BE-IIS Quick Reference

Installer, HAT+/HAT++, Journal und Repository-Struktur

Für Raspberry Pi OS: Installation aus Git, automatische HAT++-Erkennung nach dem Boot und Diagnose mit normalen Linux-Werkzeugen.

1. Installer und Repository

Komplettinstallation

`sudo apt install -y git`	Git installieren
`git clone` https://github.com/be-iis/be-iis-installer.git	Repository holen:
`cd be-iis-installer`	Repository öffnen
`sudo ./scripts/install/install-all.sh`	Alle Komponenten installieren
`sudo reboot`	Overlays, Module und Dienst aktivieren

Was der Installer einrichtet

`01_install_eepdump.sh`	HAT-EEPROM lesen
`02_install-mods.sh`	Benötigte Kernelmodule
`03_install-overlays.sh`	Device-Tree-Overlays
`04_install_hatpp_service.sh`	HAT++-Systemdienst
`05_install-udev-rules.sh`	Stabile Gerätenamen
`06_I2C_VC_ena.sh`	HAT-ID-I^{2}C aktivieren

GitHub-Struktur

`common/`	Gemeinsame Shell-Funktionen
`docs/`	Anleitungen und technische Doku
`examples/`	Beispielkonfigurationen
`platform/`	OS-Integration, systemd, udev
`products/`	Overlays, Skripte und Tests je Board
`scripts/`	Installation, Netzwerk und SSH
`tools/`	EEPROM-, Kernel- und Overlay-Helfer

Typischer Produktordner

`overlays/src/rpi/`	DTS-Quellen und Makefile
`overlays/build/rpi/`	Kompilierte DTBO-Dateien
`scripts/`	Board-Setup und Konfiguration
`systemd/ / udev/`	Optionale Integration
`test/`	Produktions- und Funktionstests

2. HAT+ und HAT++

Instanzen und ID-Adressen

I	0x50	Standard-HAT+; beim Boot erkannt
II	0x60	Nach dem Boot durch Dienst geladen
III	0x70	Nach dem Boot durch Dienst geladen
IV	0x74	Nach dem Boot durch Dienst geladen
V	0x76	Nach dem Boot durch Dienst geladen

Jeder Stack benötigt genau eine Instanz I. Weitere Boards erhalten unterschiedliche Instanzen II bis V. Taste wählen, LEDs kontrollieren.

Automatischer Boot-Ablauf

1. Raspberry-Pi-Firmware erkennt Instanz I als normales HAT+.
2. `be-iis-hatpp.service` scannt die zusätzlichen ID-Adressen.
3. `eepdump` liest den Overlay-Namen aus dem HAT-Datensatz.
4. `dtoverlay` lädt die passende Instanzvariante.
5. Kernelmodule und udev-Namen werden bereitgestellt.

Erkennung prüfen

`i2cdetect -y 0`	ID-Adressen 50/60/70/74/76 suchen
`ls /sys/bus/i2c/devices/0-00{50,60,70,74,76}`	Erkannte HAT-ID-Geräte
`dtoverlay -l`	Laufzeit-Overlays anzeigen
`lsmod | grep -E 'lan865|adin|mcp251|sc16'`	Typische Treibermodule prüfen
`ls -l /dev/beiis-*`	Stabile UART/Modbus-Gerätenamen
`ip -br link`	Netzwerk- und CAN-Interfaces

Wichtige HAT++-Regeln

1. Instanzen dürfen sich im selben Stack nicht wiederholen.
2. Nicht verwendete `spidev`-Knoten müssen frei bleiben, damit zusätzliche Chip-Selects später geladen werden können.
3. Nach Installer-, Overlay- oder Boot-Konfigurationsänderungen neu starten.
4. Produktdateien liegen unter `products/<board>/`; gemeinsame Plattformdateien nicht duplizieren.

3. systemd, Journal und Fehleranalyse

BE-IIS-Dienst und Journal

`systemctl status be-iis-hatpp.service`	Dienststatus und letzte Logs
`systemctl is-enabled be-iis-hatpp.service`	Boot-Aktivierung prüfen
`sudo journalctl -u be-iis-hatpp.service -b`	Dienst-Logs dieses Boots
`sudo journalctl -b | grep 'BE-IIS'`	Komplette HAT++-Erkennung
`sudo journalctl -u be-iis-hatpp.service -f`	Logs live verfolgen
`sudo systemctl restart be-iis-hatpp.service`	Erkennung erneut ausführen

Dienste steuern – Beispiele

`systemctl status ssh.service`	Status anzeigen
`sudo systemctl start ssh.service`	Jetzt starten
`sudo systemctl stop ssh.service`	Jetzt stoppen
`sudo systemctl restart NetworkManager.service`	Neu starten
`sudo systemctl enable ssh.service`	Beim Boot starten
`sudo systemctl disable ssh.service`	Boot-Start entfernen
`sudo systemctl enable -now my-app.service`	Aktivieren und sofort starten
`systemctl -failed`	Fehlgeschlagene Units

Unit-Dateien, Targets und Timer

`systemctl list-units -type=service`	Geladene Service-Units
`systemctl list-unit-files -type=service`	Installierte Unit-Dateien
`systemctl cat my-app.service`	Effektive Unit anzeigen
`sudoedit /etc/systemd/system/my-app.service`	Eigene Unit-Datei bearbeiten
`sudo systemctl daemon-reload`	Geänderte Units neu einlesen
`systemctl get-default`	Standard-Boot-Target
`systemctl list-units -type=target`	Geladene Targets
`systemctl list-timers -all`	Timer und nächste Ausführung

Typische Fehleranalyse

1. `systemctl status be-iis-hatpp.service`
2. `journalctl -xeu be-iis-hatpp.service`
3. `systemctl cat be-iis-hatpp.service`
4. `dtoverlay -l` und `i2cdetect -y 0`
5. `dmesg | grep -Ei 'BE-IIS|lan865|adin|mcp251|sc16'`
6. Instanz-LEDs, Verkabelung und Neustart prüfen.

Wichtig: Vier Dinge auseinanderhalten

BE-IIS
BRECHEL ELECTRONIC
Industrial Interface Solutions

`start` / `stop`

Ändert nur den aktuellen Laufzeitzustand.

Instanz I

Normale HAT+-Erkennung durch die Raspberry-Pi-Firmware.

`enable` / `disable`

Ändert den automatischen Start beim Booten.

Instanzen II–V

Nach dem Boot durch `be-iis-hatpp.service` erkannt und geladen.