

TECHNICAL FIELD GUIDE

BIGTREETECH Pi V1.2

ASL3 Guided Installer Manual

Prepared for John, N7OKN

Draft	Manual 0.3.7 Installer 0.2.7-draft 30 August 2026
Reference	W3ICQ node 651805
Validated	Debian 13 Trixie, Linux 7.0, ASL3, UCI80, AllScan
Status	Core path proven; analog-audio investigation remains open

A reproducible methodology for unusual ARM64 single-board computers

Start here: three required stages

The **BTT162-ASL3-Installer.sh** runs these stages in order.

1 PREFLIGHT

Check the clean Debian card and hardware.

```
/root/BTT121-ASL3-Installer.sh --preflight
```

2 INSTALL

Install and configure ASL3.

```
/root/BTT121-ASL3-Installer.sh --install
```

3 VERIFY INSTALL

Test the completed node.

```
/root/BTT121-ASL3-Installer.sh --verify-install
```

Purpose

This package teaches and performs a checkpointed installation of AllStarLink 3 on the BIGTREETECH Pi V1.2 family. It is not a generic Linux script. It depends on the same hardware and operating-system foundation already proved on W3ICQ's system.

The goal is twofold: build a working ASL3 node and demonstrate why unusual SBC work begins with compatibility analysis rather than a random OS image.

Current validation status

The reference system successfully completed:

1. Official BIGTREETECH/CB1 V3.1.0 minimal image
2. Debian 13 Trixie, ARM64, Linux 7.0 vendor kernel
3. ASL3 and Asterisk 22
4. [app_rpt](#) and SimpleUSB
5. AllScan UCI80 radioless interface
6. AllScan web application
7. HTTP registration on a non-default IAX port
8. Parrot node 55553
9. Live ECR QSO

One analog-noise issue remains open and is described below. This installer is therefore a guided technical draft, not yet a claim that every BTT Pi/UCI80 combination is production-perfect.

What the installer will not do

It will not register an AllStar server or node, create portal credentials, change a router, choose a node password, flash an SD card, guess an unknown board revision, or conceal failed checks. It never requests or logs the node password.

Required hardware

10. BIGTREETECH Pi V1.2-family board matching the reference system
11. Reliable microSD card, 8GB minimum
12. Ethernet strongly recommended for initial installation
13. Suitable regulated USB-C power source: **5V/2A minimum; 5V/3A recommended**
14. AllScan UCI80 and compatible speaker-microphone for the reference radioless configuration
15. Proper 2.54mm shunt for J8 when using USB-C power

J8 power warning

POWER SAFETY — When powering the board from USB-C at 5V, use a suitable regulated supply rated **5V/2A minimum; 5V/3A recommended**, and install the proper jumper at J8 before applying power. When powering from the 12–24V screw terminals, remove J8. **Never connect USB-C 5V power and 12–24V terminal power at the same time.** Never change J8 while the board is powered. The manufacturer documents these as mutually exclusive arrangements.

Required operating system

Use the official BIGTREETECH CB1/BTT Pi V3.1.0 **minimal** image, whose release identifies mainline Armbian, Linux 7.0, and Debian 13 Trixie. Preserve the exact downloaded asset filename and checksum with your project notes. Do not substitute a Klipper image, Raspberry Pi image, Ubuntu image, or an unrelated Armbian image merely because it boots.

Official sources:

16. <https://github.com/bigtreetech/BTT-Pi>
17. <https://github.com/bigtreetech/CB1/releases/tag/V3.1.0>
18. <https://allstarlink.github.io/install/debian/install/>

Before running the installer

19. Register the AllStar server and request the node number.
20. Decide which unique UDP IAX port the server will use.
21. Record the callsign, node number, hostname, and port.
22. Flash and boot the exact minimal image.
23. Complete the standard Armbian first-boot account setup.
24. Connect by SSH and confirm Internet access.
25. Copy `BTT121-ASL3-Installer.sh` onto the BTT Pi.

Complete clean-card preparation procedure

This section intentionally includes steps that occur before the script can run. A blank card contains neither Linux nor the installer, so it must first be prepared on another computer.

