Application Note

Building Low Power Human-Machine Interface with AM62L

Anshu Madwesh and Kendall Willis

ABSTRACT

For Human-Machine Interface (HMI) applications on TI's AM62Lx processor, power consumption can be reduced through active optimizations (frequency scaling, DDR efficiency improvements) and idle-state optimizations. This document covers both methodologies for Linux systems, highlighting the Always-On Display low power state as a key idle optimization technique.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Introduction to Human Machine Interface

Human-Machine Interface (HMI) applications are found in a wide variety of industrial, medical, consumer electronics among other products for a user to monitor and interact with machines. HMIs improve the ease of use of the machines, typically by utilizing a display interface for a user to interact with. As energy efficiency standards continue to be enacted by governing bodies, there is a growing emphasis on designing products that meet these requirements. Specifically, there is an emphasis on reducing the standby power of these products to minimize the idle power consumption, hence reduce energy demand. The embedded processors in these applications are responsible for entering and exiting the system from low power idle states, in addition to running the system while minimizing the total power consumption.

2 Introduction to AM62L

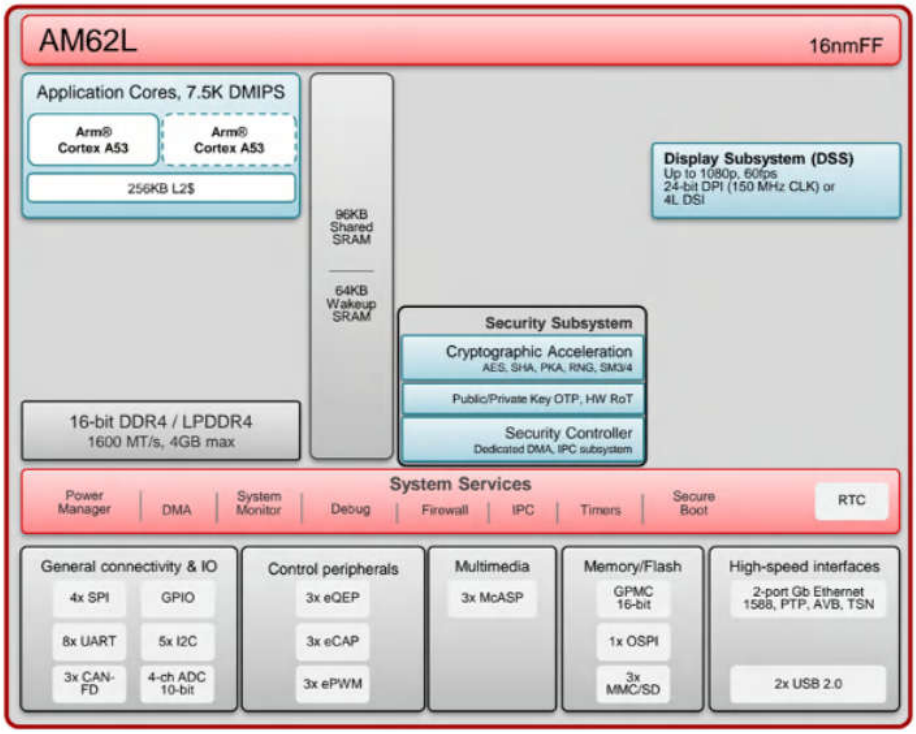

Figure 2-1. AM62L Block Diagram

AM62L is a ARM-based processor, which is a low-cost, power-efficient system-on-chip (SoC) that is designed for applications including HMI. The main computation block is the 2-Core A53 ARM cortex which can run up to 1.25GHz. AM62L offers both a 1x A53 and 2x A53 core design. The processor contains a memory controller for 16-Bit DDR interface that operates at 1600 MT/s data rate which is used to store information and allows the HLOS (High Level Operating System) to operate. The processor uses DSS (Display Subsystem) to pipeline graphics onto a display, which supports both MIPI-DSI (Display Serial Interface) and DPI (Display Parallel Interface). AM62L also offers various peripherals to interact with other ICs in the full system.

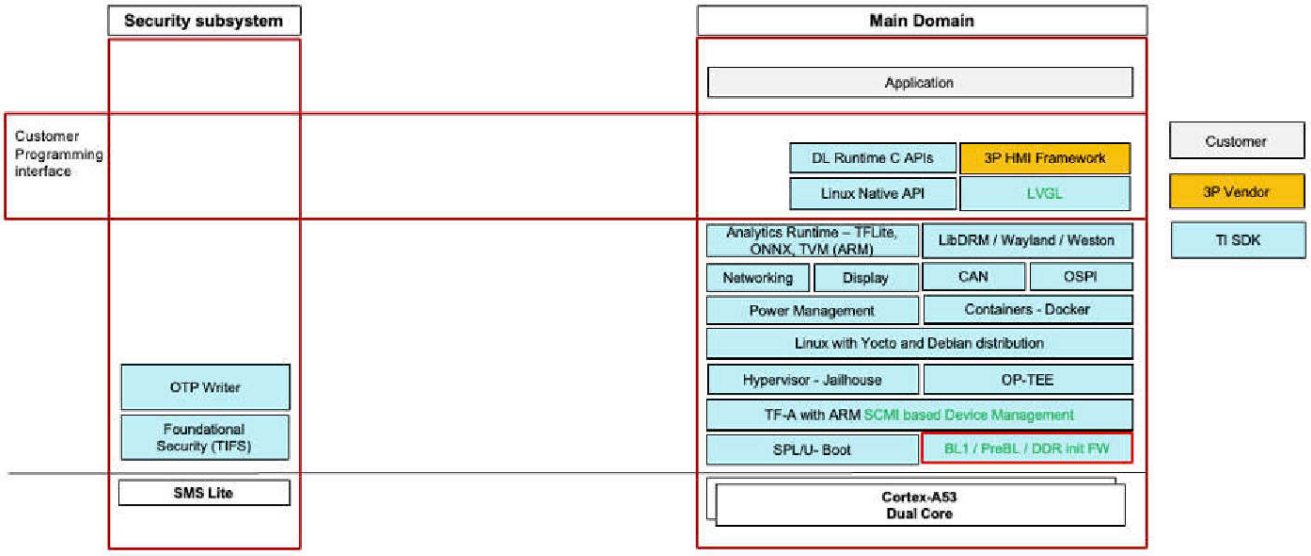

Figure 2-2. AM62L Software Stack

Figure 2-2 shows the software stack of the AM62L SoC where the developer can use the AM62L SDK to build an application.

