B9X ELECTRONICS

B9X Basic Developer Module

Getting Started Guide

PRE-INSTALLED This is an ESP32-S3 development platform with B9X Basic already installed. It is intended to let you move directly into readable embedded programming without treating the board as a blank microcontroller module.

Processor platform	ESP32-S3, dual-core, up to 240 MHz
Flash memory	16 MB
PSRAM	8 MB
Connectivity	2.4 GHz Wi-Fi and Bluetooth Low Energy
Physical format	USB-equipped, 0.1-inch pin spacing, breadboard-friendly
Software	B9X Basic pre-installed

PRODUCT DOCUMENTATION - EDITION AUGUST 2026

01 - KNOW YOUR MODULE

More than a blank ESP32-S3 board

The ESP32-S3 is the processor platform. The product you are holding is a complete breadboard-ready development module with the support hardware and B9X Basic firmware needed to turn that processor into a practical programming platform.

Part of the product	What it gives you
ESP32-S3	Dual-core embedded processing plus Wi-Fi and Bluetooth Low Energy.
16 MB flash	Nonvolatile storage for firmware and project resources.
8 MB PSRAM	Additional working memory for larger embedded applications.
USB connectors and support circuitry	A development-board form factor intended for convenient bench use.
0.1-inch pin spacing	Direct compatibility with common breadboards and 0.1-inch prototyping hardware.
B9X Basic firmware	A source-derived BASIC implementation with control flow, functions, GPIO, sensors, displays, audio, messaging, and connected-service APIs.

WHY B9X BASIC?

B9X Basic is not a generic desktop BASIC transplanted onto a microcontroller. Its language and native functions are implemented for this ESP32-S3 firmware. That means examples should follow B9X Basic syntax, not Visual Basic or QBASIC syntax.

If your package includes the optional B9X expansion board, keep it with the module during setup. The expansion board is an accessory; the B9X Basic Developer Module itself remains the programmable core of the system.

02 - FIRST CONNECTION

Start safely

A few simple habits protect the module and make first-time setup much easier.

Step	Action
1	Place the module on a nonconductive surface or correctly across a solderless breadboard. Check that no pins are shorted together.
2	Use the module's onboard USB connection for the normal development connection. Do not connect external power to random header pins.
3	Allow the host computer to detect the USB device. The exact device/port name depends on the computer and installed drivers.
4	Confirm that the B9X Basic firmware is present before changing or reflashing the board. The product is shipped with B9X Basic pre-installed.
5	Begin with a program that uses PRINT only. Add external wiring after the basic program workflow is confirmed.

IMPORTANT The USB connectors are part of the development module, but this guide does not assign functions to a particular connector because connector labeling can vary by board revision. Follow the markings on the module and the instructions supplied with your specific revision.

Before adding hardware

Check	Reason
Power off before rewiring	Reduces the chance of accidental shorts or pin damage.
Share ground with external circuits	Signals need a common electrical reference.
Use 3.3 V logic levels	ESP32-S3 GPIO is a 3.3 V domain; do not apply 5 V directly to GPIO.
Confirm the exact exposed GPIO	Not every ESP32-S3 GPIO is necessarily available or appropriate on every development-board revision.

03 - CONNECT TO WI-FI

Use the B9X BASIC Boot Menu

Wi-Fi credentials are entered through the module's serial boot menu and stored in nonvolatile memory. You do not have to place your network password inside main.bas.

Step	What to do
1	Connect the module to the computer through the USB connection that provides the serial/UART0 console for your board revision.
2	Open a serial terminal at 115200 baud, then reset or power-cycle the module.
3	When B9X BASIC Boot Menu appears, press W within the 4-second countdown.
4	At Enter SSID:, type the exact Wi-Fi network name and press Enter.
5	At Enter password:, type the Wi-Fi password and press Enter. The firmware displays asterisks instead of the password characters.
6	At Save credentials? (Y/N):, press Y. The SSID and password are stored in NVS.
7	The module continues startup and connects. Look for WiFi connected followed by IP address: x.x.x.x. Write down that IP address; you will use it for FTP.

