Okay, I have reviewed your INI and HAL files along with the LinuxCNC documentation. There are several errors and areas needing correction, particularly regarding the interaction between LinuxCNC and the Remora firmware on your NVEM controller.
Here are the corrected files and the necessary extra steps:

Corrected my-lathe.ini

```
# Corrected INI file for a standard XZ Lathe using Remora
[EMC]
# Version matches the documentation provided
VERSION = 1.1
# Standard task executable
TASK = milltask
# COORDINATES should match the physical axes (XZ for typical lathe)
COORDINATES = XZ
# KINEMATICS should match COORDINATES
KINEMATICS = trivkins coordinates=XZ

[DISPLAY]
# Use axis GUI
DISPLAY = axis
# Add LATHE = 1 for correct lathe display in AXIS
LATHE = 1
POSITION_OFFSET = RELATIVE
POSITION_FEEDBACK = ACTUAL
MAX_FEED_OVERRIDE = 1.5
MAX_SPINDLE_OVERRIDE = 1.5
DEFAULT_LINEAR_VELOCITY = 10 # Units/sec, adjust as needed
MAX_LINEAR_VELOCITY = 100 # Units/sec, adjust as needed
MIN_LINEAR_VELOCITY = 0.1 # Units/sec, adjust as needed
EDITOR = gedit
# Use ~ for home directory path
OPEN_FILE = ~/linuxcnc/nc_files
PROGRAM_PREFIX = ~/linuxcnc/nc_files

[RS274NGC]
PARAMETER_FILE = linuxcnc.var
# Set default plane to XZ for lathe
RS274NGC_STARTUP_CODE = G18 G21 G90 G40 G49 G64 P0.001 G80 G94 G97
# Ensure 'macros' directory exists in your config folder or provide
full path
SUBROUTINE_PATH = macros
# Keep remapping if m53.ngc and m54.ngc exist and are needed
REMAP=M53 modalgroup=10 ngc=m53
REMAP=M54 modalgroup=10 ngc=m54

# Add [EMCMOT] section for motion controller timing
[EMCMOT]
```

```
# Base period for software stepgen (if used, less critical with
Remora)
# Adjust based on latency test if needed, but Remora handles fast
pulses
BASE_PERIOD = 50000
# Servo period for motion planning and Remora communication (1ms =
1000000ns is common)
SERVO_PERIOD = 1000000

[HAL]
# List HAL files to load
HALFILE = custom.hal
# Load post-GUI file if you have GUI-specific HAL components (like
PyVCP/GladeVCP)
# POSTGUI_HALFILE = custom_postgui.hal
# Load HAL user interface component if using external buttons/MPG
connected via HALUI
HALUI = halui
# Remove incorrect STEPGEN entry

[TRAJ]
# Match EMC section
COORDINATES = XZ
LINEAR_UNITS = mm
ANGULAR_UNITS = deg # Only relevant if you add rotary axes later
# These velocities/accelerations are for the Trajectory planner
# Ensure they are achievable by ALL axes combined. Units are per
SECOND.
# Adjust these values based on testing. 50 mm/s = 3000 mm/min. 500
mm/s^2 is moderate.
DEFAULT_LINEAR_VELOCITY = 20
MAX_LINEAR_VELOCITY = 50
DEFAULT_LINEAR_ACCELERATION = 500
MAX_LINEAR_ACCELERATION = 1000

[KINS]
# Match EMC section
KINEMATICS = trivkins coordinates=XZ
# Set JOINTS to the number of controlled axes (2 for XZ lathe)
JOINTS = 2

# --- Define Axis Sections ---
# Max velocity/accel must be >= corresponding JOINT max velocity/accel
[AXIS_X]
