



Anshu Madwesh and Kendall Willis

摘要

对于 TI AM62Lx 处理器上的人机界面 (HMI) 应用, 可通过主动优化 (频率扩展、DDR 效率提高) 和空闲状态优化来降低功耗。本文档介绍了这两种 Linux 系统方法, 重点介绍了将常开显示低功耗状态作为一种关键的空闲优化技术。

内容

1 人机界面简介.....	3
2 AM62L 简介.....	3
3 演示概述.....	4
3.1 背景.....	4
4 降低有功功率的方法.....	4
4.1 轻量级图形应用程序.....	4
4.2 最小 Linux 器件树.....	4
4.3 本地电源睡眠控制器 (LPSC).....	5
4.4 降低时钟频率.....	5
4.5 自动时钟门控.....	5
4.6 DDR 处于自动自更新状态.....	5
4.7 性能.....	5
5 特定于演示的优化.....	6
5.1 LVGL 应用.....	6
5.2 最小器件树.....	6
5.3 低时钟速度.....	6
5.4 慢速 A53 内核时钟频率.....	8
5.5 关闭一个 A53 内核.....	8
5.6 DDR 传输速度较慢.....	8
5.7 DDR 自动自刷新.....	8
5.8 自动时钟门控.....	8
5.9 慢速面板 FPS.....	8
6 功耗数据.....	8
7 在 HMI 用例中实施低功耗模式.....	10
8 演示步骤.....	11
8.1 系统要求.....	11
8.2 准备 SD 卡.....	11
8.3 运行演示.....	14
9 总结.....	15
10 参考资料.....	15
11 附录.....	16
11.1 显示图像.....	16
11.2 最小器件树插入码.....	16
11.3 慢时钟频率.....	19
11.4 DDR 速度慢至 800MT/s 插入码.....	20
11.5 为 DDR 启用自动自刷新.....	21
11.6 启用自动时钟门控.....	21
11.7 采用半插入码的慢速面板像素时钟.....	21

11.8 U-Boot 命令.....	22
11.9 启用所有功耗优化并运行 lvgl sim 二进制文件.....	22
11.10 RPI 触摸显示屏故障排除.....	22

商标

所有商标均为其各自所有者的财产。

1 人机界面简介

人机界面 (HMI) 应用广泛应用于各种工业、医疗、消费类电子产品以及其他产品，可供用户监控机器并与机器交互。HMI 通常经由利用显示界面让用户与之交互来提高机器的易用性。随着各理事机构不断颁布能效标准，人们越来越重视设计符合这些要求的产品。具体而言，重点在于降低这些产品的待机功耗，以更大限度地降低空闲功耗，从而降低能源需求。除了在更大程度降低总功耗的同时运行系统外，这些应用中的嵌入式处理器还负责使系统从低功耗空闲状态进入和退出。