3 Demo Overview

3.1 Background

AM62L is a low power Arm-Based embedded processor with display capabilities to drive human machine interface (HMI) applications. This demo shows AM62L running Linux in a low power active state while driving a DSI connected display showing a graphics application. This application note discusses the key considerations for making a power optimized HMI application and demonstrates the execution.

4 Methods to Reduce Active Power

4.1 Light Weight Graphics Application

The application cores perform most of the computational heavy lifting of the system. For HMI systems, the application cores will be responsible for driving the graphics application. When designing a graphics application, it is recommended to use a lightweight framework that is made for embedded systems that focuses on low CPU and memory usage. Providing minimal features in the graphics application allows power savings since the processor does not have to generate large graphical outputs.

4.2 Minimal Linux Device Tree

The Linux device tree provides a method to describe the hardware by using a graph of nodes to describe aspects of the hardware. It is used by the Linux kernel for platform identification, runtime configuration, and device population. When the peripheral hardware is described in the device tree, Linux enables and power the peripheral, even if it is idle. To save power, unused device tree nodes should be disabled. When a node is disabled Linux ignores the device tree parameters and the peripheral will be left in a reset state. It is important to note that if a node does not have the *status* property, Linux enables the device node. To disable a node, the status property should be changed to *disabled*.

Additionally, the device tree nodes may need to configure pins' multiplexing (PinMux) for the peripheral's signals to work. However, since these disabled peripherals are not in use, defining their respective PinMux is not required. The pad keeps the receiver and transceiver disabled along with keeping the MuxMode in a GPIO state. These three components reduces the amount of leakage current of the I/Os.

4.3 Local Power Sleep Controller (LPSC)

Within the AM62L processor is a Power Sleep Controller (PSC) architecture which is responsible for managing the power and clocking for each peripheral, controller and core. There are multiple PSCs for AM62L, each with its own power domains. To provide more granularity on the control of the components within a PSC, there are child nodes of the power domain called local power sleep controllers (LPSCs) which allows for software control of individual modules.

LPSCs are software controllable except for the 'ALWAYS-ON' domain which cannot be turned off. The Linux device tree will initialize the required clocks for each peripheral. Since the clock will be active, the LPSC will be on, hence the peripheral will be on and consuming power. This also works in reverse where if a device tree node is disabled, then the clock will not be initialized so the LPSC will remain off.

In cases where an LPSC is not described in the device tree, the LPSC state can be changed so that the LPSC is clock gated and the local reset is de-asserted.

4.4 Reducing Clock Frequencies

The AM62L's clocking architecture groups peripherals and core in different phase lock loops (PLLs). When a PLL is disabled, all the child peripherals are disabled. If the PLL cannot be turned off, the next option is to reduce or disable the clock for unused peripherals while optimizing the clock speeds of required peripherals.

The clock speed of the processor has a linear relationship to its dynamic power consumption. In optimizing power consumption of embedded systems, find opportunities to slow down the clock frequency to reduce power consumption.

4.5 Auto Clock Gating

Clock Gating is a power management technique that can be utilized to reduce dynamic (active/switching) power consumption of a SoC. When the module or circuit within the SoC is idle, the clock signal can be removed from the clock tree.

AM62L offers Auto Clock Gating on different movement portions of the SoC including the Common Bus Architecture (CBASS), Data Movement Architecture (DMA), & SRAM. AM62L also offers individual clock gating options for peripherals like Cryptographic Module, USB, MMCSD, Display Subsystem (DSS), and so on.

When evaluating Auto Clock Gating for the system design, consider the activity levels of components. Components with high activity do not see many opportunities to use the power savings of auto clock gating.

4.6 DDR in Auto Self-Refresh

DDR is the external memory IC that the AM62L processor will use to store and fetch data which is critical systems running HLOS. When there are no expected transactions between DDR and the SoC, DDR can be put into self-refresh after a defined number of clock cycles.

DDR self-refresh is the process where the DRAM is read periodically and re-written to in order to preserve the data. DDR can be automatically put into self-refresh by defining in the controller which low power state to enter after a number of inactive clock cycles. The DDR auto self-refresh feature should be used as frequently as possible when the system is idle to save more power.

4.7 Performance

In designing for embedded systems, a key trade-off is the processor's power consumption vs performance. Optimizing around the system application allows for maximum power savings without impacting the performance by using the optimization methods discussed in previous sections.

5 Demo Specific Optimizations

5.1 LVGL Application

This demo uses Light and Versatile Graphics Library (LVGL) to build the graphics application. LVGL provides functions for building graphics applications on embedded systems while focusing on low CPU utilization and memory usage. The graphics application for this demo features a black screen that shows the current time and the TI logo.

See Appendix 11.1 for an image of the GUI.

5.2 Minimal Device Tree

The minimal device tree removes all nodes that are not necessary for the demo's functionality and removes the corresponding pins that had been configured in the PinMux. SeeAppendix 11.2 for the patch of the minimal device tree.

5.3 Slow Clock Speed

K3Conf is a debug tool that allows developers to disable devices, disable clocks, and change clock frequency. k3conf is packaged as apart of the TI Processor SDK. Use k3conf --help to see all commands that can be used. An easy way to check device status is by using the command k3conf dump clocks.

```
root@am62lxx-evm:~# k3conf dump clock
 ************************************************************************
 ************************************************************************
 * *
 * EXPERIMENTAL SUPPORT! *
 * *
 ************************************************************************
 ************************************************************************

|-------------------------------------------------------------------------------------------------------
---------|
| VERSION INFO |

|-------------------------------------------------------------------------------------------------------
---------|
| K3CONF | (version 0.4-nogit built Mon May 11 05:21:53 UTC 2026) |
| SoC | AM62Lx SR2.0 |
| SoC identifiers | [0x7234f3e6] 0x391a7 Func-Safe Secure 'O' Grade -40 C to 105 C ANB Package |
| SCMI Implementation Version | 0x0 |

|-------------------------------------------------------------------------------------------------------
---------|

|-------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------|
| Device ID | Clock ID | Clock Name | Status | Clock Frequency |

|-------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------|
| AM62LX_DEV_ADC0 | 0 | AM62LX_DEV_ADC0_ADC_CLK_PARENT_GLUELOGIC_HFOSC0_CLK | CLK_STATE_NOT_READY |
25000000 |
| AM62LX_DEV_ADC0 | 1 |
AM62LX_DEV_ADC0_ADC_CLK_PARENT_SAM62_PLL_CTRL_WRAP_WKUP_0_CHIP_DIV1_CLK_CLK12 | CLK_STATE_NOT_READY
| 33333333 |
| AM62LX_DEV_ADC0 | 2 | AM62LX_DEV_ADC0_ADC_CLK_PARENT_POSTDIV4_16FF_WKUP_0_HSDIVOUT8_CLK |
CLK_STATE_NOT_READY | 30000000 |
```