MAX_VELOCITY = 50
MAX_ACCELERATION = 1000
MIN_LIMIT = -50 # IMPORTANT: Set based on your machine's X travel
limit
```

```
MAX_LIMIT = 5   # IMPORTANT: Set based on your machine's X travel
limit

[AXIS_Z]
MAX_VELOCITY = 50
MAX_ACCELERATION = 1000
MIN_LIMIT = -5  # IMPORTANT: Set based on your machine's Z travel
limit
MAX_LIMIT = 200 # IMPORTANT: Set based on your machine's Z travel
limit

# --- Define Joint Sections ---
# Must have one section per joint defined in [KINS]JOINTS
[JOINT_0] # Corresponds to X axis in trivkins coordinates=XZ
TYPE = LINEAR
# IMPORTANT: Calculate SCALE based on motor steps, microstepping,
gearing, leadscrew pitch
# Example: 200 step/rev * 16 microstep / 5mm/rev = 640 steps/mm
SCALE = 640.0
MAX_VELOCITY = 50         # mm/sec - Tune this value!
MAX_ACCELERATION = 1000   # mm/sec^2 - Tune this value!
# Set MIN/MAX_LIMIT slightly wider than AXIS limits to avoid issues
MIN_LIMIT = -51
MAX_LIMIT = 6
FERROR = 1        # Max following error in mm
MIN_FERROR = 0.25 # Min following error in mm
# --- Homing settings for JOINT_0 (X) ---
# IMPORTANT: Configure these based on your switch locations and
desired home position
HOME = 0.0                    # Final position after homing
HOME_OFFSET = 5.5             # Position coordinate when switch/index
triggers (relative to final machine zero)
HOME_SEARCH_VEL = -20.0       # Speed/direction to search for switch
(-ve moves towards MIN_LIMIT)
HOME_LATCH_VEL = 5.0          # Speed/direction to approach switch
accurately after backing off
HOME_USE_INDEX = NO           # Set YES if using encoder index pulse
HOME_IGNORE_LIMITS = YES      # Set YES if home switch is also a
limit switch
HOME_SEQUENCE = 0             # Homing order (0 or 1 for XZ lathe,
e.g., home Z first)

[JOINT_1] # Corresponds to Z axis in trivkins coordinates=XZ
TYPE = LINEAR
# IMPORTANT: Calculate SCALE based on motor steps, microstepping,
gearing, leadscrew pitch
SCALE = 640.0
MAX_VELOCITY = 50         # mm/sec - Tune this value!
```

```
MAX_ACCELERATION = 1000   # mm/sec^2 - Tune this value!
# Set MIN/MAX_LIMIT slightly wider than AXIS limits
MIN_LIMIT = -6
MAX_LIMIT = 201
FERROR = 1
MIN_FERROR = 0.25
# --- Homing settings for JOINT_1 (Z) ---
# IMPORTANT: Configure these based on your switch locations and
desired home position
HOME = 200.0
HOME_OFFSET = 200.5
HOME_SEARCH_VEL = -20.0
HOME_LATCH_VEL = 5.0
HOME_USE_INDEX = NO
HOME_IGNORE_LIMITS = YES
HOME_SEQUENCE = 1 # Home Z after X (or vice-versa if X=1, Z=0)

# --- Optional Spindle Section ---
# Add this section if Remora controls your spindle speed
# [SPINDLE_0]
# MAX_FORWARD_VELOCITY = 4000  # Max RPM
# MIN_FORWARD_VELOCITY = 300   # Min RPM
# Add OUTPUT_SCALE, PWM settings etc. if using Remora PWM for speed
control

# Removed invalid sections: [NUM_MOTORS], [AXES], [JOINTS], [OUTPUTS]
```