2 AM62L 简介

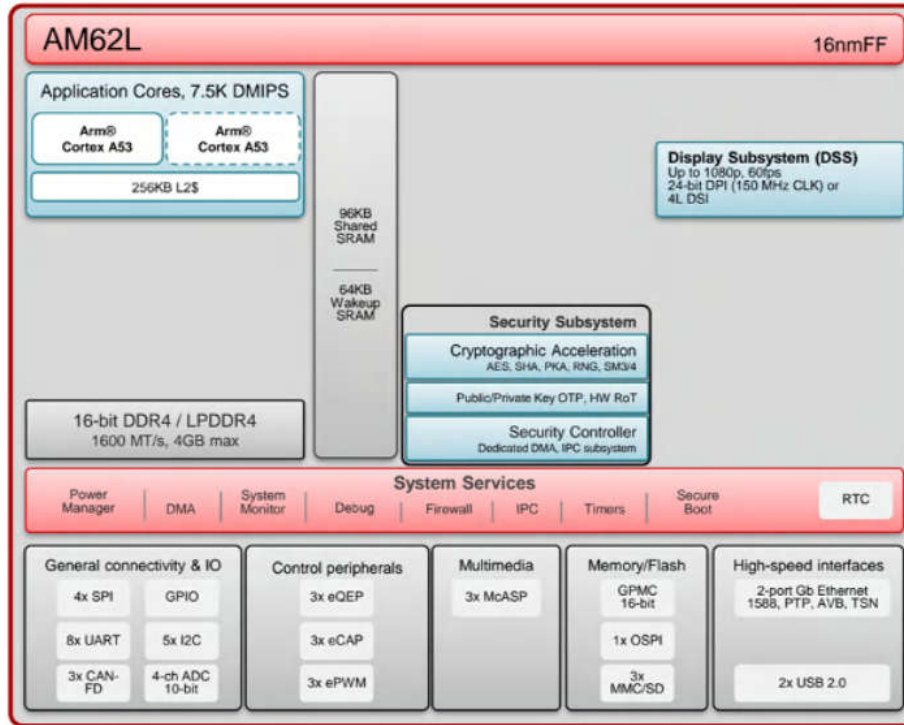


图 2-1. AM62L 方框图

AM62L 是一款基于 ARM 的处理器，是一种低成本、高能效的片上系统 (SoC)，专为 HMI 等应用而设计。主计算块是运行频率高达 1.25GHz 的双核 A53 ARM Cortex。AM62L 提供 1 个 A53 和 2 个 A53 内核设计方案。该处理器包含一个用于 16 位 DDR 接口的存储器控制器，该接口以 1600MT/s 的数据速率运行，用于存储信息并允许高级操作系统 (HLOS) 运行。处理器使用显示子系统 (DSS) 将图形流水线化到支持显示串行接口 (MIPI-DSI) 和显示并行接口 (DPI) 的显示器上。AM62L 还提供各种外设，可与整个系统中的其他 IC 交互。

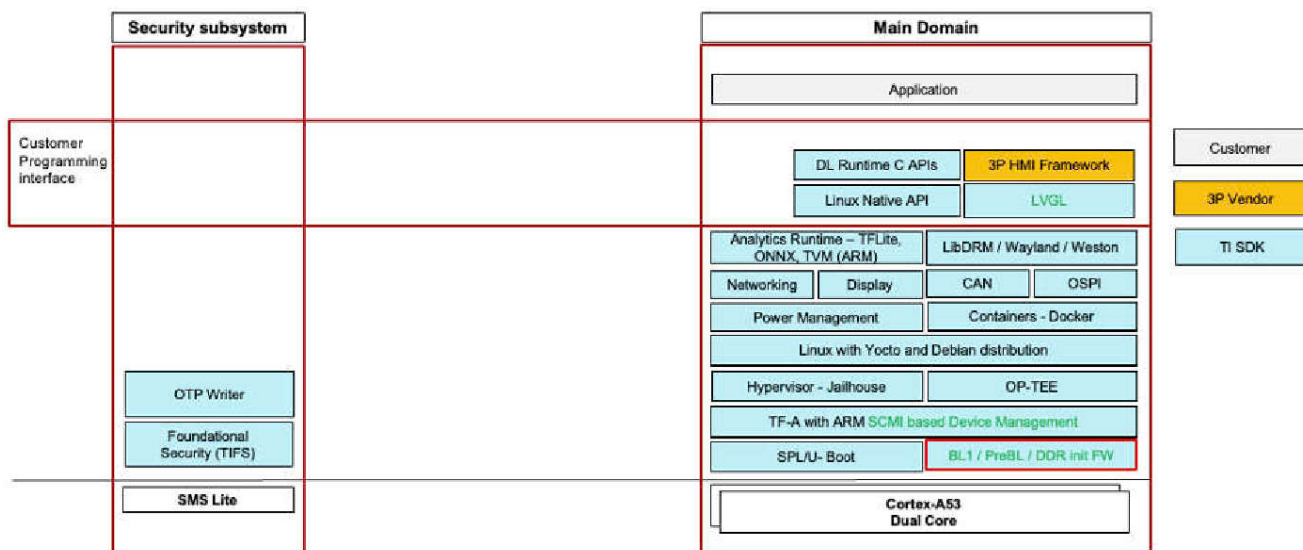


图 2-2. AM62L 软件栈

图 2-2 展示了 AM62L SoC 的软件栈，开发人员可以在其中使用 AM62L SDK 构建应用。

3 演示概述

3.1 背景

AM62L 是一款基于 Arm 的低功耗嵌入式处理器，具有显示功能，可驱动人机界面 (HMI) 应用。本演示展示了 AM62L 在驱动 DSI 连接的显示器时以低功耗活动状态运行 Linux，展示了图形应用。本应用手册讨论了进行低功耗 HMI 应用的主要注意事项并演示了执行情况。

