

EPM11

EPM11

10 July, 2026

Table of Contents

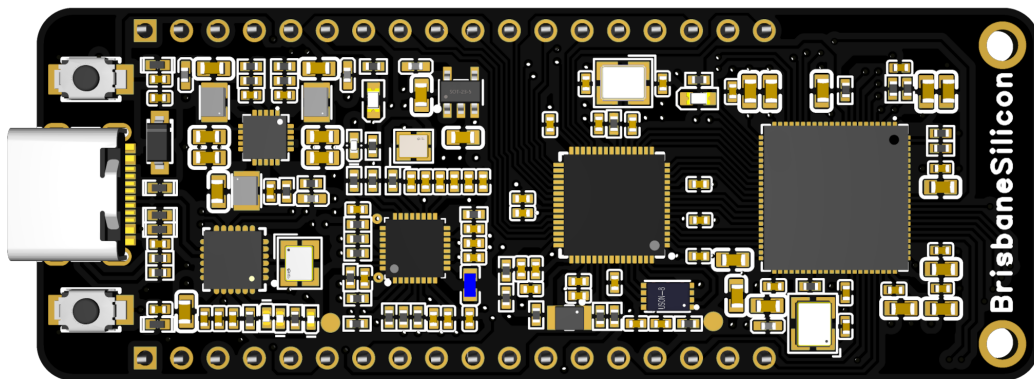
1	About	3
1.1	Image	3
2	Board Specifications.....	4
3	PC Setup	6
3.1	Prerequisites	6
3.2	Other	6
3.3	License Setup.....	8
3.3.1	Linux.....	8
3.3.2	Windows	9
3.3.3	Public License Servers.....	9
3.4	Environment Variables.....	9
3.4.1	Linux.....	9
3.4.2	Windows	11
4	FPGA Example Project.....	12
4.1	Building	12
4.1.1	Linux.....	12
4.1.2	Windows	13
4.2	Program.....	13
4.2.1	Linux.....	13
4.2.2	Windows	14
4.3	Extending.....	16
5	MCU Example Project.....	17
6	Flash RP2350 MicroPython Image	18

1 About

The EPM11 is a MCU-FPGA development board. This document details the following:

1. Specifications of the board.
2. Setting up your PC to build and program the board.
3. Building and programming the provided example project to the board.
4. Extending the example project.
5. Example project roadmap.

1.1 Image



1 EPM11

2 Board Specifications

- [RP2350](#)¹ Microcontroller
 - I/O bus connection to the FPGA.
- [GOWIN GW1NR-9](#)² FPGA.
 - 8MB internal PSRAM.
 - Internal bitstream Flash with AUTO BOOT.
- Non-volatile memory.
 - 4 MB User Flash.
- Peripherals
 - RP2350
 - 8 x GPIO.
 - 4 x ADC.
 - 1 x LED.
 - FPGA
 - 16 x GPIO.
 - 1 x LED.
 - 2 x Pushbuttons.
- USB-C connector
 - UART over USB ([Thonny](#)³)
 - JTAG over USB
- 1 x Power LED.
- 3.3V and 5V rails.
- [EPM11-MCU](#)⁴ example project (coming soon!).
 - Custom, board-specific MicroPython image.
 - Includes a scripting layer for 'out of the box' communication with the FPGA.
 - Includes a custom Pin and ADC mapping.
- [EPM11-FPGA example project](#)⁵.
 - Includes a guided setup of the FPGA build prerequisites and environment.
 - Includes a CPU interface module (coming soon!).
 - Provides an [example](#)⁶ of building and probing an [fpgacapZero](#)⁷ Embedded Logic Analyzer (ELA).

1. <https://www.raspberrypi.com/products/rp2350/>

2. https://www.gowinsemi.com/en/product/littlebee_fpga/

3. <https://thonny.org/>

4. <https://github.com/BrisbaneSilicon/EPM11-MCU>

5. <https://github.com/BrisbaneSilicon/EPM11-FPGA>

6. <https://github.com/BrisbaneSilicon/EPM11-FPGA#embedded-logic-analyzer>

7. <https://github.com/lcapossio/fpgacapZero>

- [Extensible](#)⁸, i.e. user can add their own application specific RTL modules.
- Simple [build](#)⁹ and [program](#)¹⁰ workflow stages.
- Fully scripted workflow – no GUI-based programs are required.