Change the frequency of a clock by using the command k3conf set clock 0 <clk_id> <freq> . For example,

```
root@am62lxx-evm:~# k3conf set clock 0 521 9375000
 ************************************************************************
 ************************************************************************
 * *
 * EXPERIMENTAL SUPPORT! *
 * *
************************************************************************
 ************************************************************************

|-------------------------------------------------------------------------------------------------------
---------|
```

```
| VERSION INFO |

|-------------------------------------------------------------------------------------------------------------
---------|
| K3CONF | (version 0.4-nogit built Mon May 11 05:21:53 UTC 2026) |
| SoC | AM62Lx SR2.0 |
| SoC identifiers | [0x7234f3e6] 0x391a7 Func-Safe Secure 'O' Grade -40 C to 105 C ANB Package |
| SCMI Implementation Version | 0x0 |

|-------------------------------------------------------------------------------------------------------------
---------|

|-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------|
| Device ID | Clock ID | Clock Name | Status | Clock Frequency |

|-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------|
| AM62LX_DEV_WKUP_TIMER1_CLKSEL_VD | 521 | AM62LX_DEV_WKUP_TIMER1_CLKSEL_VD_CLK | CLK_STATE_READY |
9375000 |
```

The peripherals in the WKUP domain are not used, therefore the clock that connects them should be slowed down for maximum power savings. k3conf can be used to slow the clocks in the WKUP domain. For the full list of what is changed, consult Appendix 11.3.

5.4 Slow A53 Core Clock Frequency

AM62L has two A53 CPU cores configured to run at 1.25GHz during normal operation. To reduce power while keeping the system stable, reduce the speed of the cores to 200MHz. To change the clock speed of the A53 cores, the PLL divider being used for the clock has to be changed. The CPUFreq framework in Linux can be used to change the A53 clocks easily from userspace. CPUFreq changes the frequency of the CPUs based off of the governor it uses. The userspace governor allows the user to change the clock frequency from userspace. The available clock frequencies can also be checked in userspace.

```
root@am62lxx-evm:~# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
ondemand userspace performance
root@am62lxx-evm:~# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
200000 400000 600000 800000 1000000 1250000
root@am62lxx-evm:~# echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
root@am62lxx-evm:~# echo 200000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
root@am62lxx-evm:~# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
200000
```

5.5 Turn Off One A53 Core

Only one A53 core is needed for the system to run, so turn off one of them. CPU hotplug in Linux is used to turn off one of the cores. The following command is needed to turn off the core in the Linux console.

```
root@am62lxx-evm:~# echo 0 > /sys/devices/system/cpu/cpu1/online
```

5.6 Slow DDR Transfer Speed

The DDR transfer speed can be slowed by half from 1600 MT/s to 800 MT/s. This consumes less power since fewer transfers are happening over the same amount of time. The DDR transfer speed is determined by the DDR clock speed. The memory speed of DDR is double the speed of the PLL and the data rate is double the speed of the memory. The DDR clock speed is 400 MHz. If the DDR clock speed is 400 MHz, the memory speed is 800 MHz and the data rate is 1600 MT/s. In order to halve the data rate, half the DDR clock speed to 200 MHz. Refer to Appendix 11.4 for the patch that slows DDR in Arm Trusted Firmware.

5.7 DDR Auto Self-Refresh

The AM62L EVM uses LPDDR4 which allows for auto self-refresh. The DDR controller supports three low power states. Enable the low power states along with automatic entry and exit to them. Define the amount of idle time to enter each of the state as long as possible. To set the auto self refresh configurations for DDR, see the Arm Trusted Firmware patch in Appendix 11.5.

5.8 Auto Clock Gating

AM62L offers several auto clock gating for several controllers. To enable auto clock gating for all controllers, write 0 to the WKUP_CTRL_MMR_CFG5_CLKGATE_CTRL0 register. Refer to Appendix 11.6 on how to enable auto clock gating in Arm Trusted Firmware.

5.9 Slow Panel FPS

FPS (frames per second) for a display is the amount of frames that are displayed per second. Slowing the FPS allows for power savings by decreasing the amount of times the display needs to be updated. In some cases the tool modetest can be used to decrease FPS for a display.

The RPI display is a 800 × 480 resolution display running at 60 FPS in Raster mode. For the RPI display the pixel clock needs to be slowed in the panel-simple Linux driver to slow FPS. The panel-simple Linux driver has a data structure for the RPI panel, the clock variable has the speed at which the pixel clock is running at. By default the RPI panel 60 FPS, the panel pixel clock can be halved achieve 30 FPS. See Appendix 11.7 for the patch that slows the panel pixel clock frequency in Linux.

Alternatively, if the display panel in use has built-in RAM, then DSS can be put into Command mode and auto self refresh for additional power savings.

6 Power Data

The following power data is specific to AM62L using SDK 11.00.15. Creating this same demo using other specifications will yield different results. The results shown are to demonstrate the potential difference between the power optimization steps described and the standard OS idle.

The following power measurements were taken using the described power optimizations applied and the RPI touch display showing the LVGL demo. The measurements are an average of measurements taken once per second for fifteen minutes.

Rail Name	Rail Voltage (V)	Power (mW)
vdd_core	0.75	171.07
soc_dvdd_1v8	1.80	22.54
soc_dvdd_3v3	3.30	2.69
vdda_1v8	1.80	29.29
vdd_lpddr4_pmic1	1.10	31.17
vdd_rtc	0.75	0.04
vdd_rtc_1v8	1.80	0.02
Total		256.82

For comparison, the following power measurements is of OS idle with a display connected, but without any power optimizations. The measurements are an average of measurements taken once per second for fifteen minutes.

Rail Name	Rail Voltage (V)	Power (mW)
vdd_core	0.75	321.26
soc_dvdd_1v8	1.80	27.46
soc_dvdd_3v3	3.30	3.48
vdda_1v8	1.80	36.46
vdd_lpddr4_pmic1	1.10	61.85
vdd_rtc	0.75	0.04
vdd_rtc_1v8	1.80	0.01
Total		450.56

The total difference between the regular OS idle and power optimized OS idle is about 200mW of power savings.