Corrected custom.hal

```
# Corrected HAL config file for NVEM controller board running Remora
firmware
# Configured for 2 joints (X, Z axes) for a standard lathe

# Load the Remora Ethernet interface component
# Ensure the component name matches your Remora setup (e.g.,
remora-eth)
# Use '-W' to wait for the component to be ready
loadusr -W remora-eth remora-eth

# --- Base/Servo Thread Setup ---
# Create threads (adjust periods if needed, match INI [EMCMOT])
loadrt threads name1=servo-thread period=1000000 # 1ms servo thread

# Add Remora functions to the servo thread
# IMPORTANT: Verify these function names with Remora documentation
# Common names might be remora.read / remora.update / remora.write
addf remora.read servo-thread
```

```
addf remora.write servo-thread
# addf motion-controller servo-thread # Motion controller must be
added

# --- Estop and Enable Chain ---
# Connect LinuxCNC enable/reset to Remora enable/reset
net user-enable-out <= iocontrol.0.user-enable-out
net user-request-enable <= iocontrol.0.user-request-enable =>
remora.reset

# Connect Remora status back to LinuxCNC estop input chain
# This assumes remora.status reflects the board's E-Stop state
net remora-status <= remora.status => iocontrol.0.emc-enable-in
# If using an external E-Stop button connected to a Remora input
(e.g., input 11):
# net estop-ext <= remora.input.11.in => iocontrol.0.emc-enable-in
# Make sure only ONE signal drives iocontrol.0.emc-enable-in

# --- Joint/Axis Connections ---
# Connect LinuxCNC JOINT commands/feedback to Remora STEPGEN pins
# IMPORTANT: Verify Remora stepgen numbers (e.g., 00, 02) match your
Remora config for X and Z

# Joint 0 (X Axis) -> Remora Stepgen 00 (Example)
net x-pos-cmd joint.0.motor-pos-cmd => remora.stepgen.00.position-cmd
# Corrected
net x-pos-fb joint.0.motor-pos-fb <= remora.stepgen.00.position-fb  #
Corrected
net x-enable joint.0.amp-enable-out => remora.stepgen.00.enable   #
Corrected

# Joint 1 (Z Axis) -> Remora Stepgen 02 (Example)
net z-pos-cmd joint.1.motor-pos-cmd => remora.stepgen.02.position-cmd
# Corrected
net z-pos-fb joint.1.motor-pos-fb <= remora.stepgen.02.position-fb  #
Corrected
net z-enable joint.1.amp-enable-out => remora.stepgen.02.enable   #
Corrected

# Enable Remora board when motion is enabled (connects ALL joint
enables)
net motion-enable <= joint.0.amp-enable-out
net motion-enable <= joint.1.amp-enable-out
net motion-enable => remora.enable # Assumes remora.enable enables all
outputs

# --- Spindle Connections (Example, uncomment and modify if needed)
---
```

```
# Spindle Feedback via Remora Encoder 00 (Example)
# net spindle-velocity <= remora.encoder.00.velocity => spindle.0.revs
# net spindle-velocity => spindle.0.speed-in # Use spindle.0.revs for
feedback if possible

# Spindle Speed Control via Remora PWM 00 (Example)
# setp remora.pwm.00.scale [SPINDLE_0]OUTPUT_SCALE # Set scale in INI
# net spindle-cmd-rpm <= spindle.0.speed-out-abs =>
remora.pwm.00.value
# net spindle-on <= spindle.0.on => remora.pwm.00.enable

# Spindle Phase/Index for Threading (if encoder connected)
# net spindle-index-enable <=> spindle.0.index-enable
# net spindle-phase-a <= remora.encoder.00.phase-A
# net spindle-phase-b <= remora.encoder.00.phase-B
# net spindle-phase-z <= remora.encoder.00.phase-Z

# --- MPG Handwheel (Example using Remora Encoder 01) ---
# Connect encoder counts to HALUI counts input
net mpg-counts <= remora.encoder.01.count => halui.axis.jog-counts #
Connect to appropriate halui jog counts pin

# Enable MPG encoder only when in manual mode and correct axis
selected (Example for X)
# Requires additional logic components (e.g., and2)
# loadrt and2 count=1
# addf and2.0 servo-thread
# net mpg-enable-manual <= halui.mode.is-manual
# net mpg-enable-axis <= halui.axis.x.is-selected # Check specific
halui pin name
# net mpg-enable mpg-enable-manual mpg-enable-axis => and2.0.in0
and2.0.in1
# net mpg-enable and2.0.out => remora.encoder.01.enable

# --- Limit and Home Switches ---
# IMPORTANT: Connect physical switches to Remora inputs and configure
Remora firmware mapping.
# Then, connect the Remora input pins to the corresponding LinuxCNC
joint limit/home pins.
# Replace XX with the correct Remora input pin numbers configured for
your hardware.

# Example for Joint 0 (X)
# net x-home-sw <= remora.input.XX.in => joint.0.home-sw-in
# net x-neg-limit <= remora.input.XX.in => joint.0.neg-lim-sw-in
# net x-pos-limit <= remora.input.XX.in => joint.0.pos-lim-sw-in

# Example for Joint 1 (Z)
```

```
# net z-home-sw <= remora.input.XX.in => joint.1.home-sw-in
# net z-neg-limit <= remora.input.XX.in => joint.1.neg-lim-sw-in
# net z-pos-limit <= remora.input.XX.in => joint.1.pos-lim-sw-in

# --- Other I/O (Examples) ---
# Coolant Pins (Example using Remora Output XX)
# net flood-cmd <= iocontrol.0.coolant-flood => remora.output.XX.out
# net mist-cmd <= iocontrol.0.coolant-mist => remora.output.XX.out

# Removed incorrect 'loadusr -W stepgen' and related 'setp' lines.
```

