

Application Note

AM62L を採用した低消費電力ヒューマン マシン インターフェイス
を構築する

Anshu Madwesh and Kendall Willis

概要

TI の AM62Lx プロセッサのヒューマン マシン インターフェイス (HMI) アプリケーションでは、アクティブ最適化 (周波数スケーリング、DDR の効率の向上) とアイドル状態の最適化により、消費電力を低減できます。このドキュメントでは、Linux システムの両方の方法について説明し、重要なアイドル最適化手法として常時オンのディスプレイ低消費電力状態を強調します。

目次

1 ヒューマン マシン インターフェイスの紹介.....	3
2 AM62L の概要.....	3
3 デモの概要.....	4
3.1 背景.....	4
4 動作時電力を低減する手法.....	4
4.1 軽量グラフィックス アプリケーション.....	4
4.2 最小限の Linux デバイス ツリー.....	4
4.3 ローカル パワー スリープ コントローラ (LPSC).....	5
4.4 クロック周波数の低減.....	5
4.5 自動クロック ゲーティング.....	5
4.6 オート セルフ リフレッシュにおける DDR.....	5
4.7 性能.....	5
5 具体的な最適化のデモ.....	6
5.1 LVGL アプリケーション.....	6
5.2 最小デバイス ツリー.....	6
5.3 低いクロック速度.....	6
5.4 低速の A53 コア クロック周波数.....	8
5.5 1 つの A53 コア をオフにする.....	8
5.6 遅い DDR 転送速度.....	8
5.7 DDR オート セルフ リフレッシュ.....	8
5.8 自動クロック ゲーティング.....	8
5.9 遅いパネル FPS.....	8
6 電力データ.....	9
7 HMI ユース ケース内での低電力モードの実装.....	10
8 デモ ステップ.....	11
8.1 システム要件.....	11
8.2 SD カードを準備.....	11
8.3 デモを実行.....	14
9 まとめ.....	15
10 参考資料.....	15
11 付録.....	16
11.1 画像を表示.....	16
11.2 最小デバイス ツリー パッチ.....	16
11.3 低クロック周波数.....	19
11.4 DDR 速度を 800 MT/s に下げるパッチ.....	20

11.5 DDR のオートセルフリフレッシュを有効にする.....	21
11.6 自動クロックゲーティングを有効にする.....	21
11.7 ハーフパッチによる低速パネルピクセルクロック.....	21
11.8 U-Boot コマンド.....	22
11.9 すべての電力最適化を有効にして、lvglsim バイナリを実行する.....	22
11.10 RPI タッチディスプレイのトラブルシューティング.....	22

商標

すべての商標は、それぞれの所有者に帰属します。

1 ヒューマン マシン インターフェイスの紹介

ヒューマン マシン インターフェイス (HMI) アプリケーションは、機械の監視や操作を行うための、産業、医療、コンシューマ エレクトロニクスなどの多様な製品に使われています。HMI は、通常はユーザーが対話するためのディスプレイ インターフェイスを活用する方法で、機械の使いやすさを改善します。エネルギー効率の基準が政府機関によって制定され続けている現在、これらの要件を満たす製品の設計にますます重点が置かれています。特に、アイドル時の消費電力を最小化し、エネルギー需要を低減するために、これらの製品のスタンバイ電力を低減することに重点が置かれています。これらのアプリケーションの組み込みプロセッサは、システムを動作させながら、総消費電力を最小限に抑えながら、低消費電力のアイドル状態からシステムへの出入りを実行します。