7 Implementing Low Powers Modes within an HMI Use Case

The next level for power optimization is including low power modes into the system level design. Low power modes offer opportunities to reduce power consumption when the system can be idle. AM62L offers a unique low power state called Always-On Display mode (also known as DSS + DeepSleep) which enables the display subsystem and its supporting frame buffer, stored in the on-chip RAM, to remain active while the rest of the SoC enters the DeepSleep low power state. This low power mode designed to keep a static image in the frame buffer so the display subsystem can fetch from the frame buffer and keep the static image on the panel active while minimizing the power consumption. Then after 59 seconds, the SoC will exit Always-On Display mode so the application code can update the frame buffer and then re-enter Always-On Display mode. This low power mode is useful for HMI application when the system is idle but the display needs to remain on.

Low Power Mode	Wake up source	Power (mW)	Wake time	RTC	WKUP/ Always ON	Main (A53)	Main PLL	DSS + Framebuffer	DDR
RTC-Only	EXT_WKUP Pin, RTC	<1	Cold Boot	ON	OFF	OFF	OFF	OFF	OFF
RTC-Only + DDR	EXT_WKUP Pin, RTC, I/O pins	<2	700 mS	ON	OFF	OFF	OFF	OFF	Self-Refresh
DeepSleep	RTC, USB, I2C, UART, GPIO	<10	200 mS	ON	ON	OFF	OFF	OFF	Self-Refresh
Always-On Display	DeepSleep wakeups, DSP events	~70	150 mS	ON	ON	OFF	OFF	ON	Self-Refresh

Figure 7-1. AM62L Low Power Modes

Table 7-1 lists the power consumption of Always-On Display mode:

Table 7-1. Always-On Display Power Consumption

Rail Name	Rail Voltage (V)	Power (mW)
vdd_core	0.75	44.66
soc_dvdd_1v8	1.80	1.77
soc_dvdd_3v3	3.30	3.64
vdda_1v8	1.80	18.82
vdd_lpddr4_pmic1	1.10	1.06
vdd_rtc	0.75	0.032
vdd_rtc_1v8	1.80	0.015
Total		69.99

Table 7-2 is a direct comparison of all the optimizations done:

Table 7-2. Comparing Always-On Display with OS Idle Power Consumption

Rail Name	Rail Voltage (V)	No Optimizations (mW)	Optimized OS Idle with Display Active (mW)	Always-On Display Mode (mW)
vdd_core	0.75	321.26	171.07	44.66
soc_dvdd_1v8	1.80	27.46	22.54	1.77
soc_dvdd_3v3	3.30	3.48	2.69	3.64
vdda_1v8	1.80	36.46	29.29	18.82
vdd_lpddr4_pmic1	1.10	61.85	31.17	1.06
vdd_rtc	0.75	0.04	0.04	0.032
vdd_rtc_1v8	1.80	0.01	0.02	0.015
Total		450.56	256.82	69.99

This low power mode will be added in SDK 12.1. For more information, please contact your TI representative or post a question on e2e.ti.com.

8 Demo Steps

8.1 System Requirements

- Display
  - 7-inch Raspberry Pi Touch Display panel
- AM62L
- 32 GB SD card
- Demo image
  - TI SDK 12.00.00.07 image
  - Raspberry PI DSI overlay
  - Script to reduce OS idle power
  - LVGL demo image

8.2 Prepare the SD Card

Flash the 32 GB SD card with the SDK 12.00.00.07 default image: https://www.ti.com/tool/download/AM62L-LINUX-SDK/12.00.00.07.04

Eject the SD card and insert it again. The SD card should mount with two directories. One that is /media/$USER/root/ and /media/$USER/boot/

Prebuilt files and instructions on how to install them on the SD card are found here: https://github.com/kwillis01/Beyond-SDK/tree/am62l-hmi-demo/am62l/examples/hmi-demo

8.2.1 Build the Required Files

Building all of the files takes place from the host machine.

The following GitHub repository provides all necessary build files

Use these instructions to install the necessary toolchains to build the various images:

For ease of use, create the following environment variables for the paths to each directory when they are cloned.

```
HOST$ export UBOOT_DIR=<path-to-ti-u-boot>
HOST$ export TI_LINUX_FW_DIR=<path-to-ti-linux-firmware>
HOST$ export TFA_DIR=<path-to-arm-trusted-firmware>
HOST$ export LINUX_DIR=<path-to-linux>
HOST$ export TI_LVGL_DEMO_DIR=<path-to-ti-lvgl-demo>
```

8.2.1.1 Boot Images

U-Boot, OP-TEE, Arm Trusted Firmware (TF-A), and ti-linux-firmware are needed to build the boot files.

Instructions to clone and build TF-A:

Apply the patches under Appendices 11.4, 11.5, and 11.6 to TF-A, then build TF-A.

Instructions to clone and build OP-TEE:

Instructions to clone and build U-Boot:

The boot image files are tiboot3.bin, tispl.bin, and u-boot.img. On the host computer, move the tiboot3.bin, tispl.bin, and u-boot.img files from U-Boot to the SD card boot partition.

```
HOST$ cp $UBOOT_DIR/tiboot3.bin /media/$USER/boot
HOST$ cp $UBOOT_DIR/tispl.bin /media/$USER/boot
HOST$ cp $UBOOT_DIR/u-boot.img /media/$USER/boot
```

8.2.1.2 U-Boot Commands

The uEnv.txt file allows setup U-Boot commands without having to set them every time.

This file will be used to remove the automatic console that would show up when a display is connected and to set the overlay for the RPI DSI touch display.

The contents of the uEnv.txt file is in Appendix 11.8. Move uEnv.txt to the SD card boot partition.

```
HOST$ cp uEnv.txt /media/$USER/boot
```

8.2.1.3 GUI Files

The lvglsim file is the binary for the LVGL graphics application.

Instructions to clone and build ti-lvgl-demo: https://github.com/kwillis01/ti-lvgl-demo/blob/hmi-simple-demo/README.md

First, clone the ti-lvgl-demo repository and switch to the hmi-simple-demo branch

```
HOST$ git clone --recurse-submodules https://github.com/kwillis01/ti-lvgl-demo.git
HOST$ cd ti-lvgl-demo
HOST$ git checkout hmi-simple-demo
HOST$ git submodule update --init --recursive
```

Setup docker by following this tutorial. Build the LVGL demo.

```
HOST$ ./scripts/docker_setup.sh --create-image
HOST$ ./scripts/docker_setup.sh --build-app
```

Move the lvglsim binary from ti-lvgl-demo/lv_port_linux/bin/ to the root directory in the root partition of the SD card.

```
HOST$ sudo cp $TI_LVGL_DEMO_DIR/lv_port_linux/bin/lvglsim /media/$USER/root/root
```

Download img_lv_demo_simple_high_res_ti_logo.png from here.

