Steps to install ATX software environment from scratch

NOTE – ATX Switch operation requires a custom ATX interface connector which therefore necessitates a DIY PiKVM hardware build – standard PiKVM units (v3 / v4) will not support this operation without re-purposing the ATX interface.

1. Install new base PiKVM image to SD card – use latest v3 box image as will be installing into own DIY box which has OLED / FAN / RTC incorporated. If not using this case / setup then use v2 image for HDMI-CSI bridge or HDMI-USB dongle – whichever is your setup / preference.
2. Insert SD card into Pi 4b and boot. This assumes that the unit has been assembled already (see separate instructions).
3. Once the system has completed its first boot routine, open TTY session and login to unit – IP address should be displayed on OLED screen, but, if necessary, determine the assigned DHCP address from your network / router.
4. The following steps are not strictly necessary for ATX support but might as well do them first (this assumes that you are using the DIY build that includes v3 functional components – if not then can skip them):
    a. Place file system into read/write mode by executing command - **rw**
    b. [Optional now] The latest V3 Box release appears to have fixed the fan operation, but if have issues, or want to set minimum speed lower then update **/etc/kvmd/fan.ini** to have following values:

    ```
    [main]
    pwm_pin = 12

    [speed]
    idle = 20
    low = 20

    [server]
    unix = /run/kvmd/fan.sock
    unix_rm = 1
    unix_mode = 666
    ```

    c. Update **/boot/config.txt** to change Clock dtoverlay to be dtoverlay=i2c-rtc,ds1307
    d. Disable the watchdog service - **systemctl disable --now kvmd-watchdog**
    e. Rotate the OLED display (if required – mine did!):
        i. Use the following code snippet provided on Discord by 'The Adam Bomb' on 06/06/2023 20:10 ->

        ```
        cat << EOF > /usr/lib/systemd/system/kvmd-oled-rotate.service
        [Unit]
        Description=PiKVM - A small OLED daemon
        After=systemd-modules-load.service
        ConditionPathExists=/dev/i2c-1

        [Service]
        Type=simple
        Restart=always
        RestartSec=3
        ExecStartPre=/usr/bin/kvmd-oled --interval=3 --rotate=2 --clear-on-exit --
        image=/usr/share/kvmd-oled/hello.ppm
        ExecStart=/usr/bin/kvmd-oled --rotate=2 --clear-on-exit
        TimeoutStopSec=3
        ```

```
[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload
systemctl disable kvmd-oled.service
systemctl enable kvmd-oled-rotate.service
systemctl start kvmd-oled-rotate.service
```

[Note – I had to modify his script to change the 'rotate' values to 2 as shown above]

5. Make sure that I2C group is present, and if not then add it:
   a. **cat /etc/group**
   b. **groupadd i2c**
6. Make sure that 'kvmd' user is part of i2c group - **gpasswd -a kvmd i2c**
7. Install i2c-tools by following the steps at https://arcanesciencelab.wordpress.com/2014/02/02/bringing-up-i2c-on-the-raspberry-pi-with-arch-linux/
8. Reboot the system and ensure that changes have taken effect:
   a. Fan should run at reduced speed
   b. **timedatectl** should show that RTC time is operational and matches Local / Universal time. Note – may need to do additional configuration if need to incorporate RTC in local timezone.
   c. **I2cdetect -y 1** should show addresses of installed devices: hex addresses of any installed ATX switches (usually starting at 0x20); 0x3c for OLED screen; and UU at address 0x68 for the RTC.
9. **Ensure that the PiKVM operates correctly (with KVM switch if present) before starting ATX environment installation**. **There is no point in trying to configure / debug ATX Switch operations if the basic PiKVM operation (including any KVM switch) is not working as it should!!**
10. Once the PiKVM system is operating correctly then continue with following steps:
11. Switch filesystem back to read/write mode for now - **rw**
12. Create directory **/etc/atx_switch** with 755 permissions. This directory will contain the majority of the ATX environment scripts and initialisation / mapping YAML files (as necessary – dependent upon build).
13. The scripts that will be placed here will depend upon the system configuration and how the environment is to be presented in the PiKVM WebUI. For the purpose of this exercise, we will assume that there is one 4-port ATX switch in the environment (located at the first I2C address – 0x20) and that selection of the active port will be performed via buttons in the 'Source' menu tab of the WebUI, with active indication being shown via associated LED symbols.
14. To enable switching between ATX ports from the WebUI we need to include the following scripts in the /etc/atx_switch directory (ensure that they all have 755 permissions):
    a. **atx_change_port.py** – main functional operations
    b. **selectx.py** – where 'x' is the number of the ATX port assigned within your environment. In the above case this will involve: **select1.py**; **select2.py**; **select3.py** and **select4.py**. These scripts basically call **atx_change_port.py** with a mapped ATX switch address and port number. The scripts themselves are called from the PiKVM