2 AM62L の概要

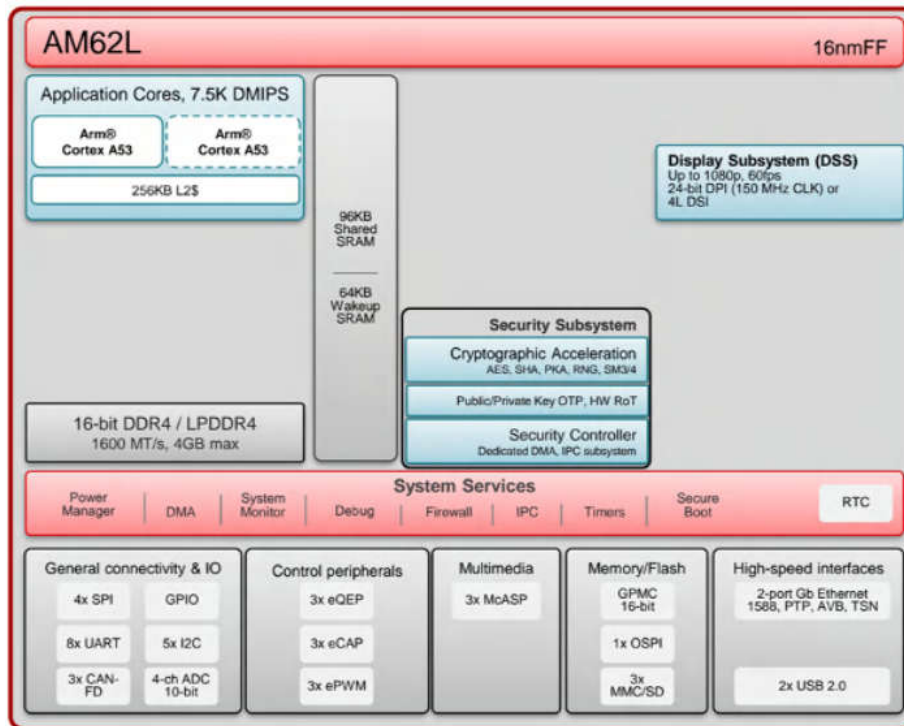


図 2-1. AM62L のブロック図

AM62L は、低コストで電力効率の優れたシステム オンチップ (SoC) で、HMI などのアプリケーション向けに設計された ARM ベースのプロセッサです。メイン計算ブロックは、最大 1.25GHz で動作可能な 2 コア A53 ARM Cortex です。AM62L は、1 個の A53 と 2 個の A53 コア デザインを提供します。プロセッサには、1600MT/s のデータレートで動作する 16 ビット DDR インタフェース用のメモリ コントローラが搭載されており、このコントローラは情報を保存するために使用され、HLOS (High Level Operating System) が動作できるようになっています。このプロセッサは DSS (ディスプレイ サブシステム) を使用して、MIPI-DSI (ディスプレイ シリアル インターフェイス) と DPI (ディスプレイ パラレル インターフェイス) の両方をサポートするディスプレイにグラフィックスのパイプラインを形成します。また、AM62L は、システム全体内の他の IC と相互動作するためのさまざまなペリフェラルも搭載しています。