4 降低有功功率的方法

4.1 轻量级图形应用程序

应用内核执行系统的大部分计算繁重任务。对于 HMI 系统，应用内核将负责驱动图形应用程序。设计图形应用程序时，建议使用为嵌入式系统制作的轻量级框架，该框架侧重于低 CPU 和内存使用。由于处理器不必生成大量图形输出，因此在图形应用程序中提供最少的功能可节省能耗。

4.2 最小 Linux 器件树

Linux 器件树提供了一种描述硬件的方法，使用节点图来描述硬件的各个方面。Linux 内核使用它来进行平台识别，运行时配置和器件组装。在器件树中描述外设硬件时，Linux 会启用外设并为其供电，即使它处于空闲状态也是如此。为了省电，应禁用未使用的器件树节点。当禁用节点时，Linux 会忽略器件树参数，外设将保持复位状态。请务必注意，如果节点没有 **status** 属性，Linux 将启用该设备节点。要禁用节点，状态属性应更改为禁用。

此外，器件树节点可能需要配置引脚的多路复用 (PinMux)，外设信号才能正常工作。但是，由于未使用这些禁用的外设，因此不需要定义其各自的 PinMux。该焊盘使接收器和收发器保持禁用状态，同时使 MuxMode 模式保持在 GPIO 状态。这三个元件减少了 I/O 的泄漏电流大小。

4.3 本地电源睡眠控制器 (LPSC)

AM62L 处理器中采用了电源睡眠控制器 (PSC) 架构，该架构负责管理每个外设、控制器和内核的电源和时钟。AM62L 有多个 PSC，每个都有自己的电源域。为了更精细地控制 PSC 内的元件，电源域的子节点称为本地电源睡眠控制器 (LPSC)，允许对各个模块进行软件控制。

LPSC 是软件可控制的，但无法关闭的“常开”域除外。Linux 器件树将初始化每个外设所需的时钟。由于时钟将激活，LPSC 将开启，因此外设将开启并消耗功率。这也在反向模式下工作，如果禁用器件树节点，则不会初始化时钟，从而使 LPSC 保持关闭状态。

如果器件树中未描述 LPSC，则可以更改 LPSC 状态，以便 LPSC 时钟门控且本地复位取消置位。

4.4 降低时钟频率

AM62L 的时钟架构会将外设和内核分组在不同锁相环 (PLL) 中。当 PLL 被禁用时，所有子外设都被禁用。如果 PLL 无法关闭，下一个选项是减少或禁用未使用外设的时钟，同时优化所需外设的时钟速度。

处理器的时钟速度与其动态功耗呈线性关系。在优化嵌入式系统功耗的过程中，应寻找降低时钟频率以降低功耗的机会。

4.5 自动时钟门控

时钟门控是一种电源管理技术，可用于降低 SoC 的动态 (工作/开关) 功耗。当 SoC 内的模块或电路空闲时，可以从时钟树中移除时钟信号。

AM62L 为 SoC 的不同移动部分 (包括通用总线架构 (CBASS)、数据移动架构 (DMA) 和 SRAM) 提供自动时钟门控。AM62L 还为加密模块、USB、MMCSD、显示子系统 (DSS) 等外设提供单独时钟门控选项。

在评估针对系统设计的自动时钟门控时，请考虑元件的活动级别。具有高活动性的组件没有太多机会使用自动时钟门控的节能功能。

4.6 DDR 处于自动自更新状态

DDR 是 AM62L 处理器将用于存储和获取数据 (运行 HLOS 的关键系统) 的外部存储器 IC。当 DDR 和 SoC 之间没有预期事务时，DDR 可以在定义数量的时钟周期后进入自刷新状态。