Make the directory ti-lvgl-demo/assets/ in folder /media/$USER/root/usr/share

Move img_lv_demo_simple_high_res_ti_logo.png to /media/$USER/root/usr/share/ti-lvgl-demo/assets/

```
HOST$ sudo mkdir /media/$USER/root/usr/share/ti-lvgl-demo
HOST$ sudo mkdir /media/$USER/root/usr/share/ti-lvgl-demo/assets
HOST$sudo cp $TI_LVGL_DEMO_DIR/lv_port_linux/lvgl/demos/simple_high_res/
img_lv_demo_simple_high_res_ti_logo.png /media/$USER/root/usr/share/ti-lvgl-demo/assets
```

8.2.1.4 Kernel Image, Modules, and Device Tree

Instructions to clone and build Linux:

Apply the patches under Appendices 11.2 and 11.7 to the Linux kernel, then build the kernel Image and device tree files.

Move k3-am62l3-evm-hmi.dtb to /media/$USER/root/boot/dtb/ti/k3-am62l3-evm.dtb and move k3-am62l3-evm-dsi-rpi-7inch-panel.dtbo to /media/$USER/root/boot/dtb/ti/

```
HOST$ sudo cp $LINUX_DIR/arch/arm64/boot/dts/ti/k3-am62l3-evm-hmi.dtb /media/$USER/root/boot/dtb/ti/
k3-am62l3-evm.dtb
HOST$ sudo cp $LINUX_DIR/arch/arm64/boot/dts/ti/k3-am62l3-evm-dsi-rpi-7inch-panel.dtbo /media/$USER/
root/boot/dtb/ti/
```

Install the kernel Image and modules on the SD root partition in the boot directory.

```
HOST$ sudo cp $LINUX_DIR/arch/arm64/boot/Image /media/$USER/root/boot
HOST$ sudo rm -rf /media/$USER/root/lib/modules/*; sudo make ARCH=arm64 INSTALL_MOD_PATH=/media/
$USER/root modules_install
```

8.2.1.5 Scripts

Create a file called wkup_periph_clk_freqs.sh with the contents found in Appendix 11.3 and another file called hmi-script.sh with the contents found in Appendix 11.9.

These script has all of the previous power optimizations added. It also starts the lvglsim binary. The scripts are optional to create, but all the commands listed in it must be done to run the demo.

Set the scripts to be executable:

```
HOST$ chmod +x wkup_periph_clk_freqs.sh
HOST$ chmod +x hmi-script.sh
```

Move hmi-script.sh to /media/$USER/root/root/

```
HOST$ sudo cp wkup_periph_clk_freqs.sh /media/$USER/root/root
HOST$ sudo cp hmi-script.sh /media/$USER/root/root
```

8.3 Run the Demo

1. After building all the files and installing them onto the SD card, eject it from the host computer.
2. Insert the SD card into AM62L.
3. Plug in the ribbon cable into the RPI Touch Display. The port will be on the back of the display. The connector pins should be facing away from the display.
4. On the backside of the AM62L, connect the ribbon cable to the DSI_CONN port. The connector pins should be facing the white part of the port.
5. From the host computer, plug in two micro USB cords to the AM62L UART and JTAG ports, respectively.
6. Open the AM62L Power Visualizer tool and follow the page for "How to Setup the Tool": https://dev.ti.com/gallery/view/SitaraMPU/AM62L%5FPower%5FVisualizer/ver/1.0.0/
7. On the host computer open /dev/ttyUSB0 for the Linux UART console.
8. Plug in power to the USB C port on AM62L.
9. When Linux starts to boot up, power the RPI display.

Note

Do not do this too late into boot or the display is not be detected.

10. Start streaming the power measurements on the AM62L Power Visualizer.
11. Once Linux has booted, run the hmi-script.sh. Reference Appendix 11.1 for an example of what the display must show.
12. When shutting down the demo, run

```
root@am62lxx-evm:~# shutdown -h now
```

9 Summary

Power optimizations for AM62L-based HMI applications are highly dependent on what the HMI application is and how it is being used. Through the use of a practical demo for an HMI use case, this document provides users of AM62L SoC various ways to reduce power consumption on their system such as graphics optimizations, device tree reduction, clock management, memory optimizations, and display optimizations. For aggressive power savings, the Always-On Display low power mode is detailed for systems that are able to suspend while using the HMI application. Based off of the information given, users can determine and implement the power optimizations that best fits the system.

10 References

1. GitHub, https://github.com/lvgl/lvgl, webpage.
2. Texas Instruments, *AM335x Low Power Design Guide*, application note.
3. GitHub, https://git.ti.com/cgit/k3conf/k3conf/, webpage.
4. Texas Instruments, AM62L Technical Reference Manual, webpage.
5. Texas Instruments, AM62L Datasheet, webpage.
6. Texas Instruments, AM62L Power Consumption Summary, webpage.
7. Texas Instruments, AM62L Power Visualizer, webpage.

