments. Proteus is a trademark of Labcenter Electronics. Keil is a trademark of Arm Limited.

1

Introduction

The ZIA TM4C123G Board for Proteus is a model of the Texas Instruments EK-TM4C123GXL LaunchPad for Labcenter Proteus. Students place it in a schematic and wire parts to its header pins in the same way they wire the real board in the lab.

The model keeps the layout of the real board. USB ports, the power switch, RESET, the RGB LED, SW1, SW2 and every header pin are drawn where they are on the EK-TM4C123GXL, so a circuit drawn in Proteus can be built on the bench without translating anything.

The board part is used in Proteus schematics today. Running the students' Keil programs on the board is the job of a second product, the ZIA TM4C123G Board Simulator: students install it on their own PC and it shows the board in the web browser, where the RGB LED lights and SW1, SW2 and RESET are pressed with the mouse (section 6). The same simulator becomes the Proteus simulation model once it is built with Labcenter's VSM SDK (section 10).

Products

Both products are made by ZIA Foundry. Their pages are **ziafoundry.com/products/tm4c123g-board-for-proteus** and **ziafoundry.com/products/tm4c123g-board-simulator**, and every manual can be downloaded from **ziafoundry.com/documentation**.

In this package

File	Purpose
`library/ZIA_TM4C123G_BOARD.bsd`	Pin description file. Proteus reads it to create the part with all 46 pins.
`library/ZIA_TM4C123G_pinmap.csv`	Pin table with on-board use and alternate functions.
`library/ZIA_TM4C123G_footprint.csv`	Pad coordinates for the PCB footprint.
`graphics/zia_tm4c123g_board_only.svg`	Drawing reference for the Proteus symbol.
`graphics/zia_tm4c123g_board.svg`	Labelled drawing of the board.
`sim/bin/zia_sim.exe`	Simulator: runs a Keil `.axf` file and prints what the board does.
`sim/viewer/viewer.py`	ZIA Board Viewer: the simulator on a live board drawing in the web browser.
`sim/proteus/`	Source of the Proteus simulation model and its board graphics.
`sim/TEST_REPORT.md`	Result of the last full test run.

File	Purpose
`tools/`	Scripts that generate the package from a single pin list.

2

Board layout

Figure 1 shows the top of the board. Signal names are printed in pairs beside the header holes, as on the real board. Numbers in grey are the pin numbers of the Proteus part.

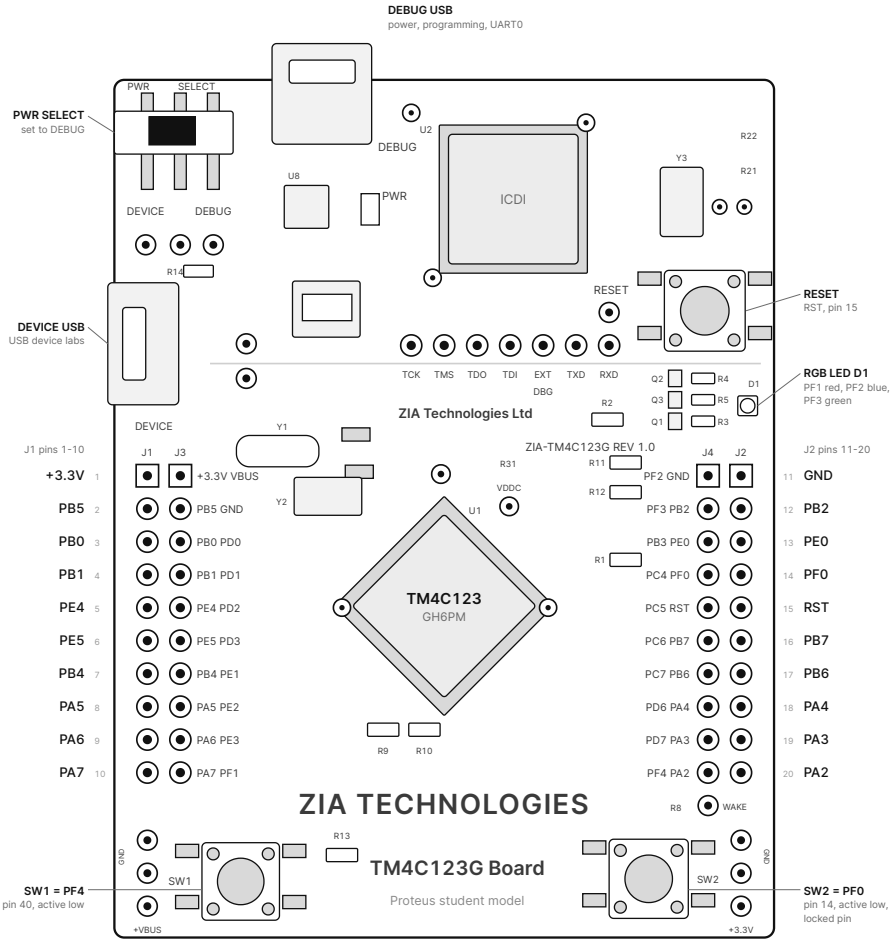
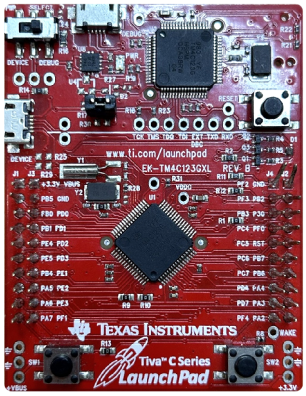
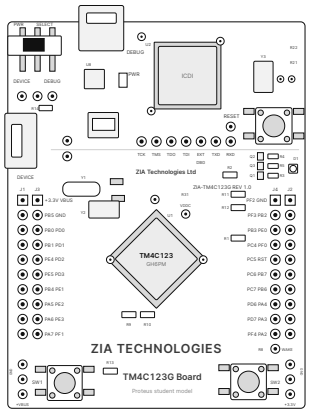