DDR 自刷新是定期读取 DRAM 并重新写入以保留数据的过程。通过在控制器中定义在多个非活动时钟周期后要进入的低功耗状态，可以自动将 DDR 置于自刷新状态。在系统空闲时，应尽可能频繁地使用 DDR 自动自刷新功能，以节省更多电量。

4.7 性能

在设计嵌入式系统时，关键的权衡是处理器的功耗与性能。通过使用前面部分中讨论的优化方法，围绕系统应用进行优化，可以在不影响性能的情况下更大限度地节省功耗。

5 特定于演示的优化

5.1 LVGL 应用

本演示使用 Light 和多功能图形库 (LVGL) 来构建图形应用程序。LVGL 提供在嵌入式系统上构建图形应用程序的功能，同时关注低 CPU 利用率和内存使用情况。此演示的图形应用程序具有黑屏，用于显示当前时间和 TI 标识。

有关 GUI 的图像，请参阅[附录 11.1](#)。

5.2 最小器件树

最小器件树删除了演示功能不需要的所有节点，并删除了 PinMux 中配置的相应引脚。请参阅[附录 11.2](#) 以获取最小器件树插入码。

5.3 低时钟速度

K3Conf 是一款调试工具，允许开发人员禁用器件、禁用时钟和更改时钟频率。k3conf 打包在 TI Processor SDK 中。使用 `k3conf --help` 查看可以使用的所有命令。检查设备状态的一种简单方法是使用命令 `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 |
```

使用命令 `k3conf set clock 0 <clk_id> <freq>` 更改时钟的频率。例如，

```
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 |
```

未使用 **WKUP** 域中的外设，因此连接它们的时钟应减慢，以尽可能节省功耗。**k3conf** 可用于减慢 **WKUP** 域中的时钟。有关更改内容的完整列表，请参阅[附录 11.3](#)。

5.4 慢速 A53 内核时钟频率

AM62L 具有两个 A53 CPU 内核，配置为在正常运行期间以 1.25GHz 运行。为了在保持系统稳定的同时降低功耗，请将内核速度降低至 200MHz。若要更改 A53 内核的时钟速度，必须更改用于时钟的 PLL 分频器。Linux 中的 CPUFreq 框架可用于从用户空间轻松更改 A53 时钟。CPUFreq 根据它使用的调速器更改 CPU 的频率。用户空间调节器允许用户更改用户空间的时钟频率。可用的时钟频率也可在 userspace 中进行检查。

```
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 关闭一个 A53 内核

系统运行只需一个 A53 内核，因此请关闭其中一个内核。Linux 中的 CPU 热插拔用于关闭其中一个内核。需要使用以下命令来关闭 Linux 控制台中的内核。

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

5.6 DDR 传输速度较慢

DDR 传输速度可以从 1600MT/s 减慢一半至 800MT/s。由于在相同时间内发生的传输较少，因此这种方法功耗更低。DDR 传输速度由 DDR 时钟速度决定。DDR 存储器速度是 PLL 速度的两倍，数据速率是存储器速度的两倍。DDR 时钟速度为 400MHz。如果 DDR 时钟速度为 400MHz，则存储器速度为 800MHz，数据速率为 1600MT/s。为了使数据速率减半，DDR 时钟速度减半以实现 200MHz。有关在 Arm Trusted Firmware 中减慢 DDR 速度的插入码，请参阅[附录 11.4](#)。

5.7 DDR 自动自刷新

AM62L EVM 使用支持自动自刷新的 LPDDR4。DDR 控制器支持三种低功耗状态。启用低功耗状态以及自动进入及退出这些状态。定义尽可能长地进入每个状态的空闲时间量。要设置 DDR 的自动自刷新配置，请参阅[附录 11.5](#)中的 Arm Trusted Firmware 修补程序。

5.8 自动时钟门控

AM62L 为多个控制器提供多种自动时钟门控。要为所有控制器启用自动时钟门控，请向 WKUP_CTRL_MMR_CFG5_CLKGATE_CTRL0 寄存器写入 0。请参阅[附录 11.6](#)，了解如何在 Arm Trusted Firmware 中启用自动时钟门控。