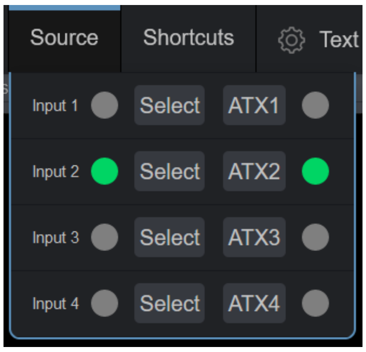
‘**cmd**’ api as defined in the **override.yaml** file. Additional selectx.py scripts and associated override.yaml entries will be required for more ports / switches.

   c. **test_port_led.py** – verifies whether a particular port is currently selected from the information held in the **atx_operational.yaml** file (see later).
   d. **port_test_x.py** - where ‘x’ is the number of the ATX port assigned within your environment. Again, in the above case this will involve: **port_test_1.py**; **port_test_2.py**; **port_test_3.py** and **port_test_4.py**. These scripts basically call **test_port_led.py** with a mapped ATX switch address and port number. The scripts themselves are called from the PiKVM ‘**cmdret**’ api as defined in the **override.yaml** file. Additional port_test_x.py scripts and associated override.yaml entries will be required for more ports / switches.

15. Create the **atx_switch_initialisation.yaml** file for the system and include in the **/etc/atx_switch** directory. The initialisation YAML file provides configuration information on any ATX switches that are connected into the system – how many switches present; and for each switch: the i2c address (in hex format); number of ports; how ATX operations are to be performed; whether to test for a daughterboard; and whether to run a port select test (effectively polling through each port on switch to get visual indication that it works). A debug flag can also be set which logs more detailed information during startup.
    a. Note (1) – the ATX operations setting is for software usage and is dependent upon the physical settings of the jumpers on the ATX Switch main board. e.g. the **atx_ops** flag can be set to i2c but if the jumpers are set to **RJ** (buffered output from ATX connector) then PiKVM operations will still be performed using the standard GPIO mechanism.
    b. Note (2) – the i2c approach to performing ATX operations is not implemented as yet! So should only use GPIO operations for now.
16. Place the **initialise_atx_switches** script (no .py extension) into **/bin** with 755 permissions.
17. Place **initialise_atx_switches.service** file into **/usr/lib/systemd/system**
18. Enable the service for startup – **systemctl enable initialise_atx_switches.service**
19. Create the **override.yaml** file for the system configuration (and user requirements for presentation / operation within the WebUI). Place this file into **/etc/kvmd**
    a. The sample override.yaml file provided is configured to work with a single 4-port ATX Switch and an XH_HK4401 KVM switch – one of the generic devices from Aliexpress e.g. https://docs.pikvm.org/xh_hk4401/

       Although I used the BLI Store version as it incorporated the USB-TTL cable required to enable programmatic switching from the Raspberry Pi (and they provide support). https://www.aliexpress.com/item/1005005149590635.html?pdp_npi=4%40dis%21GBP%21%EF%BF%A146.48%21%EF%BF%A146.48%21%21%21%21%21%40210384b217045198480158768e64a1%2112000031866592220%21sh%21UK%214412103182%21&spm=a2g0o.store_pc_allItems_or_groupList.new_all_items_2007614540057.1005005149590635

    b. The sample file presents in the PiKVM WebUI similar to:



The ‘Select’ buttons allow choice between the 4x KVM ports whose selection status is shown via the ‘Input x’ LED representations. The ATX switch ports are selected via the ‘ATXx’ buttons with selection indicated by the adjacent LED representation.

c. The **override.yaml** file indicates the PiKVM API commands being used and the associated scripts (with relevant path). The associated ATX switch address and port number being specified in the defined scripts. Need to change values there to match system configuration.
    i. A sample override.yaml file has been provided (**override_without_kvm_mapping.yaml**). I you want to use this then copy it to /etc/kvmd and rename it to **override.yaml**. Note – this assumes that a XH_HK4401 KVM Switch is present (see below).
d. Note (1) – at this stage there is no direct integration between the KVM switch selection and the associated ATX port. Both have to be chosen manually. This can be beneficial in more complex scenarios where KVM switches are chained, or integration is not supported.
e. Note (2) – integration has only been performed with XH_HK4401 KVM Switch (as that is the device I have). If integration with other switches (e.g. ezcoo and tesmart) is required then someone with these devices will need to do the work. I assume that the approach used for the XH_HK4401 should be able to be re-used but am unable to confirm / test!
f. Note (3) – without integration the KVM switch may select the last used port during startup (the XH_HK4401 does) but the ATX switch will initialise with no port selected – so the user will need to select this as required.