8. <https://github.com/BrisbaneSilicon/EPM11-FPGA?tab=readme-ov-file#development>

9. <https://github.com/BrisbaneSilicon/EPM11-FPGA?tab=readme-ov-file#build>

10. <https://github.com/BrisbaneSilicon/EPM11-FPGA?tab=readme-ov-file#program>

3 PC Setup

Fulfill the below prerequisites to setup your PC to build and program FPGA firmware to the EPM11.

3.1 Prerequisites

1. A PC running an x64 compatible, Debian-based flavour of Linux or Windows 11.
 - Other flavours of Linux may work but aren't officially supported.
 - We recommend [Ubuntu](https://ubuntu.com/)¹¹.
2. An installation of [GIT](https://git-scm.com/)¹².
3. An installation of GOWIN EDA V1.9.12.
 - Available from the official GOWIN EDA [download page](#)¹³ or via direct links, [Linux](#)¹⁴ [Windows](#)¹⁵.
 - You may need to first register as a GOWIN member [here](#)¹⁶.
 - On Linux, please ensure that GOWIN EDA is installed to one of the following directories (or be prepared to modify the build script):
 - `$HOME/Applications /opt/gowin /opt/GOWIN /opt/Gowin $HOME/Documents/Applications/`
 - On Windows, please ensure that GOWIN EDA is installed to one of the following directories (or be prepared to modify the build script):
 - `C:\Gowin, C:\Program Files\Gowin`
4. A free license for GOWIN EDA. See section [License Setup](#)¹⁷ below.
5. A copy of this repository.
 - Launch a terminal program.
 - Navigate to the directory in which you wish to host the BRS-100-GW1NR9 repository.
 - `git clone https://github.com/BrisbaneSilicon/BRS-100-GW1NR9.git`

3.2 Other

1. An FTDI driver is required if you wish to communicate with the BRS-100-GW1NR9 via UART. On most Linux distributions they are part of the default installation of the OS. On Windows, you may need to install the driver from [here](#)¹⁸. FTDI also provide installation guides, available [here](#)¹⁹.

11. <https://ubuntu.com/>

12. <https://git-scm.com/>

13. https://www.gowinsemi.com/en/support/download_eda/

14. https://cdn.gowinsemi.com.cn/Gowin_V1.9.12_linux.tar.gz

15. https://cdn.gowinsemi.com.cn/Gowin_V1.9.12_x64_win.zip

16. <https://www.gowinsemi.com/en/member/>

17. <https://github.com/BrisbaneSilicon/BRS-100-GW1NR9#license-setup>

18. <https://ftdichip.com/drivers/vcp-drivers/>

2. Windows version require matching FTDI driver versions, if you already have FTDI installed earlier. Mismatched FTDI driver versions can cause Windows to crash with `KERNEL_SECURITY_CHECK_FAILURE (0x139)`.

To check for driver conflicts, run:

```
pnputil /enum-drivers | Select-String -Pattern "ftdi" -Context 5
```

Ensure all listed driver versions match. If they do not, follow the full FTDI reinstall procedure:

1. Remove all FTDI devices and drivers Open Device Manager (devmgmt.msc), then enable hidden devices via View > Show hidden devices. Look under:

```
"Universal Serial Bus controllers" – any FTDI entries
"Ports (COM & LPT)" – any "USB Serial Port" entries
"USB Debugger A" entries
"USB Serial Converter" entries
Right-click each device > Uninstall device > check "Attempt to remove the driver for
this device" (important).
```

2. Clear ghost devices

With the board unplugged, open an admin PowerShell:

```
``pnputil /enum-drivers | Select-String -Pattern "ftdi" -Context 5``
```

This lists any FTDI drivers still in the driver store. For each one, note the published name (e.g. oem12.inf) and remove it:

```
``pnputil /delete-driver oem12.inf /force``
```

3. Clean leftover files

Check **for** stale copies:

```
Get-ChildItem C:\Windows\System32\drivers\ftd*.sys
Get-ChildItem C:\Windows\System32\ftd2xx*.dll
Get-ChildItem C:\Windows\SysWOW64\ftd2xx*.dll
```

These should be gone after step 2. If any remain, note the versions (Right-click > Properties > Details > File version) before deleting – **this** tells you what was installed.