Figure 1 Top view of the ZIA TM4C123G Board. Pins 1 to 40 follow the BoosterPack numbering of J1 to J4.



EK-TM4C123GXL, photo straightened



ZIA model

Figure 2 The real board and the model side by side.

3

Pin reference

J1 and J3 are on the left of the board, J4 and J2 on the right. The DBG pads above the headers carry the JTAG signals and UART0, which reaches the PC as a virtual COM port.

J1 left, outer

Pin	Header	Signal	On the board	Alternate functions
1	J1.1	**+3.3V**	3.3 V supply	
2	J1.2	**PB5**		AIN11, SSI2Fss, M0PWM3, T1CCP1, CAN0Tx
3	J1.3	**PB0**		U1Rx, T2CCP0, USB0ID
4	J1.4	**PB1**		U1Tx, T2CCP1, USB0VBUS
5	J1.5	**PE4**		AIN9, U5Rx, I2C2SCL, M0PWM4, M1PWM2, CAN0Rx
6	J1.6	**PE5**		AIN8, U5Tx, I2C2SDA, M0PWM5, M1PWM3, CAN0Tx
7	J1.7	**PB4**		AIN10, SSI2Clk, M0PWM2, T1CCP0, CAN0Rx
8	J1.8	**PA5**		SSI0Tx
9	J1.9	**PA6**		I2C1SCL, M1PWM2
10	J1.10	**PA7**		I2C1SDA, M1PWM3

J3 left, inner

Pin	Header	Signal	On the board	Alternate functions
21	J3.1	**VBUS**	5 V from USB	
22	J3.2	**GND**	Ground	
23	J3.3	**PD0**	Tied to PB6 by R9 (0 ohm)	AIN7, SSI3Clk, SSI1Clk, I2C3SCL, M0PWM6, M1PWM0, WT2CCP0

Pin	Header	Signal	On the board	Alternate functions
24	J3.4	**PD1**	Tied to PB7 by R10 (0 ohm)	AIN6, SSI3Fss, SSI1Fss, I2C3SDA, M0PWM7, M1PWM1, WT2CCP1
25	J3.5	**PD2**		AIN5, SSI3Rx, SSI1Rx, M0FAULT0, WT3CCP0, USB0EPEN
26	J3.6	**PD3**		AIN4, SSI3Tx, SSI1Tx, IDX0, WT3CCP1, USB0PFLT
27	J3.7	**PE1**		AIN2, U7Tx
28	J3.8	**PE2**		AIN1
29	J3.9	**PE3**		AIN0
30	J3.10	**PF1**	RGB LED red	U1CTS, SSI1Tx, M1PWM5, T0CCP1, C1o, TRD1

J4 right, inner

Pin	Header	Signal	On the board	Alternate functions
31	J4.1	**PF2**	RGB LED blue	SSI1Clk, M0FAULT0, M1PWM6, T1CCP0, TRD0
32	J4.2	**PF3**	RGB LED green	SSI1Fss, CAN0Tx, M1PWM7, T1CCP1, TRCLK
33	J4.3	**PB3**		I2C0SDA, T3CCP1
34	J4.4	**PC4**		C1-, U4Rx, U1Rx, M0PWM6, IDX1, WT0CCP0, U1RTS
35	J4.5	**PC5**		C1+, U4Tx, U1Tx, M0PWM7, PhA1, WT0CCP1, U1CTS
36	J4.6	**PC6**		C0+, U3Rx, PhB1, WT1CCP0, USB0EPEN
37	J4.7	**PC7**		C0-, U3Tx, WT1CCP1, USB0PFLT
38	J4.8	**PD6**		U2Rx, M0FAULT0, PhA0, WT5CCP0
39	J4.9	**PD7**	Locked pin	U2Tx, PhB0, WT5CCP1, NMI
40	J4.10	**PF4**	SW1 (active low)	M1FAULT0, IDX0, T2CCP0, USB0EPEN