5.9 慢速面板 FPS

显示屏的 FPS（每秒帧数）是每秒显示的帧数。降低 FPS 可以通过减少显示器需要更新的次数来节省电量。在某些情况下，可以使用工具 modetest 来降低显示的 FPS。

RPI 显示屏是 800 × 480 分辨率显示器，在光栅模式下以 60 FPS 的速率运行。对于 RPI 显示，需要在面板简单的 Linux 驱动程序中减慢像素时钟，以减慢 FPS。面板简单 Linux 驱动程序具有 RPI 面板的数据结构，时钟变量具有像素时钟的运行速度。默认情况下，RPI 面板 60FPS，面板像素时钟可减半以实现 30FPS。有关在 Linux 中减慢面板像素时钟频率的插入码，请参阅[附录 11.7](#)。

或者，如果使用的显示面板具有内置 RAM，则可以将 DSS 置于命令模式并自动自刷新，以进一步节能。

6 功耗数据

以下电源数据特定于使用 SDK 11.00.15 的 AM62L。使用其他规格创建相同的演示会产生不同结果。显示的结果旨在展示所述的功耗优化步骤与标准操作系统空闲之间的潜在差异。

以下功率测量是使用所述的已应用电源优化和显示 LVGL 演示的 RPI 触摸显示屏进行的。测量值是每秒进行一次 15 分钟测量时的平均值。

电源轨名称	电压轨电压 (V)	功耗 (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
总计		256.82

为了进行比较，以下功率测量是在连接显示器的情况下在操作系统空闲状态下进行的，但不会进行任何功耗优化。测量值是每秒进行一次 15 分钟测量时的平均值。

电源轨名称	电压轨电压 (V)	功耗 (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
总计		450.56

常规操作系统空闲和低功耗操作系统空闲之间的总差异约为 200mW 的节能量。

7 在 HMI 用例中实施低功耗模式

功耗优化的下一个层面是将低功耗模式纳入系统级设计。低功耗模式提供了在系统空闲时降低功耗的机会。AM62L 提供独特的低功耗状态，称为常开显示模式（也称为 DSS + DeepSleep），使显示子系统及其支持帧缓冲区（存储在片上 RAM 中）能够保持活动状态，而 SoC 的其余部分进入 DeepSleep 低功耗状态。这种低功耗模式旨在将静态图像保留在帧缓冲区中，以便显示子系统可以从帧缓冲区获取、并在尽可能降低功耗的同时保持面板上的静态图像处于活动状态。59 秒后，SoC 将退出常开显示模式，以便应用程序代码可以更新帧缓冲区，然后重新进入常开显示模式。当系统空闲但显示需要保持打开状态时，这种低功耗模式对于 HMI 应用非常有用。

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

图 7-1. AM62L 低功耗模式

表 7-1 列出了常开显示模式的功耗：

表 7-1. 常开显示功耗

电源轨名称	电压轨电压 (V)	功耗 (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
总计		69.99

表 7-2 是对所做的所有优化的直接比较：

表 7-2. 比较常开显示器与操作系统空闲功耗

电源轨名称	电压轨电压 (V)	无优化 (mW)	在显示处于活动状态时优化的操作系统空闲功耗 (mW)	常开显示模式 (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
总计		450.56	256.82	69.99

该低功耗模式将于 SDK 12.1 中添加。如需更多信息，请联系您的 TI 代表或在 e2e.ti.com 上发布问题。

8 演示步骤

8.1 系统要求

- 显示
 - 7 英寸 [Raspberry Pi 触摸显示屏面板](#)
- AM62L
- 32GB SD 卡
- 演示图像
 - [TI SDK 12.00.00.07 映像](#)
 - [Raspberry Pi DSI 覆盖](#)
 - 降低操作系统空闲功耗的脚本
 - LVGL 演示图像

8.2 准备 SD 卡

使用 SDK 12.00.00.07 默认映像刷写 32GB SD 卡：<https://www.ti.com/tool/download/AM62L-LINUX-SDK/12.00.00.07.04>

弹出 SD 卡并重新插入。SD 卡应安装两个目录。一个是 /media/\$USER/root/，以及 /media/\$USER/boot/ 预构建文件以及有关如何在 SD 卡上安装它们的说明可在以下位置找到：<https://github.com/kwillis01/Beyond-SDK/tree/am62l-hmi-demo/am62l/examples/hmi-demo>