Preserve any working reference card

If the BTT Pi already has a working card, do not experiment on it.

26. Leave the working card installed long enough to shut Linux down properly:

```
poweroff
```

27. Wait for the activity and Ethernet lights to stop.
28. Disconnect board power.
29. Remove the working card.
30. Label it clearly, for example:

```
BTT ASL3 651805 – WORKING REFERENCE
```

31. Store it somewhere safe. It is both the recovery card and the reference against which the new installation will be compared.

Prepare the clean card on Windows

32. Put a different blank microSD card into a USB card reader connected to the Windows PC. Do not insert an unprepared blank card into the BTT Pi.
33. Open the official [BIGTREE TECH CB1 V3.1.0 release page](#) and expand **Assets 6**.
34. Select this exact 451MB image:

```
CB1_Debian13_minimal_kernel7.0_20260430.img.xz
```

Its filename contains `Debian13`, `minimal`, `kernel7.0`, and ends in `img.xz`. Do not select the larger `Klipper` image and do not select the 65-byte checksum file as though it were the operating-system image.



Figure 1. Select the 451MB Debian 13 minimal image—not the Klipper image.

35. Also download the matching checksum file when you are prepared to verify the download:

CB1_Debian13_minimal_kernel7.0_20260430.img.sha256

36. Record both complete filenames and the verified checksum in the project notes.
37. Open **Armbian Imager** or **balenaEtcher**. Both are suitable multi-board imaging programs and avoid Raspberry Pi-specific terminology.
38. Choose the downloaded **.img.xz** file as a custom image. It does not need to be manually decompressed first when the imaging program accepts **.xz** files.
39. Select the microSD card by its drive identity and capacity. Stop if the target is ambiguous; selecting the wrong disk can erase unrelated data.
40. Write and verify the image.
41. Allow Windows to finish all writes, then safely eject the microSD card.

Exchange cards in the BTT Pi

42. Confirm that BTT Pi power is disconnected.
43. Confirm the working reference card has already been removed and stored safely.
44. Insert the newly flashed card into the BTT Pi.
45. Recheck the power arrangement:
 46. USB-C power requires J8 installed and a regulated **5V/2A minimum; 5V/3A recommended** supply.
 47. 12–24V terminal power requires J8 removed.
 48. Never connect both power inputs simultaneously.
 49. Never move J8 while powered.
50. Connect Ethernet for the initial installation whenever practical.
51. Apply power and allow the first boot to finish.
52. Determine the new LAN address from the router or network-scanning tool.

53. Make the first SSH connection using these temporary factory credentials:

```
User name: root
Temporary password: root
```

The password prompt displays no characters while you type; this is normal. Type `root` and press Enter.

54. Follow the first-login prompts to replace the temporary root password and complete the standard Armbian setup, including the ordinary user, locale, timezone, and hostname. Record the new root password securely.
55. If the setup ends the SSH session or changes the hostname, reconnect using `root` and the new root password.
56. Confirm that the prompt belongs to the newly flashed test card before making further changes.

Transfer the installer with WinSCP

57. Download the current `BTT121-ASL3-Installer.sh` to the Windows PC.
58. Open WinSCP and create an SFTP connection:

```
File protocol: SFTP
Host name: the BTT Pi LAN address
Port: 22
User name: root
Password: the root password created during first boot
```

59. After login, the Windows filesystem is displayed on the left and the BTT Pi on the right.
60. In the right pane, open `/root`. The blue bar above the remote pane shows the current remote path.
61. Locate `BTT121-ASL3-Installer.sh` in the Windows pane.
62. Drag it from the Windows pane into `/root` on the BTT Pi.
63. Return to the SSH window and run:

```
chmod 755 /root/BTT121-ASL3-Installer.sh
/root/BTT121-ASL3-Installer.sh --preflight
```

The first run is deliberately `--preflight`. It establishes that the board, exact OS, architecture, package system, time, storage, and network foundation are suitable before ASL3 is installed. When preflight passes, the script asks whether to continue directly to Step 2. Answer `y` to continue or `n` to stop safely and install later.