J2 right, outer

Pin	Header	Signal	On the board	Alternate functions
11	J2.1	**GND**	Ground	
12	J2.2	**PB2**		I2C0SCL, T3CCP0
13	J2.3	**PE0**		AIN3, U7Rx
14	J2.4	**PF0**	SW2 (active low), locked pin	U1RTS, SSI1Rx, CAN0Rx, M1PWM4, PhA0, T0CCP0, NMI, C0o

Pin	Header	Signal	On the board	Alternate functions
15	J2.5	**RST**	Reset (active low)	
16	J2.6	**PB7**	Tied to PD1 by R10 (0 ohm)	SSI2Tx, M0PWM1, T0CCP1
17	J2.7	**PB6**	Tied to PD0 by R9 (0 ohm)	SSI2Rx, M0PWM0, T0CCP0
18	J2.8	**PA4**		SSI0Rx
19	J2.9	**PA3**		SSI0Fss
20	J2.10	**PA2**		SSI0Clk

DBG pads

Pin	Header	Signal	On the board	Alternate functions
41	DBG.1	**TCK**	JTAG clock (PC0)	SWCLK
42	DBG.2	**TMS**	JTAG mode (PC1)	SWDIO
43	DBG.3	**TDI**	JTAG data in (PC2)	
44	DBG.4	**TDO**	JTAG data out (PC3)	SWO
45	DBG.5	**PA0/RXD**	UART0 RX, virtual COM port	U0Rx, CAN1Rx
46	DBG.6	**PA1/TXD**	UART0 TX, virtual COM port	U0Tx, CAN1Tx

Note

PB6 is joined to PD0 through R9, and PB7 to PD1 through R10. A program should not drive both pins of a pair.

4

On-board components

A few parts on the real board are wired to the microcontroller. The simulator and the ZIA Board Viewer include them. In a Proteus schematic, until the simulation model is installed, place them beside the board part so the schematic behaves like the hardware.

Part	Pin	On the real board	In Proteus
RGB LED, red	PF1 30	Transistor driver, on when the pin is high	`LED-RED` and 330 Ω to GND
RGB LED, blue	PF2 31	As above	`LED-BLUE` and 330 Ω
RGB LED, green	PF3 32	As above	`LED-GREEN` and 330 Ω
SW1	PF4 40	Button to GND, no external pull-up	`BUTTON` to GND, internal pull-up on
SW2	PF0 14	Button to GND, PF0 is locked after reset	`BUTTON` to GND, unlock PF0, pull-up on
RESET	RST 15	Button to GND with pull-up	`BUTTON` to GND, 10 kΩ to 3.3 V

Supply The TM4C123 is a 3.3 V device. VBUS on pin 21 is the 5 V USB supply and must not be connected to a port pin.

PF0 Write `0x4C4F434B` to `GPIO_PORTF_LOCK_R` and set bit 0 of `GPIO_PORTF_CR_R` before configuring PF0.

5 Installing the model in Proteus

The instructor does this once and shares the resulting library with students. Proteus 8.8 or later is recommended because it includes **Library › Import Parts**. Saving a library needs a licensed copy.

5.1 Import the pin file

The screenshots in this section were taken in Proteus 9.2. Proteus 8 looks almost the same.

1 Open **Schematic Capture** and choose **Library › Import Parts**.

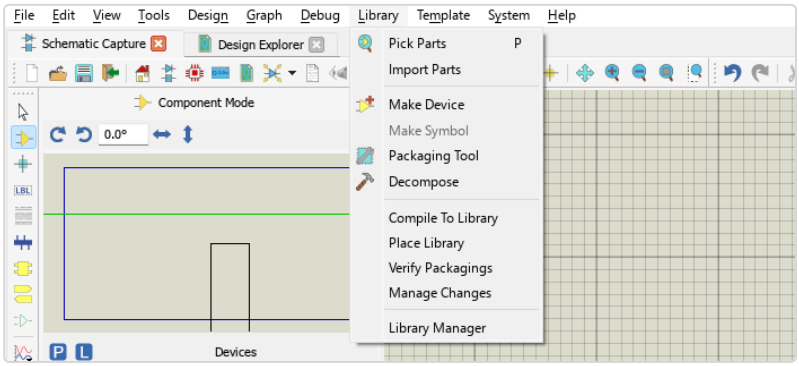


Figure 3 The Library menu. Import Parts reads the pin file.

2 Click **Select File**. Set the file type to **BSDL (*.bsdl *.bsd)**, open the `library` folder of this package and choose `ZIA_TM4C123G_BOARD.bsd`. Older Proteus 8 releases offer the same BSDL import on the pin page of **Make Device**.