8.2.1 构建所需文件

所有文件的构建都是在主机上进行的。

以下 [GitHub 存储库](#) 提供了所有必要的构建文件

按照以下 [说明](#) 安装必要的工具链，以构建各种映像：

为便于使用，克隆每个目录的路径时创建以下环境变量。

```
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 引导映像

构建引导文件需要 U-Boot、OP-TEE、Arm Trusted Firmware (TF-A) 和 ti-linux-firmware。

克隆及构建 TF-A 的 [说明](#)：

将 [附录 11.4](#)、[11.5](#) 和 [11.6](#) 下的修补程序应用于 TF-A，然后构建 TF-A。

克隆及构建 OP-TEE 的 [说明](#)：

克隆及构建 U-Boot 的 [说明](#)：

引导映像文件为 tiboot3.bin、tispl.bin 和 u-boot.img。在主机上，将 tiboot3.bin、tispl.bin 和 u-boot.img 文件从 U-Boot 移至 SD 卡引导分区。

```
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 命令

uEnv.txt 文件允许设置 U-Boot 命令，而无需每次都设置这些命令。

此文件将用于移除连接显示器时将显示的自动控制台，并设置 RPI DSI 触摸显示器的覆盖层。

uEnv.txt 文件的内容见附录 11.8。将 uEnv.txt 移至 SD 卡引导分区。

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

8.2.1.3 GUI 文件

lvglsim 文件是 LVGL 图形应用程序的二进制文件。

克隆和编译 ti-lvgl-demo 的说明：<https://github.com/kwillis01/ti-lvgl-demo/blob/hmi-simple-demo/README.md>

首先、克隆 ti-lvgl-demo 存储库并切换至 hmi-simple-demo 分支

```
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
```

按照本教程设置 Docker。构建 LVGL 演示。

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

将 lvglsim 二进制文件从 ti-lvgl-demo/lv_port_linux/bin/ 移至 SD 卡根分区中的根目录。

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

从[此处](#)下载 img_lv_demo_simple_high_res_ti_logo.png。

将目录 ti-lvgl-demo/assets/ 设置为文件夹 /media/\$USER/root/usr/share

将 img_lv_demo_simple_high_res_ti_logo.png 移至 /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 内核映像、模块和器件树

克隆及构建 Linux 的说明：

将附录 11.2 和 11.7 下的插入码应用到 Linux 内核，然后构建内核映像和器件树文件。

将 k3-am6213-evm-hmi.dtb 移至 /media/\$USER/root/boot/dtb/ti/k3-am6213-evm.dtb 并且将 k3-am6213-evm-dsi-rpi-7inch-panel.dtbo 移至 /media/\$USER/root/boot/dtb/ti/

```
HOST$ sudo cp $LINUX_DIR/arch/arm64/boot/dts/ti/k3-am6213-evm-hmi.dtb /media/$USER/root/
boot/dtb/ti/k3-am6213-evm.dtb
HOST$ sudo cp $LINUX_DIR/arch/arm64/boot/dts/ti/k3-am6213-evm-dsi-rpi-7inch-panel.dtbo /media/$USER/
root/boot/dtb/ti/
```