11 Appendix

11.1 Display Image

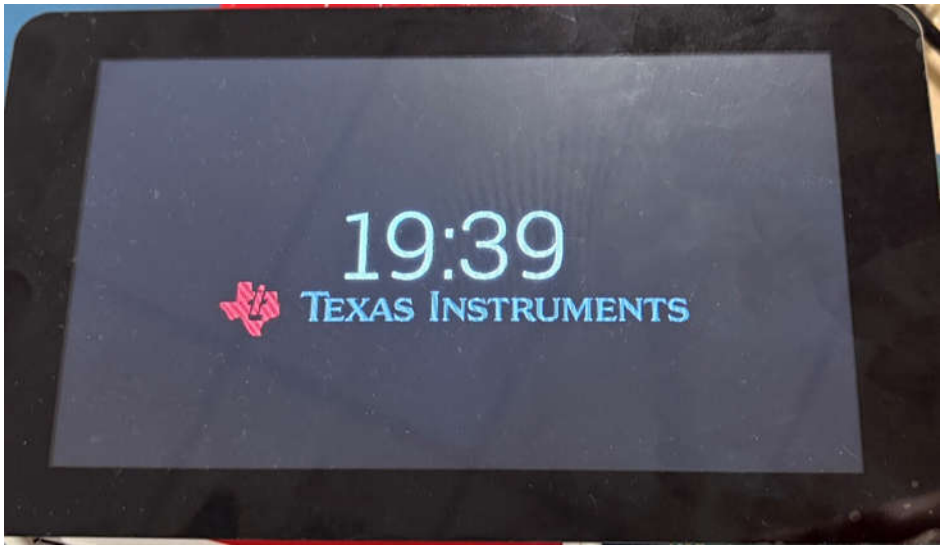

Figure 11-1. Display Image

11.2 Minimal Device Tree Patch

Apply patch to Linux kernel.

```
diff --git a/arch/arm64/boot/dts/ti/Makefile b/arch/arm64/boot/dts/ti/Makefile
index cb2623885b0c..5b61be824f75 100644
--- a/arch/arm64/boot/dts/ti/Makefile
+++ b/arch/arm64/boot/dts/ti/Makefile
@@ -67,6 +67,7 @@ dtb-$(CONFIG_ARCH_K3) += k3-am62l3-badge.dtb

 # Boards with AM62Lx SoCs
 dtb-$(CONFIG_ARCH_K3) += k3-am62l3-evm.dtb
+dtb-$(CONFIG_ARCH_K3) += k3-am62l3-evm-hmi.dtb
 dtb-$(CONFIG_ARCH_K3) += k3-am62l3-evm-dsi-rpi-7inch-panel.dtbo
 dtb-$(CONFIG_ARCH_K3) += k3-am62l3-evm-lpm-wkup-sources.dtbo
 dtb-$(CONFIG_ARCH_K3) += k3-am62l3-evm-m2-cc3351.dtbo
@@ -433,6 +434,8 @@ k3-j784s4-evm-usxgmii-exp1-exp2-dtbs := k3-j784s4-evm.dtb \
     k3-j784s4-evm-usxgmii-exp1-exp2.dtbo
 k3-am62l3-evm-dsi-rpi-7inch-panel-dtbs := k3-am62l3-evm.dtb \
     k3-am62l3-evm-dsi-rpi-7inch-panel.dtbo
+k3-am62l3-evm-hmi-dsi-rpi-7inch-panel-dtbs := k3-am62l3-evm-hmi.dtb \
+    k3-am62l3-evm-dsi-rpi-7inch-panel.dtbo
 k3-am62l3-evm-nand-dtbs := k3-am62l3-evm.dtb \
     k3-am62l3-evm-nand.dtbo
 k3-am62l3-evm-tida-010939-dtbs := k3-am62l3-evm.dtb \
diff --git a/arch/arm64/boot/dts/ti/k3-am62l3-evm-dsi-rpi-7inch-panel.dtso b/arch/arm64/boot/dts/ti/
k3-am62l3-evm-dsi-rpi-7inch-panel.dtso
index 4964469be515..83b7b4204e21 100644
--- a/arch/arm64/boot/dts/ti/k3-am62l3-evm-dsi-rpi-7inch-panel.dtso
+++ b/arch/arm64/boot/dts/ti/k3-am62l3-evm-dsi-rpi-7inch-panel.dtso
@@ -131,7 +131,3 @@ &dphy_tx0 {
 &dsi0 {
     status = "okay";
 };
-
-&sii9022 {
-    status = "disabled";
-};
diff --git a/arch/arm64/boot/dts/ti/k3-am62l3-evm-hmi.dts b/arch/arm64/boot/dts/ti/k3-am62l3-evm-
hmi.dts
new file mode 100644
index 000000000000..e11aca006505
--- /dev/null
+++ b/arch/arm64/boot/dts/ti/k3-am62l3-evm-hmi.dts
@@ -0,0 +1,180 @@
+// SPDX-License-Identifier: GPL-2.0-only or MIT
```

```
+/*
+ * Device Tree file for the AM62L3 Evaluation Module with DSI Display
+ * Copyright (C) 2025 Texas Instruments Incorporated - https://www.ti.com/
+ *
+ * Minimal device tree configured for DSI display support with the RPi 7-inch
+ * touchscreen panel. The k3-am62l3-evm-dsi-rpi-7inch-panel.dtso overlay
+ * must be applied to complete the display configuration.
+ *
+ * Technical Reference Manual: https://www.ti.com/lit/pdf/sprujb4
+ * Data Sheet: https://www.ti.com/lit/pdf/sprspa1
+ */
+
+/dts-v1/;
+
+#include <dt-bindings/gpio/gpio.h>
+#include <dt-bindings/interrupt-controller/irq.h>
+#include "k3-am62l3.dtsi"
+#include "k3-pinctrl.h"
+
+/ {
+    compatible = "ti,am62l3-evm", "ti,am62l3";
+    model = "Texas Instruments AM62L3 Evaluation Module with DSI";
+
+    chosen {
+        stdout-path = &uart0;
+    };
+
+    memory@80000000 {
+        reg = <0x00000000 0x80000000 0x00000000 0x80000000>;
+        device_type = "memory";
+        bootph-all;
+    };
+};
+
+/* Pinctrl configuration for basic peripherals */
+&pmx0 {
+    main_uart0_pins_default: main-uart0-default-pins {
+        pinctrl-single,pins = <
+            AM62LX_IOPAD(0x01b4, PIN_INPUT, 0)    /* (D13) UART0_RXD */
+            AM62LX_IOPAD(0x01b8, PIN_OUTPUT, 0)    /* (C13) UART0_TXD */
+        >;
+        bootph-all;
+    };
+
+    i2c0_pins_default: i2c0-default-pins {
+        pinctrl-single,pins = <
+            AM62LX_IOPAD(0x01cc, PIN_INPUT_PULLUP, 0) /* (B7) I2C0_SCL */
+            AM62LX_IOPAD(0x01d0, PIN_INPUT_PULLUP, 0) /* (A7) I2C0_SDA */
+        >;
+    };
+
+    main_mmc1_pins_default: main-mmc1-default-pins {
+        pinctrl-single,pins = <
+            AM62LX_IOPAD(0x0230, PIN_INPUT, 0) /* (Y3) MMC1_CMD */
+            AM62LX_IOPAD(0x0228, PIN_OUTPUT, 0) /* (Y2) MMC1_CLK */
+            AM62LX_IOPAD(0x0224, PIN_INPUT, 0) /* (AA1) MMC1_DAT0 */
+            AM62LX_IOPAD(0x0220, PIN_INPUT, 0) /* (Y4) MMC1_DAT1 */
+            AM62LX_IOPAD(0x021c, PIN_INPUT, 0) /* (AA2) MMC1_DAT2 */
+            AM62LX_IOPAD(0x0218, PIN_INPUT, 0) /* (AB2) MMC1_DAT3 */
+            AM62LX_IOPAD(0x0234, PIN_INPUT, 0) /* (B6) MMC1_SDCD */
+        >;
+        bootph-all;
+    };
+
+        mmc0_pins_default: mmc0-default-pins {
+                pinctrl-single,pins = <
+                        AM62LX_IOPAD(0x0214, PIN_INPUT, 0) /* (D2) MMC0_CMD */
+                        AM62LX_IOPAD(0x020c, PIN_OUTPUT, 0) /* (B2) MMC0_CLK */
+                        AM62LX_IOPAD(0x0208, PIN_INPUT, 0) /* (D3) MMC0_DAT0 */
+                        AM62LX_IOPAD(0x0204, PIN_INPUT_PULLUP, 0) /* (D4) MMC0_DAT1 */
+                        AM62LX_IOPAD(0x0200, PIN_INPUT_PULLUP, 0) /* (C1) MMC0_DAT2 */
+                        AM62LX_IOPAD(0x01fc, PIN_INPUT_PULLUP, 0) /* (C2) MMC0_DAT3 */
+                        AM62LX_IOPAD(0x01f8, PIN_INPUT_PULLUP, 0) /* (C4) MMC0_DAT4 */
+                        AM62LX_IOPAD(0x01f4, PIN_INPUT_PULLUP, 0) /* (B3) MMC0_DAT5 */
+                        AM62LX_IOPAD(0x01f0, PIN_INPUT_PULLUP, 0) /* (A3) MMC0_DAT6 */
+                        AM62LX_IOPAD(0x01ec, PIN_INPUT_PULLUP, 0) /* (B4) MMC0_DAT7 */
+                >;
+                bootph-all;
```

```
+        };
+};
+
+/* UART0 */
+&uart0 {
+    pinctrl-names = "default";
+    pinctrl-0 = <&main_uart0_pins_default>;
+    bootph-all;
+    status = "okay";
+};
+
+/* I2C0 - Required for DSI display regulator and touch controller */
+&i2c0 {
+    pinctrl-names = "default";
+    pinctrl-0 = <&i2c0_pins_default>;
+    clock-frequency = <400000>;
+    status = "okay";
+};
+
+&sdhci0 {
+        /* eMMC */
+        pinctrl-names = "default";
+        pinctrl-0 = <&mmc0_pins_default>;
+        non-removable;
+        status = "okay";
+        bootph-all;
+};
+
+/* SD/MMC */
+&sdhci1 {
+    pinctrl-names = "default";
+    pinctrl-0 = <&main_mmc1_pins_default>;
+    disable-wp;
+    bootph-all;
+    status = "okay";
+};
+
+/* Display subsystem - DPI output to DSI bridge */
+&dss {
+    status = "okay";
+    bootph-all;
+};
+
+&dss_ports {
+    #address-cells = <1>;
+    #size-cells = <0>;
+
+    /* VP1: DPI Output to DSI Bridge */
+    port@0 {
+        reg = <0>;
+
+        dpi_out: endpoint {
+            remote-endpoint = <&dsi0_in>;
+        };
+    };
+};
+
+/* DSI Controller - DPI to DSI bridge connection */
+&dsi0 {
+    status = "okay";
+    #address-cells = <1>;
+    #size-cells = <0>;
+
+    ports {
+        #address-cells = <1>;
+        #size-cells = <0>;
+
+        /* DSI output port (to bridge/panel) */
+        port@0 {
+            reg = <0>;
+            dsi0_out: endpoint {
+                /* Will be connected by overlay */
+            };
+        };
+
+        /* DSI input port (from DPI) */
+        port@1 {
+            reg = <1>;
```

```
+            dsi0_in: endpoint {
+                remote-endpoint = <&dpi_out>;
+            };
+        };
+    };
+};
+
+/* DSI Physical Layer */
+&dphy_tx0 {
+    status = "okay";
+};
+
+&gpio0 {
+    status = "disabled";
+};
+
+&gpio2 {
+    status = "disabled";
+};
+
+&wkup_rtc0 {
+    status = "disabled";
+};
--
2.34.1
```