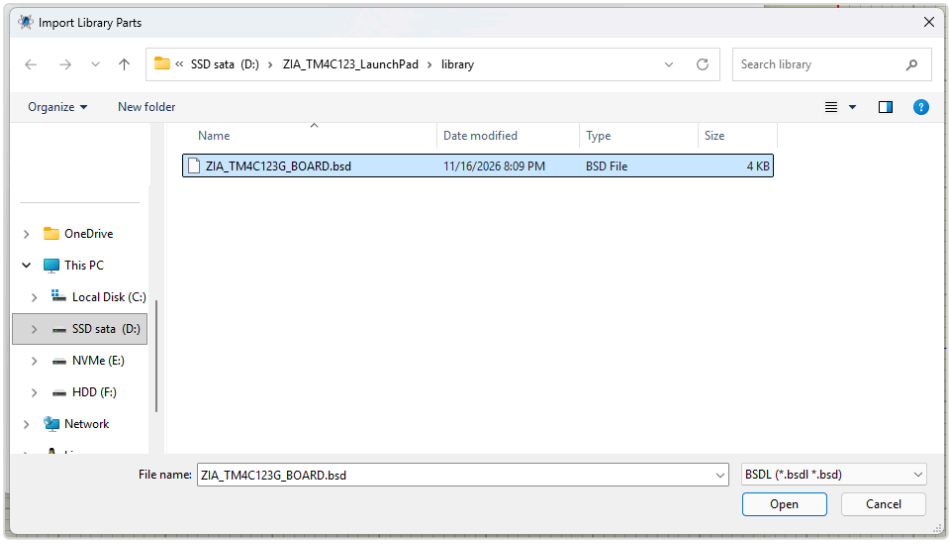


Figure 4 Choosing the pin file. The file type must be BSDL, or the file is not listed.

3 The log under the preview must end with *Parsed 1 part(s) with 0 warning(s)*, and the package box shows `ZIA_TM4C123G_BRD`. Click **Import Part**.

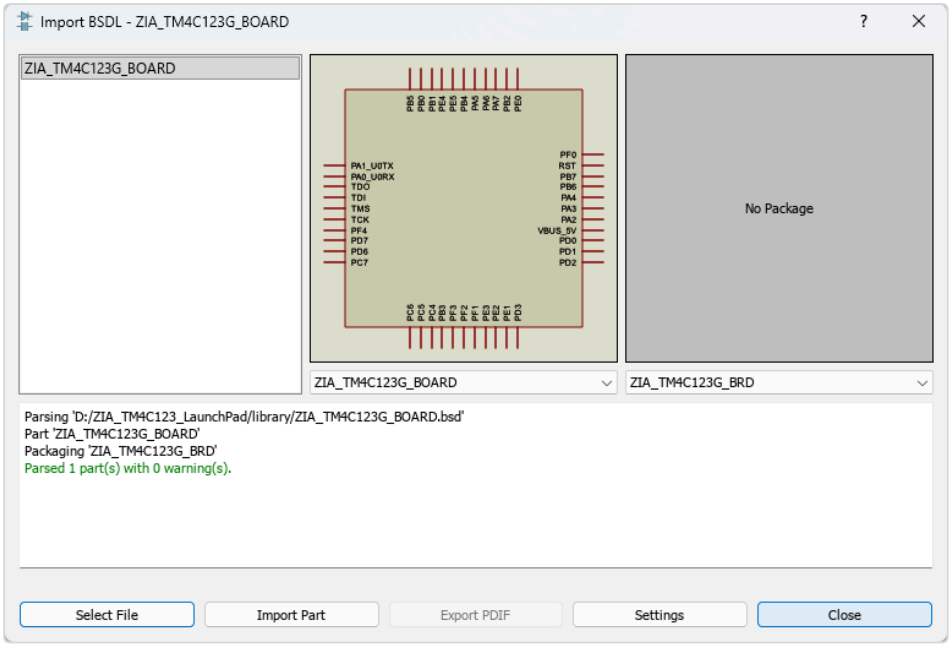


Figure 5 The pin file read with no warnings. Proteus draws a temporary chip outline; the board drawing comes in section 5.2.

4 The **Make Device** wizard opens on **Pins Configuration**. Untick **Hide Power Pins?** and check that the list holds 46 pins with the names in section 3. Set `VDD_3V3`, `VBUS_5V`, `GND1` and `GND2` to Power, `RST` to Input and the port pins to Bidirectional, then click **Next**.

5 On **Device Properties** keep the device name and set the reference prefix to `BRD`. Leave the Active Component Properties empty; they are used by the simulation model (section 10).

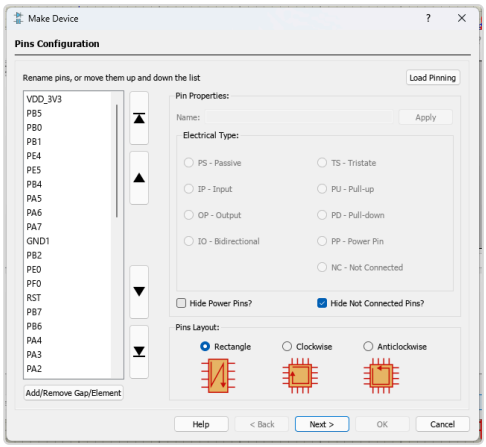


Figure 6 Pins Configuration with Hide Power Pins unticked.