将内核映像和模块安装在 boot 目录的 SD 根分区上。

```
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 脚本

使用附录 11.3 中的内容，创建一个名为 wkup_periph_clk_freqs.sh 的文件，并使用附录 11.9 中的内容创建另一个名为 hmi-script.sh 的文件。

这些脚本添加了之前的所有功耗优化。它还会启动 lvglsim 二进制文件。脚本在创建时是可选的，但必须执行脚本中列出的所有命令才能运行演示。

将脚本设置为可执行：

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

将 hmi-script.sh 移至 /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 运行演示

1. 构建所有文件并将其安装到 SD 卡后，将其从主机上弹出。
2. 将 SD 卡插入 AM62L。
3. 将带状电缆插入 RPI 触摸显示屏。端口将位于显示屏背面。连接器插针应背对显示屏。
4. 在 AM62L 的背面，将带状电缆连接到 DSI_CONN 端口。连接器引脚应面向端口的白色部分。
5. 从主机上，分别将两根 Micro USB 线缆插入 AM62L UART 和 JTAG 端口。
6. 打开 AM62L Power Visualizer 工具，并按照页面中的“如何设置工具”进行操作：<https://dev.ti.com/gallery/view/SitaraMPU/AM62L%5FPower%5FVisualizer/ver/1.0.0/>
7. 在主机上，为 Linux UART 控制台打开 /dev/ttyUSB0。
8. 将电源插入 AM62L 上的 USB C 端口。
9. 当 Linux 开始启动时，为 RPI 显示供电。

备注

请勿在启动时过晚执行此操作，否则无法检测到显示屏。

-
10. 开始在 AM62L Power Visualizer 上流式传输功率测量值。
 11. Linux 引导完成后，运行 hmi-script.sh。有关显示屏必须显示的内容示例、请参考附录 11.1。
 12. 关闭演示时，运行

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

9 总结

基于 AM62L 的 HMI 应用的电源优化在很大程度上取决于 HMI 应用的内容及其使用方式。通过针对 HMI 用例使用实际演示，本文档为 AM62L SoC 用户提供了各种降低系统功耗的方法，例如图形优化、器件树精简、时钟管理、存储器优化和显示优化。为了大幅降低功耗，这里为能够在使用 HMI 应用时暂停的系统详细介绍了常开显示低功耗模式。根据给定的信息，用户可以确定和实施更适合系统的电源优化。

10 参考资料

1. GitHub, <https://github.com/lvgl/lvgl>, 网页。
2. 德州仪器 (TI), [AM335x 低功耗设计指南](#), 应用手册。
3. GitHub, <https://git.ti.com/cgit/k3conf/k3conf/>, 网页。
4. 德州仪器 (TI), [AM62L 技术参考手册](#), 网页。
5. 德州仪器 (TI), [AM62L 数据表](#), 网页。
6. 德州仪器 (TI), [AM62L 功耗总结](#), 网页。
7. 德州仪器 (TI), [AM62L 电力可视化仪](#), 网页。

11 附录

11.1 显示图像

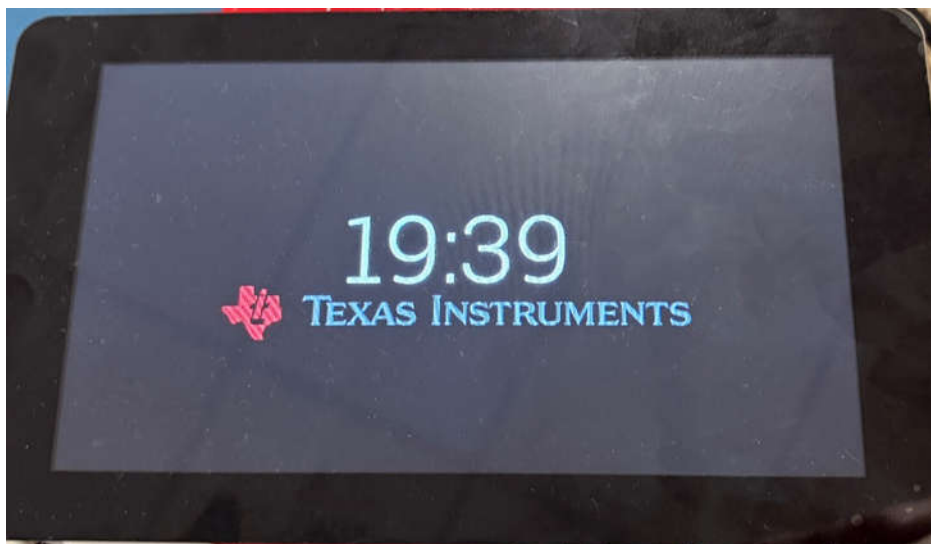


图 11-1. 显示图像

11.2 最小器件树插入码

将插入码应用于 Linux 内核。