11.3 Slow Clock Frequencies

Bash script that can be run in the kernel.

wkup_periph_clk_freqs.sh

```
#!/bin/bash

# Each entry is "clk_id freq"
CLOCK_FREQS=(
    "1 1562500"
    "5 9375000"
    "6 9375000"
    "7 4687500"
    "8 4687500"
    "16 18750000"
    "33 9375000"
    "34 4687500"
    "35 9375000"
    "36 9375000"
    "37 4687500"
    "38 9375000"
    "39 4687500"
    "44 18750000"
    "45 18750000"
    "46 4687500"
    "60 4687500"
    "65 4687500"
    "79 4687500"
    "84 4687500"
    "86 9375000"
    "93 9375000"
    "95 4687500"
    "96 9375000"
    "98 9375000"
    "102 4687500"
    "137 18750000"
    "140 4687500"
    "141 4687500"
    "142 4687500"
    "146 4687500"
    "151 9375000"
    "170 4687500"
    "173 2343750"
    "179 4687500"
    "185 4687500"
    "191 4687500"
    "209 9375000"
    "223 9375000"
    "237 9375000"
    "244 4687500"
```

```
    "248 4687500"
    "252 4687500"
    "256 4687500"
    "260 4687500"
    "267 2343750"
    "271 2343750"
    "278 4687500"
    "284 4687500"
    "285 4687500"
    "286 9375000"
    "287 9375000"
    "289 9375000"
    "291 4687500"
    "292 781250"
    "297 18750000"
    "298 4687500"
    "300 4687500"
    "303 4687500"
    "306 4687500"
    "309 4687500"
    "313 4687500"
    "315 4687500"
    "317 4687500"
    "319 4687500"
    "321 4687500"
    "323 4687500"
    "325 4687500"
    "359 4687500"
    "410 4687500"
    "451 9375000"
    "455 18750000"
    "457 18750000"
    "461 9375000"
    "466 4687500"
    "480 18750000"
    "514 9375000"
    "521 9375000"
)

for entry in "${CLOCK_FREQS[@]}"; do
    clk_id=${entry% *}
    freq=${entry#* }
    echo "Running: k3conf set clock 0 $clk_id $freq"
    k3conf set clock 0 "$clk_id" "$freq" > /dev/null
done
```