Running it

Make the script executable:

```
chmod 755 BTT121-ASL3-Installer.sh
```

Run read-only checks first:

```
sudo ./BTT121-ASL3-Installer.sh --preflight
```

Do not proceed if the script reports **STOP**. Correct the stated foundation problem rather than bypassing it.

After a successful preflight, the script asks:

```
Preflight passed. Continue now to Step 2 – ASL3 installation? [y/N]:
```

Answer **y** to continue immediately. The Step 1 J8/power confirmation and operating-system checks are recorded and are not repeated in Step 2. Answer **n** to return to the prompt; start Step 2 later with the command below.

Start the guided installation:

```
sudo ./BTT121-ASL3-Installer.sh --install
```

If a reboot, disconnect, or deliberate pause interrupts the process:

```
sudo ./BTT121-ASL3-Installer.sh --resume
```

To verify the completed installation:

```
sudo ./BTT121-ASL3-Installer.sh --verify-install
```

--verify-install is the final step, not the clean-card preflight. It detects the installed callsign, node number, IAX bind port, and hostname from the existing system. It does not ask for or save a future operator's proposed identity. The obsolete **--verify** option stops and directs the operator to the correct command.

Reading the results

- 64. **PASS** means the checkpoint met its test.
- 65. **WARNING** means installation may continue, but the item requires review.
- 66. **STOP** means continuing would undermine the experiment or risk a broken configuration.

Logs are stored under:

```
/var/log/btt-asl3-guided-installer/
```

Checkpoint state is stored under:

```
/var/lib/btt-asl3-guided-installer/
```

The answers file contains callsign, node, port, hostname, and interface type—never the node password.

The interactive ASL3 checkpoint

The installer opens `asl-menu` rather than automating the registration password. The official menu does not provide a supported way for this installer to prefill its dialogs. **The Node Number field will be blank even though the number was already entered in this installer.** Enter the same saved value again when `asl-menu` asks for it.

Before opening the menu, maximize or enlarge the MobaXterm terminal. `asl-menu` requires at least **22 visible terminal lines**. The installer checks this before launching the menu. If the terminal is too short, enlarge it and continue with:

The installer connects `asl-menu` directly to the SSH terminal instead of its normal log stream. This is required for the full-screen menu to recognize the real terminal dimensions; the menu session itself is intentionally not copied into the text log.

```
sudo ./BTT121-ASL3-Installer.sh --resume
```

The installer displays the planned non-secret values and also saves them in:

```
/root/BTT121-ASL3-Menu-Values.txt
```

At the prompt `Open asl-menu now?`, answer `y`, then follow this path:

67. Select **Node Settings**.
68. Select **AllStar Node Setup Menu**.
69. Select **Add Node**.
70. Enter the node number displayed by the installer; the menu's field starts blank.
71. Enter the portal-issued node password only inside `asl-menu`.
72. Enter the callsign displayed by the installer.
73. Select **AllScan UCI / SimpleUSB** as the radio interface.
74. Select duplex mode **3** for the radioless UCI80.
75. Select **Yes** for posting node status.
76. Set the server IAX UDP port to the value displayed by the installer if the Add Node sequence did not include it.
77. Save the configuration and exit `asl-menu`.

After the menu closes, the installer does not merely assume success. It verifies the requested node stanza, callsign, SimpleUSB channel, duplex mode 3, HTTP registration, and IAX UDP port. If any value is missing or different, it reports the precise item and stops so the operator can correct it and run `--resume`.

Whenever `asl-menu` closes, the installer gives an explicit choice: **1 — Edit asl-menu again** or **2 — Resume installation**. If the subsequent checks find a missed or incorrect value, the installer offers to reopen `asl-menu` immediately instead of leaving the operator at an unexplained shell prompt.

After those saved-file checks pass, the installer restarts Asterisk, waits for it to become active, and confirms that Asterisk is actually listening on the selected IAX UDP port. This restart is necessary because changing `bindport` in `iax.conf` does not move the port of an already-running Asterisk process.

After saving, verify that COS changes from clear to keyed and PTT changes correspondingly when the speaker-mic PTT is pressed. Do not tune audio until these logical controls work.