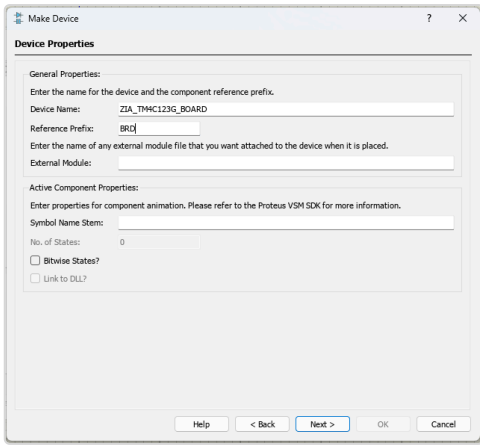


Figure 7 Device Properties with the prefix BRD.

6 **Packagings** lists `ZIA_TM4C123G_BRD` and says it is not found. That is expected until the footprint is made in section 5.4. Click **Next**.

7 **Component Properties** already holds the PACKAGE property. Click **Next**, and **Next** again on the data sheet page.

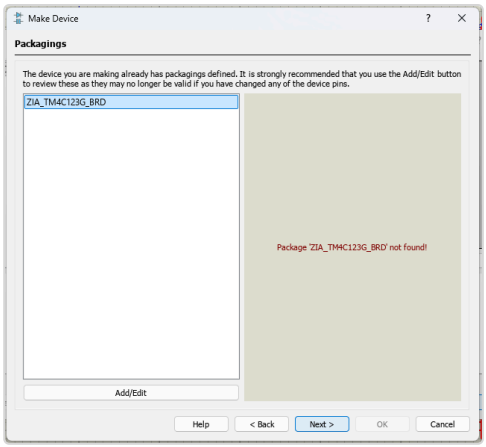


Figure 8 Packagings: the footprint is added in section 5.4.

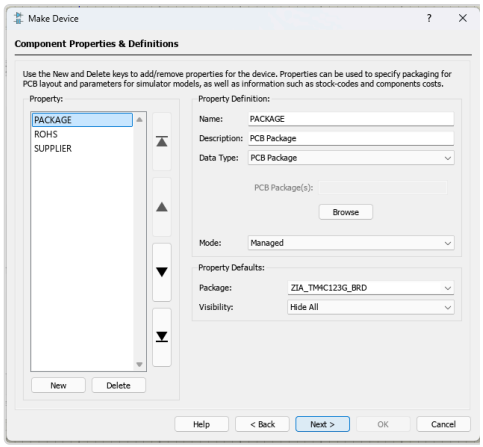


Figure 9 Component Properties, left as they are.

8 On **Indexing and Library Selection** enter the values in section 5.3, choose the library to save the part in and click **OK**.

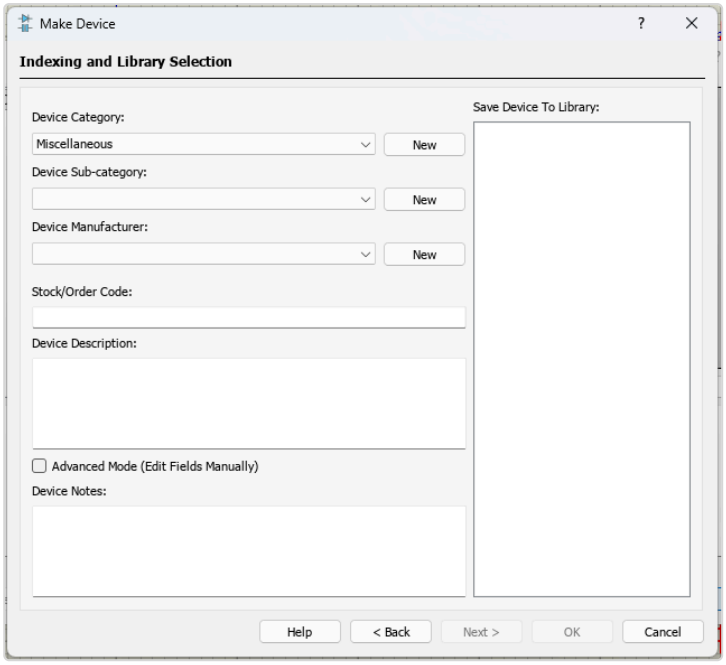


Figure 10 The last page. In the free Proteus demo the library list is empty and OK stays grey, because the demo cannot save parts.

Power pins Keep the four power pins visible on the symbol. Proteus joins a hidden power pin to a net with the same name as the pin, so hidden `GND1` and `GND2` would not join the `GND` net.

Demo The Proteus demo goes as far as step 7. Saving the part needs a licensed copy of Proteus.

5.2 Draw the board symbol

The symbol should look like the board rather than a chip outline. Draw it over `graphics/zia_tm4c123g_board_only.svg` with the grid at 100 mil, which matches the header pitch.

1. Draw a 2.0 × 2.6 in board outline.
2. Put each pin on its header hole using the coordinates in Appendix A. The symbol and the footprint share these values.
3. J1 pins point left and J2 pins point right, out of the board. J3 pins point right and J4 pins point left, into the board. Use 100 mil pins and hide pin names; the printed labels show them.
4. Place DBG pins 41 to 46 on the DBG holes, pointing up.
5. Add the USB connectors, PWR SELECT switch, RESET, RGB LED, microcontroller, SW1 and SW2 in their drawn positions, then the printed labels.

