# **Boot Optimization and Kernel Configuration Guide for AM43xx EVM**

## U-Boot Configuration

### Reducing or Increasing Boot Delay

To change the U-Boot autoboot delay time, modify the following:  
  
File: **configs/am43xx\_evm\_defconfig**

Add or update the line:

**CONFIG\_BOOTDELAY=0 // Delay in seconds**  
  
Example: Setting CONFIG\_BOOTDELAY=100 will introduce a 100-second delay.  
  
To confirm the applied configuration:  
After building U-Boot, check the generated .config file:

**CONFIG\_AUTOBOOT=y  
CONFIG\_BOOTDELAY=0**

### Disabling Kernel Boot Logs (Reducing Verbosity)

To suppress logs during the Linux kernel boot, modify the U-Boot environment:  
1. Edit the file: **include/configs/am43xx\_evm.h**

2. Change the nandargs:

=> **printenv nandargs**  
**nandargs=setenv bootargs console=${console} ${optargs} root=${nandroot} rootfstype=${nandrootfstype} quiet loglevel=3**

3. This sets a quieter kernel output (loglevel 3).

## Linux Kernel Configuration

### Locate the Defconfig File

**$ find -iname tisdk\_am437x-evm-rt\_defconfig**  
**Typical location:**  
**./arch/arm/configs/tisdk\_am437x-evm-rt\_defconfig  
  
Edit the file using:**  
**$ vim ./arch/arm/configs/tisdk\_am437x-evm-rt\_defconfig**

### Relevant Kernel Config Options

#### CAN (Controller Area Network) Support:

CONFIG\_CAN\_DEV=y // CAN Device Driver Interface  
CONFIG\_CAN\_RAW=y // Raw CAN sockets  
CONFIG\_CAN=y // Enable CAN subsystem  
CONFIG\_CAN\_BCM=y // CAN Broadcast Manager  
CONFIG\_CAN\_GW=y // CAN Gateway support  
CONFIG\_CAN\_C\_CAN=y // Bosch C\_CAN core  
CONFIG\_CAN\_C\_CAN\_PLATFORM=y // Platform support for C\_CAN

#### Ethernet and PRUSS Support:

CONFIG\_HSR=y // High-availability Seamless Redundancy  
CONFIG\_TI\_PRUSS=y // TI PRUSS support  
CONFIG\_PRU\_REMOTEPROC=y // PRU remoteproc driver  
CONFIG\_TI\_ICSS\_IEP=y // Industrial Ethernet Programmable Unit  
CONFIG\_TI\_PRUETH=y // PRU Ethernet driver  
CONFIG\_TI\_PRUSS\_INTC=y // PRUSS Interrupt Controller

#### Networking Support:

CONFIG\_LLC=y // IEEE 802.2 Logical Link Control  
CONFIG\_STP=y // Spanning Tree Protocol  
CONFIG\_BRIDGE=y // Ethernet bridging support