```
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-am6213-badger.dtb

# Boards with AM62Lx SoCs
dtb-$(CONFIG_ARCH_K3) += k3-am6213-evm.dtb
+dtb-$(CONFIG_ARCH_K3) += k3-am6213-evm-hmi.dtb
dtb-$(CONFIG_ARCH_K3) += k3-am6213-evm-dsi-rpi-7inch-panel.dtbo
dtb-$(CONFIG_ARCH_K3) += k3-am6213-evm-lpm-wkup-sources.dtbo
dtb-$(CONFIG_ARCH_K3) += k3-am6213-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-am6213-evm-dsi-rpi-7inch-panel-dtbs := k3-am6213-evm.dtb \
k3-am6213-evm-dsi-rpi-7inch-panel.dtbo
+k3-am6213-evm-hmi-dsi-rpi-7inch-panel-dtbs := k3-am6213-evm-hmi.dtb \
+ k3-am6213-evm-dsi-rpi-7inch-panel.dtbo
+ k3-am6213-evm-nand-dtbs := k3-am6213-evm.dtb \
k3-am6213-evm-nand.dtbo
k3-am6213-evm-tida-010939-dtbs := k3-am6213-evm.dtb \
diff --git a/arch/arm64/boot/dts/ti/k3-am6213-evm-dsi-rpi-7inch-panel.dtso b/arch/arm64/
boot/dts/ti/k3-am6213-evm-dsi-rpi-7inch-panel.dtso
index 4964469be515..83b7b4204e21 100644
--- a/arch/arm64/boot/dts/ti/k3-am6213-evm-dsi-rpi-7inch-panel.dtso
+++ b/arch/arm64/boot/dts/ti/k3-am6213-evm-dsi-rpi-7inch-panel.dtso
@@ -131,7 +131,3 @@ &dphy_tx0 {
&dsi0 {
status = "okay";
};
-
-&si9022 {
status = "disabled";
-};
diff --git a/arch/arm64/boot/dts/ti/k3-am6213-evm-hmi.dts b/arch/arm64/boot/dts/ti/k3-am6213-evm-
hmi.dts
new file mode 100644
index 000000000000..e11aca006505
--- /dev/null
+++ b/arch/arm64/boot/dts/ti/k3-am6213-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.dts 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/sprspal
+ */
+
+/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>;
+          dsio_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 慢时钟频率

可在内核中运行的 Bash 脚本。

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 DDR 速度慢至 800MT/s 插入码

将插入码应用于 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 为 DDR 启用自动自刷新

将插入码应用于 TF-A。

```
diff --git a/fdts/am62l1x-skevm-lp4-50-800.dtsi b/fdts/am62l1x-skevm-lp4-50-800.dtsi
index 9daba8d6b..a65cde57a 100644
--- a/fdts/am62l1x-skevm-lp4-50-800.dtsi
+++ b/fdts/am62l1x-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 启用自动时钟门控

将插入码应用于 TF-A。

```
diff --git a/plat/ti/k3low/common/am62l_b131_setup.c b/plat/ti/k3low/common/am62l_b131_setup.c
index 54da76395..8a77b91ba 100644
--- a/plat/ti/k3low/common/am62l_b131_setup.c
+++ b/plat/ti/k3low/common/am62l_b131_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_fw_configs();

+ /* Enable auto clock gating */
+ mmio_write_32(WKUP_CTRL_MMR0_BASE + WKUP_CTRL_MMR0_CLKGATE_CTRL0_OFFSET, 0x0U);
+
ret = ti_scmi_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_CTL_CFG_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 采用半插入码的慢速面板像素时钟

将插入码应用于 Linux 内核。

```
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 命令

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-am6213-evm-dsi-rpi-7inch-panel.dtbo
```

11.9 启用所有功耗优化并运行 lvgl sim 二进制文件

Bash 脚本。

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 emptyty
sleep 1
# Execute Display Demo
./lvgl sim
```

11.10 RPI 触摸显示屏故障排除

如果 Linux 无法检测到 RPI 触摸屏，请断开 AM62L 和 RPI 触摸显示屏的电源。按照运行演示中所述将它们重新启动。

如果出现相同的问题，请重试。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 [TI 销售条款](#)、[TI 通用质量指南](#) 或 [ti.com](#) 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

版权所有 © 2026，德州仪器 (TI) 公司

最后更新日期：2025 年 10 月