What the boot menu shows

```
B9X BASIC Boot Menu
W - Configure WiFi
E - Erase stored WiFi
F - Format Flash Drive
B - Continue boot
```

BE CAREFUL WITH F F erases/formats the module's SPIFFS storage. That can remove main.bas and other files. Do not select F merely to change Wi-Fi settings.

To remove saved network credentials later, restart the module and press E during the boot-menu countdown. You can also press W and submit an empty SSID to clear the stored Wi-Fi settings.

04 - COPY MAIN.BAS TO THE MODULE

Program updates can be as simple as copy and paste

Once Wi-Fi is connected, the B9X firmware starts an FTP server backed by its SPIFFS storage. A Windows FTP client such as WinSCP provides a familiar way to copy a new main.bas onto the module.

WinSCP setting	B9X module value
File protocol	FTP
Encryption	No encryption / plain FTP
Host name	The IP address printed by the module after Wi-Fi connects
Port	21
User name	admin
Password	admin
Connection mode	Passive (WinSCP uses passive FTP by default)

Copy and paste your program

Step	Action
1	Create or export your B9X Basic program and make sure the file is named exactly main.bas.
2	Connect WinSCP to the module using the settings above. The FTP root maps to the module's /spiffs storage.
3	In Windows File Explorer, select main.bas and press Ctrl+C.
4	Switch to the WinSCP remote file panel and choose Files > Paste or press Ctrl+V.
5	If main.bas already exists, confirm that you want to replace/overwrite it.
6	Wait for the transfer to complete, then reset or power-cycle the B9X module. The firmware reads /spiffs/main.bas during boot, so the reset loads your new program.

LOCAL NETWORK ONLY

This firmware uses standard unencrypted FTP and the default login admin / admin. Use it on a trusted local network. Do not expose or port-forward the FTP server to the public Internet.

If FTP will not connect

Confirm that the computer and module are on the same LAN, use the current IP printed by the module, choose FTP (not SFTP or FTPS), use port 21, and keep passive mode enabled. The firmware's passive data range is 50000-50100.

Start with something you can see

This first program uses only text output, so it proves the language workflow without depending on external wiring.

```
' My first B9X Basic program
let greeting$ = "Hello from B9X";
counter = 3;
print greeting$;
print counter;
```

What to notice

Syntax	Meaning
' comment	An apostrophe starts a comment that continues to the end of the line.
name$	A trailing dollar sign identifies a string variable.
let name$ = ...;	String assignment requires LET in this B9X Basic grammar.
counter = 3;	Numeric assignment may omit LET.
print expression;	PRINT writes a numeric or string expression to the BASIC output.
;	Most simple statements end with a semicolon.

B9X RULE Use == for equality comparisons. A single = performs assignment. This small difference is important when moving from another BASIC dialect.

Once this program runs successfully, move on to GPIO. Keeping the first test hardware-free separates a programming or connection issue from a wiring issue.

06 - DIGITAL I/O

Control hardware with readable functions

B9X Basic includes direct GPIO functions. Use a GPIO number that is actually exposed and suitable on your module revision.

```
' Example: pulse a user-selected output GPIO
' Replace GPIO_NUMBER with a safe exposed GPIO number.
makeOutput(GPIO_NUMBER);

while 1 do
    setHigh(GPIO_NUMBER);
    wait(500);
    setLow(GPIO_NUMBER);
    wait(500);
wend
```

Function	Purpose
makeInput(gpio)	Configure a positive GPIO as an input with internal pulldown.
getInput(gpio)	Read a digital input and return 0 or 1.
makeOutput(gpio)	Configure a positive GPIO as an output.
setHigh(gpio)	Drive a configured GPIO high.
setLow(gpio)	Drive a configured GPIO low.
wait(milliseconds)	Delay the current BASIC/VM task for the requested time.

PINOUT NOTE

This guide deliberately does not invent a board pinout. Use the silkscreen/labels on your B9X module revision and the seller-supplied pinout for exact header positions. GPIO numbers in example programs are logical ESP32-S3 GPIO numbers, not physical header position numbers.