#### Additional Hardware Support:

CONFIG\_EEPROM\_AT25=y // Support for AT25 series EEPROMs

### Modify BitBake .bb File and Rebuild Custom Tiny Image

**Edit the following file:**

**$ ./meta-arago-distro/recipes-core/images/tisdk-tiny-image.bb**

### Content:

**# Enable SysVinit for this image**

**ARAGO\_SYSVINIT = "1"**

**# Include common image settings**

**require arago-image.inc**

**# Remove unnecessary features**

**IMAGE\_FEATURES\_remove = "package-management splash"**

**# Output formats**

**IMAGE\_FSTYPES += "cpio cpio.xz"**

**# Optional installs**

**ARAGO\_TINY\_IMAGE\_EXTRA\_INSTALL ?= ""**

**# Install these packages in the image**

**IMAGE\_INSTALL = " \**

**packagegroup-arago-sysvinit-boot \**

**net-snmp \**

**sntp \**

**iproute2 \**

**can-utils \**

**openssh-scp \**

**openssh-ssh \**

**bridge-utils \**

**mstpd \**

**mtd-utils \**

**${ARAGO\_TINY\_IMAGE\_EXTRA\_INSTALL} \**

**"# Optional: Disable do\_packagedata to save build time**

**deltask do\_packagedata**

**# Set image basename**

**export IMAGE\_BASENAME = "tisdk-tiny-image"**

**Navigate to the build directory**

**$ cd ~/yocto/build**

**Run the build command**:

MACHINE=am437x-evm bitbake tisdk-tiny-image

**Extract and Deploy Root Filesystem**

**$ <build\_dir>/deploy/images/am437x-evm/tisdk-tiny-image-am437x-evm-<timestamp>.rootfs.tar.xz**

**Create or clean the root filesystem directory**

**$ sudo mkdir -p /rootfs**

**Extract the root filesystem:**

**$ sudo tar -xJf tisdk-tiny-image-am437x-evm-20250515075447.rootfs.tar.xz -C /rootfs**

**Navigate into the extracted rootfs directory:**

**$ cd rootfs**

## Rootfs Configuration:

### Auto Login Configuration

**Edit etc/inittab:**

**$ vi etc/inittab**

**Update or add the following lines:**

l1:1:wait:/etc/init.d/rc 1

l2:2:wait:/etc/init.d/rc 2

l3:3:wait:/etc/init.d/rc 3

l4:4:wait:/etc/init.d/rc 4

**#l5:5:wait:/etc/init.d/rc 5**  **# Commented**

**st:5:once:/etc/init.d/start\_services.sh # Added**

l6:6:wait:/etc/init.d/rc 6

z6:6:respawn:/sbin/sulogin

**S:2345:respawn:/sbin/getty 115200 ttyS0 -n -l /bin/autologin # Added**

**#S0:12345:respawn:/bin/start\_getty 115200 ttyS0 vt102 # Commented**

### Create autologin Script

**Create file:**

**$ vi bin/autologin**

**Insert the following content:**

**#!/bin/sh**

**exec /bin/login -f root**

**Make it executable:**

**$ chmod +x bin/autologin**

**Custom Init Script for Service Startup**

**Create the script:**

**$ sudo etc/init.d/start\_services.sh**

**Make it executable**:

**sudo chmod +x etc/init.d/start\_services.sh**

**Content:**

**echo "[DEBUG] Launching /etc/network/network.sh"**

**/etc/network/network.sh &**

**sleep 1**

**echo "[DEBUG] Setting executable permission for /home/root/sw\_update/sw\_update.sh"**

**chmod 777 /home/root/sw\_update/sw\_update.sh**

**echo "[DEBUG] Launching /home/root/sw\_update/sw\_update.sh"**

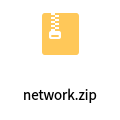
**/home/root/sw\_update/sw\_update.sh &**

**Network Initialization Script**

**Create and edit network.sh:**

**$ vi etc/network/network.sh**

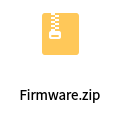
**Content:**

****

**Update TIPRUSS Firmware in Rootfs:**

**$ cd lib/firmware/tipruss**

**Copy the TISDK base image firmware files:**



**$ cp /path/to/Downloads/Firmware \* .**

**Verify that the correct firmware files are now present in:**

**$ <rootfs\_path>lib/firmware/tipruss**

**Software Update Script**

**Create sw\_update.sh:**

**$ vi home/root/sw\_update/sw\_update.sh**

**content:**

**#!/bin/sh**

**LOGFILE="/home/root/sw\_update/app\_run\_log.txt"**

**chmod 777 /home/root/sw\_update/Main\_OS\_app**

**chmod 777 /home/root/sw\_update/firmware-update**

**CMDLINE=$(cat /proc/cmdline)**

**echo "====== Boot Check: $(date) ======" >> "$LOGFILE"**

**echo "Kernel Cmdline: $CMDLINE" >> "$LOGFILE"**

**if echo "$CMDLINE" | grep -q "NAND.file-system-1"; then**

**echo "Detected NAND.file-system-1. Launching Main\_OS\_app..." >> "$LOGFILE"**

**/home/root/sw\_update/Main\_OS\_app >> "$LOGFILE" 2>&1 &**

**elif echo "$CMDLINE" | grep -q "NAND.file-system-2"; then**

**echo "Detected NAND.file-system-2. Launching firmware-update..." >> "$LOGFILE"**

**/home/root/sw\_update/firmware-update >> "$LOGFILE" 2>&1 &**

**else**

**echo "ERROR: Unknown rootfs partition." >> "$LOGFILE"**

**fi**

**Edit etc/init.d/bootmisc.sh**

**$ vi etc/init.d/bootmisc.sh**

**# This is as good a place as any for a sanity check**

**#**

**# Set the system clock from hardware clock**

**# If the timestamp is more recent than the current time,**

**# use the timestamp instead.**

**#test -x /etc/init.d/hwclock.sh && /etc/init.d/hwclock.sh start #Commented**

**if test -e "$TIMESTAMP\_FILE"**

**then**

**SYSTEMDATE=`date -u +%4Y%2m%2d%2H%2M%2S`**

**read TIMESTAMP < "$TIMESTAMP\_FILE"**

**if [ ${TIMESTAMP} -gt $SYSTEMDATE ]; then**

**# format the timestamp as date expects it (2m2d2H2M4Y.2S)**

**TS\_YR=${TIMESTAMP%??????????}**

**TS\_SEC=${TIMESTAMP#????????????}**

**TS\_FIRST12=${TIMESTAMP%??}**

**TS\_MIDDLE8=${TS\_FIRST12#????}**

**date -u ${TS\_MIDDLE8}${TS\_YR}.${TS\_SEC}**

**#test -x /etc/init.d/hwclock.sh && /etc/init.d/hwclock.sh stop #Commented**

**fi**

**fi**

**To create a UBI-compatible root filesystem:**

**ubinize.cfg Configuration File**

**$ vim ubinize.cfg**

**Content:**

**[ubifs]**

**mode=ubi**

**image=rootfs.ubifs**

**vol\_id=0**

**vol\_size=28MiB**

**vol\_type=dynamic**

**vol\_name=rootfs**

**vol\_alignment=1**

**vol\_flags=autoresize**

**Create UBIFS Image**

**$ sudo mkfs.ubifs -r <rootfs path>/rootfs -o rootfs.ubifs -F -m 2048 -e 126976 -c 3989**

**Create UBI Image from UBIFS**

**$ sudo ubinize -o rootfs.ubi -m 2048 -p 128KiB -s 512 -O 2048 ubinize.cfg**

**Copy UBI Image to SD Card**

**$ cp rootfs.ubi /path/to/sdcard/**

**Flash UBI Image to NAND.....**