4. Reboot

Reboot before reinstalling anything. This ensures the kernel fully unloads the old drivers.

5. Reinstall clean

Download the latest D2XX driver from FTDI (not from GOWIN):
<https://ftdichip.com/drivers/d2xx-drivers/>

Run the installer. Then plug in the board – Windows should pick up the **new** drivers.

6. Verify versions match

After reinstall, check **if** the driver versions match by running:

```

...
pnputil /enum-drivers | Select-String -Pattern "ftdi" -Context 5
...

```

3.3 License Setup

There are two options (both free) for licensing GOWIN EDA.

1. Using a local license file.
2. Using a floating license server.

Option (1) requires applying for a license from GOWIN [here](#)²⁰. It can take up to a few working days for GOWIN to provide you with a license, which will be valid for a period of one year. Option (2) requires the same initial step if you wish to host your own license server. Alternatively you can point the GOWIN license manager at a public license server; this is likely the quickest path forwards. See [Public License Servers](#)²¹ for a list of public license servers.

3.3.1 Linux

Launch a bash terminal and perform the following:

1. Change directory to the GOWIN IDE installation 'bin' directory.
 - `cd <GOWIN IDE install directory>/IDE/bin/`
2. Run the licensing manager.
 - `./license_config_gui`
 - Alternatively you can run the GOWIN IDE `./gw_ide` and click 'Help' - 'Manage License'.
3. Either point the licensing manager at your local license file (Option 1 above) or a floating license server.
4. Press 'Check' to validate the license.
 - If the license has been successfully validated, it should produce a popup window **INFO** with the message **Server is OK**.
5. Click 'Save' to save your license setup.

Warning

20. <https://www.gowinsemi.com/en/support/license/>

21. <https://github.com/BrisbaneSilicon/BRS-100-GW1NR9#public-license-servers>

Sometimes the first license check (step 4) will fail - simply repeat the step to validate the license.

3.3.2 Windows

To configure the license on Windows GOWIN EDA:

1. Open GOWIN EDA (`gw_ide.exe`) from your install directory.
2. Click **Help** → **Manage License**.
3. Enter either your local license file path or the server IP and port.
4. Click **Check** – a popup saying **Server is OK** confirms it works.
5. Click **Save**.

Warning

Sometimes the first license check (step 4) will fail - simply repeat the step to validate the license.

3.3.3 Public License Servers

A list of public GOWIN EDA license servers is below. These are community reported and might not be official or stable (see the previous instructions on how to check their validity).

IP Address	Port
106.55.34.119	10559
43.128.7.128	10559

3.4 Environment Variables

3.4.1 Linux

1. Prior to building the project, set 'LD_LIBRARY_PATH' to point at the GOWIN IDE libraries at `<GOWIN IDE install directory>/IDE/lib`. For example:

```
export LD_LIBRARY_PATH=/opt/Gowin/Gowin_V1.9.12_linux/IDE/lib
```

2. Prior to programming the BRS-100-GW1NR9 board, add `<GOWIN IDE install directory>/Programmer/bin/` to your PATH. For example:

```
export PATH=$PATH:/opt/Gowin/Gowin_V1.9.12_linux/Programmer/bin/
```

Warning

When attempting to build the project, you may get the following error:

```
symbol lookup error: /<system library path>/libfontconfig.so.1: undefined symbol:
FT_Done_MM_Var.
```

To rectify this, setup the following environment variable as part of your pre-build environment, substituting '<system library path>' to be as per your specific error message:

```
export LD_PRELOAD=$LD_PRELOAD:/<system library path>/libfreetype.so
```

When attempting to program the board, you may get the following error:

```
programmer_cli: /<GOWIN EDA install path>/Programmer/bin/libz.so.1: version
`ZLIB_1.2.3.4' not found (required by /lib/x86_64-linux-gnu/libpng16.so.16)
```

To rectify this, setup the following environment variable as part of your pre-build environment:

```
export LD_PRELOAD=$LD_PRELOAD:/<system library path>/libz.so.1
```

Note that the '<system library path>' may differ from the build error - you might need to use the 'ldd' program to locate libz.so.1.