AllScan

The script installs Apache before running the official AllScan installer. This order matters: installing PHP before the complete web server can leave Apache serving PHP source as text. The script enables the appropriate Apache PHP module and explicitly checks that source is not exposed.

Some SSH terminals fail to display the official AllScan installer's questions. The guided installer therefore explains and supplies three fixed answers after one visible confirmation: install AllScan, skip a redundant full operating-system upgrade, and copy the optional DTMF support files. On resume, it also detects an existing `/var/www/html/allscan/index.php` and does not reinstall AllScan.

The installer assigns `/var/www/html/allscan/user` to Apache's `www-data` group and confirms group write access. It then follows the initial AllScan redirect to the actual user page and requires a complete HTTP 200 AllScan document. Accepting the initial HTTP 302 alone can conceal a PHP permissions failure on the destination page.

AllScan account credentials are separate from Linux, the AllStar portal, the node registration password, and Asterisk Manager credentials.

Actions deliberately left for supervised completion

The first draft does not automatically rewrite `iax.conf` or `rpt.conf`. After core verification, perform these with a knowledgeable reviewer:

78. Disable unused `iaxrpt`, `iaxclient`, and `allstar-public` access.
79. Retain `[radio]`, `[allstar-sys]`, and the final custom include.
80. Enable local telemetry with COP 33.
81. Enable foreign-link local announcements with COP 36.
82. Make the telemetry controls persistent through a checked startup macro.
83. Forward the selected UDP IAX port in the router.
84. Confirm the external registration reports the intended public port.

These steps are intentionally supervised until the generated configurations are validated on more than one node.

Successful-installation screen

After every automated installation checkpoint passes, the installer finishes with a bright-green, all-capitals congratulations panel. It identifies the installed node, prints both the hostname and LAN-address AllScan links, and directs the operator to make the first Parrot test at node 55553. This panel is displayed only after installed-node verification succeeds.

Acceptance test

85. Asterisk is enabled and active.
86. `app_rpt.so` and `chan_simpleusb.so` show `Running`.
87. All UCI80 USB components are detected.
88. COS and PTT behave correctly.
89. Node registration shows `Registered`.
90. Asterisk listens on the intended UDP port.
91. AllScan renders normally.
92. Connect to node 55553.
93. Transmit silence, then speech, and listen to the replay.
94. Make a short real-network test only after Parrot succeeds.
95. Reboot and repeat registration, telemetry, and Parrot checks.

Open static investigation

This investigation is presented after Step 3 verification, not during the clean-card preflight.

On W3ICQ's BTT Pi reference system, Parrot playback contains distracting low-frequency static whenever the Retevis speaker-microphone is connected. Controlled tests established:

96. The same UCI80, speaker-mic, and cable are quiet on an Orange Pi.
97. Static exists in raw ALSA capture, proving Parrot and Asterisk do not create it.
98. Raw capture is essentially quiet with the speaker-mic disconnected.
99. Muting the UCI80 speaker playback does not eliminate it.
100. Changing BTT USB ports does not eliminate it.
101. Two external power supplies, power-cable ferrites, and a powered USB hub do not eliminate it.
102. Moving and touching the temporary J8 bridge does not alter it.

John's BTT Pi is valuable as an independent comparison. Record whether it has the same noise, the exact board revision, power arrangement, J8 jumper, kernel, USB topology, UCI80, and microphone model. Do not "tune away" the noise before recording the baseline.

Recovery

The script is designed to stop rather than force its way past errors. Configuration work must be backed up before edits. Installer checkpoint data can be cleared without uninstalling ASL3:

```
sudo ./BTT121-ASL3-Installer.sh --reset-state
```

This does not remove ASL3, Asterisk, AllScan, logs, or operator configuration.

Feedback record

When reporting results, include:

103. Board model and printed revision
104. Image release and exact filename
105. `uname -a`
106. Node number with password omitted
107. Interface and microphone model
108. Power method and J8 state
109. Installer log
110. Parrot result
111. Whether silent Parrot playback contains static

That evidence lets the next revision improve the method rather than merely accumulate fixes.