5.3 Set the device properties

Property	Value
Device name	`ZIA_TM4C123G_BOARD`
Reference prefix	`BRD`

Property	Value
Category	`Microprocessor ICs`, sub-category `ZIA Foundry`
Manufacturer	`ZIA TECHNOLOGIES LTD`
Description	`ZIA TM4C123G board model, EK-TM4C123GXL compatible`
Library	`ZIA_TIVA.LIB`

5.4 Create the PCB footprint

In **PCB Layout**, place 46 through-hole pads, 70 mil round with a 40 mil drill and a square pad on pin 1 of each header, at the coordinates in Appendix A. Add a 2000 × 2600 mil outline on the silkscreen layer, select everything and choose **Library › Make Package**. Save it as `ZIA_TM4C123G_BRD` and assign it on the packaging page of **Make Device**.

5.5 Share the library

Give students `ZIA_TIVA.LIB`. They copy it into the user library folder shown under **System › System Settings › Library folders**, restart Proteus and find the part in **Pick Devices** by searching for `ZIA`.

6

Running programs

The ZIA TM4C123G Board Simulator runs the program a student builds in Keil on a model of the board. It does not need Proteus or the real board, so students can check their code at home. It gives the same results as the board for the parts listed in section 6.3, and it explains common mistakes in plain words (section 6.4).

Licence

The simulator is installed with its own setup program from **ziafoundry.com/products/tm4c123g-board-simulator**. It runs on computers signed in to a ZIA licence: a university or company gets the licence, and its students and staff sign in at **portal.ziafoundry.com** with their own university or company email.

Manual

The *ZIA TM4C123G Board Simulator User Manual* (ZIA-DOC-TM4C-002) shows every step with screenshots: installing, signing in, Keil uVision and the viewer.

6.1 Run a Keil program

1. Install the simulator. Setup adds **Run on ZIA Board** to the **Tools** menu of Keil uVision 5.
2. Build the project in Keil as usual with `F7`. The program is the `.axf` file in the project's `Objects` folder.
3. Click **Tools › Run on ZIA Board**. The board opens in the web browser with the program running.

The simulator also runs without the browser, which is useful to see exactly when each pin changed. In a Command Prompt, in the `bin` folder of the installed simulator:

```
zia_sim.exe lab1.axf --ms 1000 --events
```

Option	What it does
`--ms 1000`	Board time to run, in milliseconds.
`--press SW1:200:400`	Hold SW1 from 200 ms to 400 ms. Also `SW2` and `RESET`; repeat the option for more presses.
`--analog PE3=1.65`	Voltage on an analog input, by pin or by name (`AIN0=1.65`).
`--temp 25`	Chip temperature in °C for the internal temperature sensor.
`--i2c 0:24C02:0x50`	Put a 24C02 EEPROM on I²C0 at address 0×50. `LM75` adds a temperature sensor.
`--spi 2:ram:PB5`	Put an SPI RAM on SSI2 with its chip select on PB5. `wire` joins MOSI to MISO.
`--events`	Print every pin change with its time.
`--regs`	Print the CPU registers at the end.

6.2 ZIA Board Viewer

The ZIA Board Viewer shows the simulator on the board drawing in a web browser, at `http://127.0.0.1:8765` on the student's own PC. The program runs in real time. SW1, SW2 and RESET are pressed with the mouse where they sit on the board, the RGB LED glows in the colour and brightness the program drives (PWM dimming too), and UART0 appears in a terminal that also sends typed text to the board.