Note

If you are going to be building the project often, we recommend (on Linux) you add a bash alias for the above environment variables to your '.bashrc'. For example:

```
alias gowin_ini='export LD_LIBRARY_PATH=/opt/Gowin/Gowin_V1.9.12_linux/IDE/lib; \
export PATH=$PATH:/opt/Gowin/Gowin_V1.9.12_linux/Programmer/bin; export \
LD_PRELOAD=/lib/x86_64-linux-gnu/libfreetype.so'
```

Then to initialise the build environment, you can simply run 'gowin_ini' from bash.

3.4.2 Windows

3.4.2.1 PowerShell Execution Policy

The scripts require PowerShell's execution policy to be set to `RemoteSigned` or higher. To check your current policy:

```
Get-ExecutionPolicy
```

If it does not return `RemoteSigned` or `Unrestricted`, open PowerShell as Administrator and run:

```
Set-ExecutionPolicy RemoteSigned
```

Or for current user only:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

4 FPGA Example Project

The example project allows the user to build an FPGA bitstream and program it onto the [EPM11](#)²² FPGA development board.

The example project can also be extended by the user to include their custom, application-specific, RTL modules.

The example project workflow is fully scripted (fetch, build, program); it does not require the use of a GUI-based program at any point.

4.1 Building

After fulfilling all of the prerequisites, and initializing your build environment, you are ready to build the EPM11 FPGA firmware! Simply perform the following:

4.1.1 Linux

```
cd <EPM11 repository directory>/EPM11/build
./build.sh
```

That's all there is to it!

The build script supports various customizations via command line arguments. To view the full set of supported command line arguments, simply perform the following:

```
./build.sh -h
```

The most commonly used are listed below.

Build Argument	Description
-y, --list_supported_system_clock_frequencies	List the supported system clock frequencies and exit.
-e, --embedded_logic_analyzer	Include an Embedded Logic Analyzer (fpgacapZero) in the bitstream.
-r, --disable_pushbutton_reset	Disable pushbutton 1 as hard reset.

22. <https://brisbanesilicon.com.au/epm11/>

Build Argument	Description
-k, --clock_frequency FREQUENCY_MHZ	Use a frequency of FREQUENCY_MHZ for the system clock (default 51 MHz).
-a, --clean_all_platforms	Perform cleanup of the entire build and exit.
-p, --proj_only	Only generate the project file, then exit. Useful if the user wishes to utilize GOWIN IDE.
-s, --synth_only	Only proceed with build until synthesis is complete, then exit. Useful to check FPGA utilization, timing etc.

4.1.2 Windows

Open PowerShell (Admin not required), `cd` into the repository root, then the 'build' directory, and then run:

```
.\build.ps1
```

4.2 Program

After building the EPM11 firmware you are ready to program it to the board! It is worth noting that this stage also programs the internal non-volatile bitstream flash, so your firmware will auto-load after board power on!

After plugging the board into your PC via the USB-C cable, simply perform the following:

4.2.1 Linux

The '<this repository directory>' is the directory in which you performed Step (4) of [prerequisites](#)²³ - i.e. the directory in which you cloned this repository.

```
cd <this repository directory>/EPM11/prog
./program.sh
```

That's all there is to it!

Note

23. <https://github.com/BrisbaneSilicon/BRS-100-GW1NR9#prerequisites>

The 'program' script will trigger a build the BRS-100-GW1NR9 firmware (with no command line customization arguments) if it detects it hasn't already been built.

Again, the program script supports various customizations via command line arguments. To view the full set of supported command line arguments, simply perform the following:

```
./program.sh -h
```

The most commonly used are listed below.

Program Argument	Description
-d, --list_default_target	List the default build target.
-f, --program_flash	Program the bitstream to the onboard flash.
-c, --clean_target_prior	Clean TARGET build prior to building and programming the EPM11 board.
-b, --check_if_target_built	Print firmware built status of provided target board and exit.