Explanation of Major Corrections:

- **INI File:**
  - Standardized sections like [EMC], [TASK], [HAL], [TRAJ], [KINS].
  - Corrected COORDINATES, KINEMATICS, and JOINTS for a standard XZ lathe.
  - Added the required LATHE = 1 for the AXIS GUI.
  - Added standard [EMCMOT] section for timing.
  - Removed non-standard sections ([NUM_MOTORS], [AXES], etc.).
  - Added proper [AXIS_X], [AXIS_Z], [JOINT_0], [JOINT_1] sections with placeholders for crucial machine-specific parameters (SCALE, Limits, Homing).
  - Corrected potentially extreme velocity/acceleration values to more common starting points. **You must tune these.**
  - Added G18 (XZ plane selection) to the startup codes.
- **HAL File:**
  - Removed the incorrect loading and configuration of the userspace stepgen component. Remora handles the step generation based on commands sent to its HAL pins.
  - Corrected net commands to connect LinuxCNC joint.N.* pins to the appropriate remora.stepgen.##.* pins for position commands, feedback, and enables.
  - Added example addf commands to link Remora functions to the servo-thread (you need to verify the exact function names from Remora documentation).
  - Corrected the E-stop logic example and noted the potential conflict.
  - Added placeholder net commands for limit/home switches and spindle control, emphasizing the need for user configuration based on hardware and Remora setup.
  - Added a net motion-enable example connecting joint enables to the main Remora enable pin.

Extra Steps You Need to Take:

1. **Verify Machine Type:** Confirm if your machine is a standard XZ lathe or has a C-axis (mill-turn). The corrections assume a standard XZ lathe. If it's different, the INI (COORDINATES, KINEMATICS, JOINTS) and HAL (joint./axis. connections) will need significant changes.
2. **Remora Configuration:** This is critical. The HAL file assumes specific Remora pins exist (e.g., remora.stepgen.00, remora.stepgen.02, remora.encoder.00, remora.input.11, remora.enable, remora.status, remora.reset). You **must** configure your Remora firmware (likely via its own configuration file or interface) to map these functions to the correct physical pins on your NVEM board according to your wiring.

3. **HAL Function Names:** Find the correct function names provided by your remora-eth component (e.g., remora.read, remora.write, remora.update) and ensure they are added to the servo-thread using addf commands in custom.hal. The provided names are examples.
4. **Hardware Wiring:** Double-check that your motors, encoders, limit/home switches, E-stop, spindle interface, MPG, etc., are physically wired to the NVEM pins corresponding to your Remora configuration and the pins used in the corrected custom.hal file.
5. **INI Parameter Values:**
    - **SCALE:** Accurately calculate the SCALE value for [JOINT_0] and [JOINT_1] based on your motor steps/revolution, microstepping setting on the driver, gear/pulley ratios, and leadscrew pitch (mm per revolution).
    - **Limits:** Measure your machine's travel range and set MIN_LIMIT and MAX_LIMIT accurately in both the [AXIS_L] and [JOINT_n] sections. Joint limits should be slightly wider than axis limits.
    - **Homing:** Configure all HOME_... variables in the [JOINT_n] sections based on your switch locations, desired final home position, and homing direction/speed. Pay close attention to HOME_OFFSET.
    - **Velocity/Acceleration:** Start with the provided conservative values for MAX_VELOCITY and MAX_ACCELERATION in [TRAJ], [AXIS_L], and [JOINT_n] sections and carefully tune them upwards by testing later.
    - **FERROR/MIN_FERROR:** These define allowable following error. Start with values like 1 and 0.25 respectively for mm units.
6. **HAL Limit/Home/Spindle Connections:** Edit the commented-out sections in custom.hal to connect the actual remora.input.XX.in pins (for switches) and remora.encoder/pwm.XX pins (for spindle) based on your Remora configuration and wiring.
7. **Latency Test:** Run the latency test (latency-test in terminal) for an extended period (15+ minutes) while stressing the PC (moving windows, copying files). Ensure the SERVO_PERIOD in [EMCMOT] is significantly larger than the maximum jitter reported. 1ms (1000000ns) is usually safe with hardware step generation like Remora.
8. **Testing and Tuning:**
    - Start LinuxCNC. Address any immediate HAL or INI errors reported.
    - Use HAL tools (halshow, halmeter) to verify signals (E-stop, enables, limits) before attempting to enable the machine.
    - Test homing for each axis individually. Verify direction, speed, and final position.
    - Carefully test jogging and small G-code moves, gradually increasing speed and acceleration limits while listening for motor stalls or lost steps. Tune MAX_VELOCITY and MAX_ACCELERATION.
9. **Remap Files:** If you kept the REMAP lines in the INI file, ensure the corresponding .ngc files (m53.ngc, m54.ngc) exist in the directory specified by SUBROUTINE_PATH (macros in this case).

Take these steps carefully, especially verifying the Remora configuration and physical wiring against the corrected HAL file. Good luck!