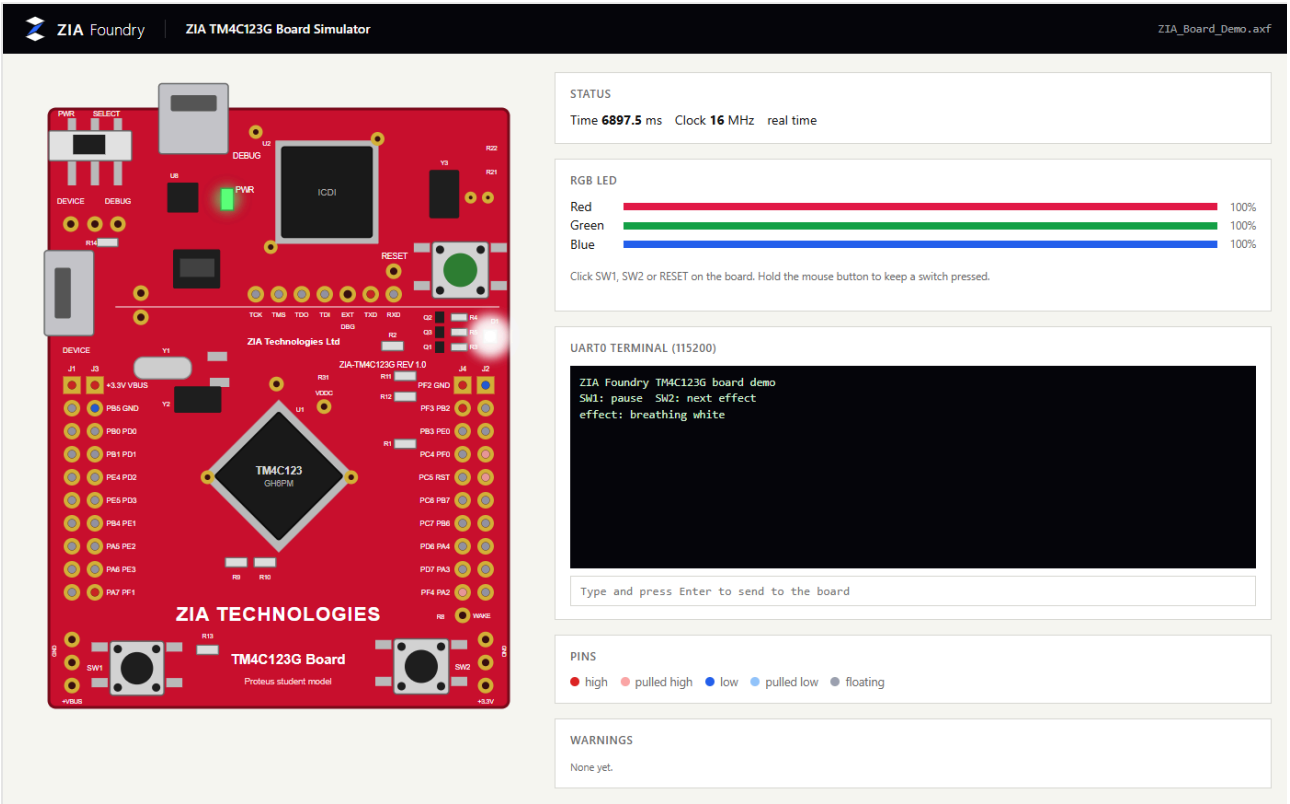


Figure 11 The ZIA Board Viewer running the demo program that comes with the simulator (Start menu, *ZIA Board demo*). SW2 changes the LED effect and SW1 pauses it. Header pins are coloured by what the board drives on them.

Speed

At 16 MHz the simulator runs 1.1 to 3.5 times faster than the real board, depending on the program. The viewer slows it to real time.

6.3 What the simulator covers

Part of the chip	Simulated
CPU	Cortex-M4F instruction set with the floating point unit, NVIC with priorities and nesting, SysTick, faults.
Clock	Main oscillator, PLL up to 80 MHz, sleep with `WFI`.
GPIO	Ports A to F, pull-up and pull-down, open drain, interrupts, the PF0 lock, the R9 and R10 links.
Board	RGB LED, SW1, SW2 and RESET.
UART	UART0 to UART7 with FIFOs, interrupts and loopback. UART0 appears in the terminal, as on the board's virtual COM port.
Timers	TIMER0 to TIMER5 and WTIMER0 to WTIMER5: periodic, one-shot, PWM, edge count and edge time capture.

Part of the chip	Simulated
ADC	ADC0 and ADC1, all sequencers, AIN0 to AIN11, temperature sensor, averaging, timer and PWM triggers.
PWM	PWM0 and PWM1 modules, count-down and up/down, deadband, interrupts and ADC triggers.
I²C and SSI	I²C0 to I²C3 and SSI0 to SSI3 in master mode, bit by bit on the pins, with the parts in section 6.1.

6.4 Warnings

The simulator watches for mistakes that are hard to find on the real board, and prints a warning instead of failing silently. Some examples:

Warning	What to fix
Port F was used before its clock was turned on (RCGCGPIO). On the real board this causes a HardFault.	Set the port's bit in `SYSCTL_RCGCGPIO_R` first.
The pin is an input with nothing connected and no pull-up or pull-down.	Turn on the pull-up for a button (`GPIO_PORTF_PUR_R`).
UART is sending but its pins are not set up (AFSEL, PCTL, DEN).	Set the alternate function of the TX and RX pins.
SDA is floating; an I²C bus needs pull-up resistors.	Add pull-ups, and set SDA to open drain.
A part of the chip that is not simulated yet was used.	The program still runs; its registers read as 0. See section 9.

7 Exercises for students

Each exercise uses standard Proteus parts. Students can build the same circuit on the real board afterwards. The simulator in section 6 already runs the program for each exercise and shows what it drives on the pins; the outside parts themselves need Proteus.