Warning

If you have previously loaded an FTDI driver in order to utilise the user comms, you will need to remove it prior to programming the board:

```
sudo rmmod ftdi_sio
```

4.2.2 Windows

4.2.2.1 Program Board

Warning

Make sure there are no conflicts between FTDI driver versions before running this script – see [Other](#)²⁴.

Plug the board into your PC via the USB-C cable, then open PowerShell, `cd` into the repository root, then the 'prog' directory, and then run:

24. <https://github.com/BrisbaneSilicon/BRS-100-GW1NR9#ftdi-driver-setup>

```
.\program_board.ps1
```

Note

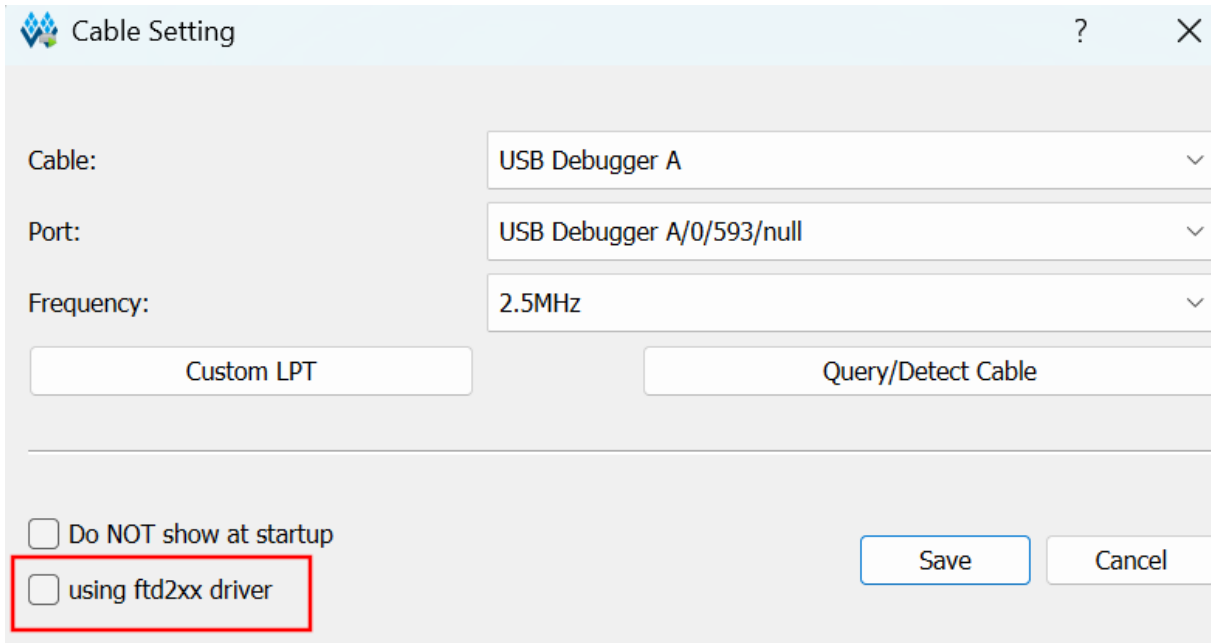
The script may freeze during programming at Operation "embFlash Erase,Program" for device#1... line and then hangs indefinitely with no further output. Typically occurs after 3-6 rapid programming commands. This is due to the FTDI chip accumulating internal states across rapid programmer_cli.exe calls. After a few cycles, the chip stops responding to new JTAG commands. This may take a few tries depending on the state of the FTDI chip.

Manual Recovery steps:

1. Press Ctrl+C in the terminal to stop the script.
2. Open Task Manager (Ctrl+Shift+Esc) and check if programmer_cli.exe is still running. If it is, right-click it and select End Task. Alternatively, run in a new PowerShell window: `Stop-Process -Name programmer_cli -Force -ErrorAction SilentlyContinue`
3. Unplug the USB-C cable from the board.
4. Wait at least 3 seconds before reconnecting. The FTDI chip's internal microcontroller needs time to fully power down and clear its state. If you replug too quickly (<3 seconds), the chip resumes with stale state and the first programming attempt will likely fail again.
5. Plug the USB-C cable back in.
6. Wait for Windows to finish enumerating the device (Device Manager will show "USB Debugger A" again – usually takes 2-3 seconds).
7. Run the programming script again.