11.4 Slow DDR Speed to 800 MT/s Patch

Apply patch to TF-A

```
diff --git a/fdts/am62lx-skevm-lp4-50-800.dtsi b/fdts/am62lx-skevm-lp4-50-800.dtsi
index a498030c5..9daba8d6b 100644
--- a/fdts/am62lx-skevm-lp4-50-800.dtsi
+++ b/fdts/am62lx-skevm-lp4-50-800.dtsi
@@ -11,8 +11,8 @@
 */

 #define DDRSS_PLL_FHS_CNT 3
-#define DDRSS_PLL_FREQUENCY_1 400000000
-#define DDRSS_PLL_FREQUENCY_2 400000000
+#define DDRSS_PLL_FREQUENCY_1 200000000
+#define DDRSS_PLL_FREQUENCY_2 200000000
 #define DDRSS_SDRAM_IDX 15
 #define DDRSS_REGION_IDX 15

--
2.34.1
```

11.5 Enable Auto Self Refresh for DDR

Apply patch to TF-A.

```
diff --git a/fdts/am62lx-skevm-lp4-50-800.dtsi b/fdts/am62lx-skevm-lp4-50-800.dtsi
index 9daba8d6b..a65cde57a 100644
--- a/fdts/am62lx-skevm-lp4-50-800.dtsi
```

```
+++ b/fdts/am62lx-skevm-lp4-50-800.dtsi
@@ -183,9 +183,9 @@
 #define DDRSS_CTL_164_DATA 0x08080801
 #define DDRSS_CTL_165_DATA 0x0000080E
 #define DDRSS_CTL_166_DATA 0x00040003
-#define DDRSS_CTL_167_DATA 0x00000007
-#define DDRSS_CTL_168_DATA 0x00000000
-#define DDRSS_CTL_169_DATA 0x00000000
+#define DDRSS_CTL_167_DATA 0x07074007
+#define DDRSS_CTL_168_DATA 0x00000f07
+#define DDRSS_CTL_169_DATA 0x0303000f
 #define DDRSS_CTL_170_DATA 0x00000000
 #define DDRSS_CTL_171_DATA 0x00000000
 #define DDRSS_CTL_172_DATA 0x00000000
--
2.34.1
```

11.6 Enable Auto Clock Gating

Apply patch to TF-A.

```
diff --git a/plat/ti/k3low/common/am62l_bl31_setup.c b/plat/ti/k3low/common/am62l_bl31_setup.c
index 54da76395..8a77b91ba 100644
--- a/plat/ti/k3low/common/am62l_bl31_setup.c
+++ b/plat/ti/k3low/common/am62l_bl31_setup.c
@@ -6,6 +6,7 @@
  */

 #include <common/debug.h>
+#include <lib/mmio.h>
 #include <lpm_stub.h>
 #include <plat_private.h>
 #include <plat_scmi_def.h>
@@ -66,6 +67,9 @@ int ti_soc_init(void)
     /* Apply firewall configurations over BL31 and BL32 memory regions */
     update_fwl_configs();

+    /* Enable auto clock gating */
+    mmio_write_32(WKUP_CTRL_MMR0_BASE + WKUP_CTRL_MMR0_CLKGATE_CTRL0_OFFSET, 0x0U);
+
     ret = ti_sci_proc_request(PLAT_PROC_START_ID);
     if (ret != 0) {
         ERROR("Unable to request host (%d)\n", ret);
diff --git a/plat/ti/k3low/include/platform_def.h b/plat/ti/k3low/include/platform_def.h
index 2063ebd03..20636bf16 100644
--- a/plat/ti/k3low/include/platform_def.h
+++ b/plat/ti/k3low/include/platform_def.h
@@ -22,6 +22,7 @@
 #define EMIF_CTLCFG_SIZE UL(0x8000)

 #define WKUP_CTRL_MMR0_DEVICE_RESET_OFFSET    UL(0x54000)
+#define WKUP_CTRL_MMR0_CLKGATE_CTRL0_OFFSET    UL(0x54050)

 /* Device Control Region - covers PSC, PLL, and other control registers */
 #define K3LOW_DEVCTRL_BASE    UL(0x200000)    /* Start of device control registers */
--
2.34.1
```

11.7 Slow Panel Pixel Clock by Half Patch

Apply patch to Linux kernel.

```
diff --git a/drivers/gpu/drm/panel/panel-simple.c b/drivers/gpu/drm/panel/panel-simple.c
index 29838b6de12b..a5dfd677f583 100644
--- a/drivers/gpu/drm/panel/panel-simple.c
+++ b/drivers/gpu/drm/panel/panel-simple.c
@@ -3855,7 +3855,7 @@ static const struct panel_desc rocktech_rk043fn48h = {
 };

 static const struct drm_display_mode raspberrypi_7inch_mode = {
-    .clock = 28569600 / 1000,
+    .clock = 28569600 / 2000,
     .hdisplay = 800,
     .hsync_start = 800 + 48,
     .hsync_end = 800 + 48 + 32,
```

```
--
2.34.1
```

11.8 U-Boot Commands

uEnv.txt

```
# This uEnv.txt file can contain additional environment settings that you
# want to set in U-Boot at boot time.  This can be simple variables such
# as the serverip or custom variables.  The format of this file is:
#    variable=value

# NOTE: This file will be evaluated after the bootcmd is run and the
#       bootcmd must be set to load this file if it exists (this is the
#       default on all newer U-Boot images.  This also means that some
#       variables such as bootdelay cannot be changed by this file since
#       it is not evaluated until the bootcmd is run.

# remove console from automatically starting on display
optargs= fbcon=map:2
# RPI touch display overlay
name_overlays=ti/k3-am62l3-evm-dsi-rpi-7inch-panel.dtbo
```

11.9 Enable All Power Optimizations and Run lvglsim Binary

Bash script.

hmi-script.sh

```
#!/bin/bash

# Slow A53 to 200 MHz
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo 200000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed

# Disable A53 core 1
echo 0 > /sys/devices/system/cpu/cpu1/online

sleep .5

# Slow all clocks in WKUP domain
./wkup_periph_clk_freqs.sh

sleep .5

systemctl stop emptty
sleep 1
# Execute Display Demo
./lvglsim
```

11.10 RPI Touch Display Troubleshooting

If Linux cannot detect the RPI touchscreen, remove power from the AM62L and the RPI Touch Display. Power them back up as described in *Run the demo*.

Try again if the same issues occur.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025