Match the function to the interface

The module can work with far more than simple digital I/O. Choose pins deliberately and verify that each selected GPIO is available on your board revision before wiring.

Project need	B9X Basic support	Connection planning
Digital input/output	makeInput, getInput, makeOutput, setHigh, setLow	One suitable GPIO per signal plus common ground.
Analog measurement	initADC, readADCRaw, readADCMV	Use ADC-capable GPIO/channel mapping appropriate to the board.
DS18B20 temperature	readTemperature	One suitable GPIO plus sensor power/ground and proper 1-Wire wiring.
SSD1306 OLED	displayInit, displayText	Two selected GPIOs for I2C SDA/SCL; firmware target uses 128x64 at address 0x3C.
WS2812 LEDs	ws2812Init and strip functions	One suitable data GPIO; provide adequate external LED power for larger strips.
Audio	playInit / voice functions	Selected audio/I2S GPIOs; avoid assigning the same peripheral resources twice.

EXPANSION BOARD

If your purchase includes the optional expansion board, treat its labeled connections as the authoritative accessory interface. Do not infer an expansion-board signal from an ESP32-S3 chip pinout found online; the B9X module is a complete board with its own routing and support circuitry.

Breadboard tip

The 0.1-inch header format is intended to make prototyping straightforward. Before applying power, visually trace power, ground, and signal rows on the breadboard. A one-row offset is a common source of first-project faults.

One language, practical embedded features

After the first program and GPIO test, the same language can be used for progressively richer applications.

Area	Available capability
Language	Numeric and string variables, numeric arrays, IF, WHILE, FOR, SUB, numeric FUNCTION, and string-returning FUNCTION.
Math and formatting	Common math functions, random values, string conversion/search, numeric formatting, and time functions.
Sensors and display	GPIO, ADC, DS18B20 temperature, and SSD1306 OLED text.
Lighting and infrared	WS2812 addressable LEDs and infrared remote decoding.
Messaging	MQTT connect, subscribe, unsubscribe, publish, and event polling.
Audio and local speech	WAV playback and SYN6988 text-to-speech support.
Connected services	Text questions, speech transcription, and natural text-to-speech through asynchronous APIs when network service credentials are configured.

DESIGN PATTERN

For asynchronous features, submit work once and then poll for the result with a short WAIT. Do not submit a new network/audio request on every pass through a loop.

B9X Ladder Editor compatibility

B9X Ladder Editor can be used as a visual control-logic companion to B9X Basic. It can export B9X Basic source, making the module useful for both text-based embedded programming and ladder-logic-oriented workflows.

09 - REFERENCE

Specifications and first-line troubleshooting

Specification	B9X Basic Developer Module
MCU platform	Espressif ESP32-S3
CPU	Dual-core, up to 240 MHz
Flash	16 MB
PSRAM	8 MB
Wireless	2.4 GHz Wi-Fi and Bluetooth Low Energy
Development connections	Onboard USB connectors and support circuitry
Header format	0.1-inch spacing; breadboard-friendly
Installed software	B9X Basic firmware
Software platform basis	ESP32-S3 firmware / ESP-IDF v5.5.1 source snapshot documented July 2026

If something does not work

Symptom	Check first
Computer does not see the board	USB cable, connector choice/labeling, host USB port, and required host driver.
Program parse error	Missing semicolon, string assignment without LET, or use of = where == was intended.
GPIO does nothing	Correct GPIO number, output configured with makeOutput, common ground, and pin availability on this board revision.
Unexpected resets	Power quality, accidental shorts, external load current, and wiring.

DOCUMENT BASIS

B9X Basic syntax and function descriptions in this guide are based on the source-derived B9X Basic Language Reference for ESP32-S3 / ESP-IDF v5.5.1 (July 2026). Hardware specifications are the stated specifications of the B9X Basic Developer Module. Exact connector and header routing should be confirmed against the markings/documentation for the board revision shipped with the product.