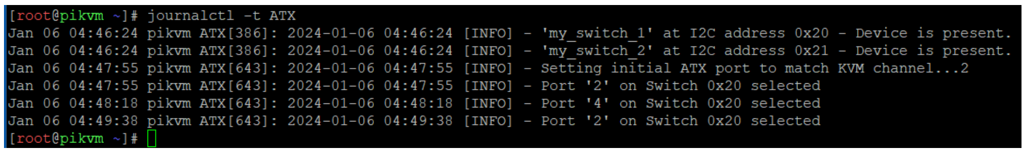
20. Reboot the Pi controller
21. If everything has worked correctly then should be able to login to the PiKVM WebUI at the IP address of the DIY PiKVM unit. Can then view ATX switch port status, and switch between them from the GUI with the relevant LED representation being illuminated (physical operation can be confirmed via LED on associated port on ATX Switch).

    Note – this assumes that the physical environment has all been configured and connected correctly:

    a) DIY PiKVM ATX port connected to ATX Switch input;
    b) pass-thru connections to other ATX Switches (if installed);
    c) power to ATX switches;
    d) ATX cables connected from ATX Switch outputs to the ATX adapter board on associated target PCs / servers;
    e) ATX adapter boards connected to ATX motherboard headers for Power switch / Reset switch / HDD LED / Power LED;
    f) ATX Adapter boards connected to PC / server case harness (if required) for above functions;
    g) jumpers on ATX switches configured as required for proposed system operation (assume everything via GPIO for now);
    h) address of ATX Switch(es) configured via associated unit rocker switch.

22. ATX operations / status on target device should now be available via the ATX menu in the WebUI.
23. If unable to login to WebUI, or ATX operations do not work as expected, then will need to debug the PiKVM environment.
    a. Login to DIY PiKVM unit via TTY and review **journalctl** messages for errors.

i. **journalctl -t ATX** will show any messages that have been logged for the ATX Switch environment specifically e.g.



```
[root@pikvm ~]# journalctl -t ATX
Jan 06 04:46:24 pikvm ATX[386]: 2024-01-06 04:46:24 [INFO] - 'my_switch_1' at I2C address 0x20 - Device is present.
Jan 06 04:46:24 pikvm ATX[386]: 2024-01-06 04:46:24 [INFO] - 'my_switch_2' at I2C address 0x21 - Device is present.
Jan 06 04:47:55 pikvm ATX[643]: 2024-01-06 04:47:55 [INFO] - Setting initial ATX port to match KVM channel...2
Jan 06 04:47:55 pikvm ATX[643]: 2024-01-06 04:47:55 [INFO] - Port '2' on Switch 0x20 selected
Jan 06 04:48:18 pikvm ATX[643]: 2024-01-06 04:48:18 [INFO] - Port '4' on Switch 0x20 selected
Jan 06 04:49:38 pikvm ATX[643]: 2024-01-06 04:49:38 [INFO] - Port '2' on Switch 0x20 selected
[root@pikvm ~]#
```

ii. **journalctl** and **journalctl -fu kvmd** can be used to look through the various steps that the PiKVM system goes through to startup / operate. Any errors / problems during startup (or operation) should be flagged here – typically issues with the ATX Switch environment relate to permissions issues or scripts not being present in correct place. However, they could be physical if environment not connected / powered correctly.

iii. **I2cdetect -y 1** can be used to ensure that the DIY PiKVM unit can see the configured ATX switches.

iv. Other Linux / PiKVM API utilities can be used for more detailed debugging if required: i2cset /i2cget; https://docs.pikvm.org/api/ ; potentially https://www.acmesystems.it/gpiod (although in this case you cannot directly address the GPIO pins reserved by PiKVM, but info commands do work).

v. If necessary, the **debug** flag can be set in the **atx_switch_initialisation.yaml** file (which will provide further debug data during startup) and the system rebooted.

b. Fix any determined configuration issues and restart system.

Integration with XH-HK4401 HDMI KVM Switch

24. Create **atx_kvm_mapping.yaml** file and place in **/etc/atx_switch**
    a. This mapping file consists of key pairs that map HDMI KVM Switch 'channel' numbers to an associated ATX Switch port (that consists of a switch address and port number).
25. Place the **change_atx_from_kvm.py** script into **/etc/atx_switch** with 755 permissions.
26. Place the **atx_xh_hk4401.py** PiKVM device driver into **/usr/lib/python3.11/site-packages/kvmd/plugins/ugpio**
27. Modify the **override.yaml** file to change the driver type for the HDMI KVM Switch to **atx_xh_hk4401** – if desired can also change name of driver to **atxhk:** or alternative (although I just left it as **hk:**)
    a. Again a sample override.yaml file has been provided (**override_with_kvm_mapping.yaml**). As above, copy this to /etc/kvmd and rename to override.yaml if desired.
28. Reboot the system and login to the WebUI – the ATX port associated with the previously chosen HDMI KVM channel should be selected (highlighted)
29. Switching the HDMI KVM switch to another channel, via the WebUI, will also switch the ATX selection to the associated ATX Switch port.
30. Debug if necessary.