If you use GOWIN EDA or GOWIN Programmer to flash the BRS-100-GW1NR9, ensure that you connect to the board with 'using ftd2xx driver' unselected:



The image shows a 'Cable Setting' dialog box with a light blue header bar containing a logo, the title 'Cable Setting', and standard window controls (help, close). The main area has a light gray background. It contains three dropdown menus: 'Cable:' set to 'USB Debugger A', 'Port:' set to 'USB Debugger A/0/593/null', and 'Frequency:' set to '2.5MHz'. Below these are two buttons: 'Custom LPT' and 'Query/Detect Cable'. At the bottom, there are two checkboxes: 'Do NOT show at startup' and 'using ftd2xx driver', which is highlighted with a red rectangular box. To the right of the checkboxes are 'Save' and 'Cancel' buttons.

4.3 Extending

Extending the project with your custom firmware is quite straightforward, simply modify the [user.sv](https://github.com/BrisbaneSilicon/BRS-100-GW1NR9/blob/public/proj/common/systemverilog/user.sv)²⁵ file (located in <EPM11 repository directory>/EPM11/proj/common/systemverilog/). You can also instantiate your own Systemverilog or VHDL modules, but ensure you add them to the appropriate build script file [synth.tcl](https://github.com/BrisbaneSilicon/BRS-100-GW1NR9/blob/public/proj/common/scripts/synth.tcl)²⁶ ('scripts' directories).



More information will be added to this section in good time.

25. <https://github.com/BrisbaneSilicon/BRS-100-GW1NR9/blob/public/proj/common/systemverilog/user.sv>

26. <https://github.com/BrisbaneSilicon/BRS-100-GW1NR9/blob/public/proj/common/scripts/synth.tcl>

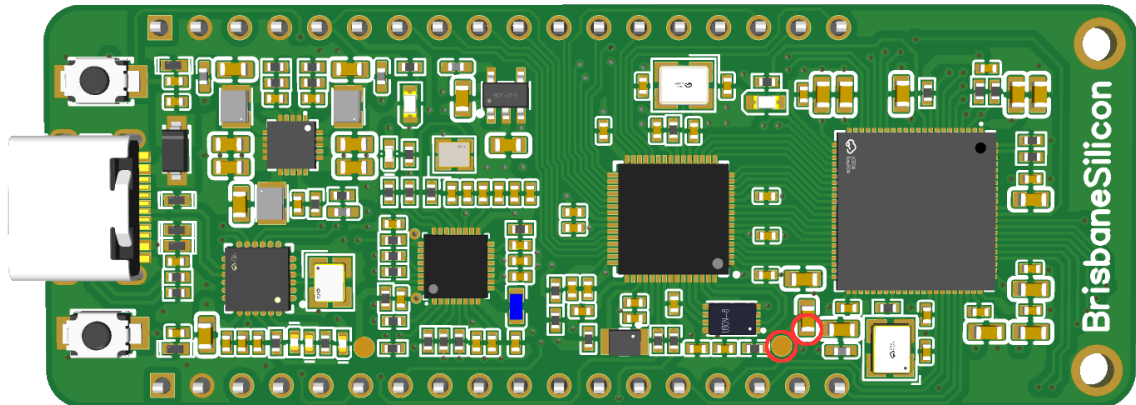
5 MCU Example Project

Coming Soon !

6 Flash RP2350 MicroPython Image

The process to flash a RP2350 MicroPython image to the board is detailed below.

1. Unplug the EPM11 from your PC.
2. Short TP2 to GND. For reference, TP2 and the closest GND pad are circled in the diagram below. A small pair of tweezers or a single-strand wire are suggested to be utilized for this step.



3. Plug the EPM11 into your PC, keeping the TP2 - GND short in place.
4. Once the EPM11 is plugged into your PC, remove the TP2 - GND short.
5. A new folder 'RP2350' should now be mounted by your OS. Simply drag and drop the new MicroPython image to that folder.
6. Once the new image has been uploaded, the 'RP2350' folder will unmount and the image will boot.