Exercise	Proteus parts	Board pins
RGB LED and buttons	LED-RED, LED-GREEN, LED-BLUE, BUTTON	PF1, PF2, PF3, PF4, PF0
Seven-segment counter	7SEG-COM-CATHODE, 330 Ω × 7	PB0 to PB7
Keypad scanning	KEYPAD-SMALLCALC	PE0 to PE3, PC4 to PC7
Character LCD	LM016L	PB0 to PB3, PA6, PA7
Serial output	VIRTUAL TERMINAL	PA1 / TXD 46, PA0 / RXD 45
Analog input	POT-HG	PE3 / AIN0
PWM and motors	MOTOR-PWMSERVO, L293D	PB6, PF2
I²C	I2C DEBUGGER, 24C02	PB2 / SCL, PB3 / SDA

8

Testing

Every change to the simulator is checked by a full test run (`python sim/run_tests.py`). It compares the model with references made without it, so a mistake in the model cannot hide in its own tests.

Check	How
Student lab programs	ENCS4110 lab programs give the same results as on the board, including a lab with a bug that crashes the board.
Instruction by instruction	Each instruction is compared with the Unicorn emulator, on the lab programs, on compiled C and on random instructions.
Test programs	25 C programs built with the Keil compiler, one or more for each part of the chip. Results are checked against values worked out separately.
Reset values	Registers after reset match the TM4C123GH6PM data sheet.
Planted bugs	33 mistakes are put into a copy of the simulator on purpose. The tests must catch every one.
Proteus connection	A stand-in for Proteus drives the 46 pins. An outside EEPROM and SPI RAM answer the program through the wires only.
Viewer and graphics	The viewer runs the demo, presses the buttons and reads the terminal. The board graphics find the right button for each click.

Result

Last run, 17 November 2026: 96 tests passed, none failed, and all 33 planted bugs were caught.

9

Known limitations

Area	Status in release 1.0
Running code in Proteus	The simulation model is written but not built. It needs the VSM SDK, which Labcenter supplies on request. Until then, use the simulator and viewer in section 6.
Not simulated yet	CAN, USB, QEI, watchdog, hibernation, EEPROM, uDMA, timer RTC mode, I²C and SSI slave mode, PWM fault inputs, ADC comparators and the TivaWare `ROM_` functions. Programs that use them run, but those registers read as 0 and a warning is printed.
Footprint	Header pads and outline only. Parts mounted on the real board are not included.
Verification	The pin file was imported into the Proteus 9.2 demo with no warnings, and the Make Device wizard listed all 46 pins. The demo cannot save a library, so the steps after section 5.1 have not been tried. The simulator has not yet been compared with the real board side by side.

10

Roadmap

Stage 2 adds a simulation model to the same part using the Proteus VSM SDK. The model loads the student's Keil `.axf` or `.hex` file, executes the Cortex-M4 instruction set of the TM4C123GH6PM and drives the header pins. The RGB LED lights on the board drawing, and SW1, SW2 and RESET can be clicked where they sit on the real board. The simulator for it is finished:

Release	Adds	Status
2.0	CPU, GPIO, SysTick, clock, on-board LED and buttons	Built and tested
2.1	UART0 to UART7, NVIC, GPIO interrupts	Built and tested
2.2	General purpose timers, timer PWM and capture	Built and tested
2.3	ADC, I²C, SSI, PWM modules	Built and tested
2.4	ZIA Board Viewer, board graphics for Proteus	Built and tested; the Proteus graphics need the SDK
2.5	Installer, licences through the ZIA Foundry Portal, updates	Built and tested; released with the portal

Next

1. Build the Proteus model with the VSM SDK, and try each exercise in a licensed Proteus.
2. Read analog voltages from the Proteus circuit on the AIN pins, and support I²C devices that hold the clock low.
3. Put the ZIA Foundry Portal online for the first university pilots.
4. Run the same programs on the real board and in the model and compare them, before each release.

A

Footprint coordinates

Hole positions in mil from J1 pin 1, with Y increasing downwards. Pads are 70 mil with a 40 mil drill. The board outline is 2000 × 2600 mil with its top-left corner at X −100, Y −1200. All values were measured on a straightened photo of the real board.

Pad	Signal	X	Y	Pad	Signal	X	Y
1	+3.3V	0	0	24	PD1	100	300
2	PB5	0	100	25	PD2	100	400
3	PB0	0	200	26	PD3	100	500
4	PB1	0	300	27	PE1	100	600
5	PE4	0	400	28	PE2	100	700
6	PE5	0	500	29	PE3	100	800
7	PB4	0	600	30	PF1	100	900
8	PA5	0	700	31	PF2	1700	0
9	PA6	0	800	32	PF3	1700	100
10	PA7	0	900	33	PB3	1700	200
11	GND	1800	0	34	PC4	1700	300
12	PB2	1800	100	35	PC5	1700	400
13	PE0	1800	200	36	PC6	1700	500
14	PF0	1800	300	37	PC7	1700	600
15	RST	1800	400	38	PD6	1700	700
16	PB7	1800	500	39	PD7	1700	800
17	PB6	1800	600	40	PF4	1700	900
18	PA4	1800	700	41	TCK	800	-395
19	PA3	1800	800	42	TMS	900	-395
20	PA2	1800	900	43	TDI	1100	-395
21	VBUS	100	0	44	TDO	1000	-395
22	GND	100	100	45	PA0/RXD	1400	-395
23	PD0	100	200	46	PA1/TXD	1300	-395