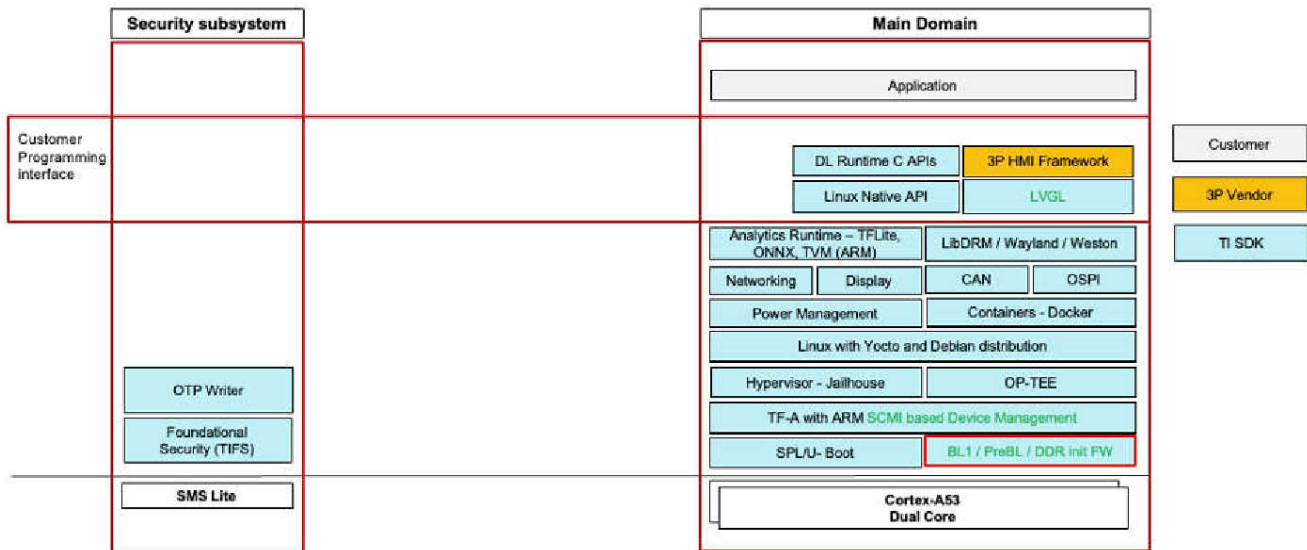


図 2-2. AM62L ソフトウェア スタック

図 2-2 に、AM62L SoC のソフトウェア スタックを示します。開発者は AM62L SDK を使用してアプリケーションを構築できます。

3 デモの概要

3.1 背景

AM62L は、ヒューマン マシン インターフェイス (HMI) アプリケーションを駆動するためのディスプレイ機能を搭載した、低消費電力の Arm ベース組込みプロセッサです。このデモでは、グラフィック アプリケーションを示す DSI 接続ディスプレイを駆動しながら、AM62L が低消費電力のアクティブ状態で Linux を動作させる方法を示します。このアプリケーション ノートは、消費電力が最適化された HMI アプリケーションを製作するための主な検討事項について解説し、その実行のデモを実施します。

4 動作時電力を低減する手法

4.1 軽量グラフィックス アプリケーション

アプリケーション コアは、システムの計算上の重量物を持ち上げる作業のほとんどを実行します。HMI システムの場合、アプリケーション コアがグラフィックス アプリケーションを駆動します。グラフィックス アプリケーションを設計する場合は、CPU とメモリの使用量が少ない組込みシステム向けに作られた軽量フレームワークを使用することをお勧めします。グラフィックス アプリケーションの機能を最小限に抑えることで、プロセッサは大きなグラフィカル出力を生成する必要がないため、消費電力を削減できます。

4.2 最小限の Linux デバイス ツリー

Linux デバイス ツリーは、ノードのグラフを使用してハードウェアの側面を記述することによって、ハードウェアを記述する方法を提供します。Linux カーネルでは、プラットフォームの識別、実行時の設定、デバイスの取得に使用されます。ペリフェラル ハードウェアがデバイス ツリーに記述されると、ペリフェラルがアイドル状態であっても、Linux はペリフェラルをイネーブルにして電源を供給します。消費電力を節約するには、未使用のデバイス ツリー ノードを無効にする必要があります。ノードがディセーブルのとき、Linux はデバイス ツリー パラメータを無視し、ペリフェラルはリセット状態のままになります。ノードにステータス プロパティがない場合、Linux はデバイス ノードを有効にすることに注意してください。ノードを無効にするには、ステータス プロパティをディセーブルに変更する必要があります。

さらに、ペリフェラルの信号を機能させるために、デバイス ツリー ノードでピン多重化 (PinMux) を構成する必要があります。ただし、これらのディセーブル ペリフェラルは使用されていないため、それぞれの PinMux を定義する必要はありません。パッドは、MuxMode を GPIO 状態に維持するとともに、レシーバとトランシーバをディセーブルに維持します。これら 3 つの成分により、I/O のリーク電流の量が減少します。

4.3 ローカル パワー スリープ コントローラ (LPSC)

AM62L プロセッサには電源スリープ コントローラ (PSC) アーキテクチャが搭載されており、各ペリフェラル、コントローラ、コアの電源とクロックを管理します。AM62L には複数の PSC があり、それぞれが独自の電源ドメインを持っています。PSC 内のコンポーネントの制御をより細かくするために、ローカル パワー スリープ コントローラ (LPSC) と呼ばれる電力ドメインの子ノードがあり、これを使用すると個々のモジュールのソフトウェア制御が可能になります。

LPSC は、オフにできない「常時オン」ドメインを除いて、ソフトウェアで制御できます。Linux デバイス ツリーが各ペリフェラルに必要なクロックを初期化します。クロックがアクティブになるので、LPSC はオンになり、ペリフェラルはオンになり、消費電力が発生します。これは逆の場合も動作し、デバイス ツリー ノードが無効になっていればクロックは初期化されず、LPSC はオフのままになります。

LPSC がデバイス ツリーに記述されていない場合は、LPSC の状態を変更して、LPSC がクロック ゲートされ、ローカル リセットがデアサートされるようにできます。

4.4 クロック周波数の低減

AM62L のクロック アーキテクチャは、さまざまなフェーズ ロック ループ (PLL) 内にペリフェラルとコアをグループ化します。PLL がディセーブルになると、すべての子ペリフェラルがディセーブルになります。PLL をオフにできない場合、次のオプションは、必要なペリフェラルのクロック速度を最適化しながら、未使用のペリフェラルのクロックを低減またはディセーブルにすることです。

プロセッサのクロック速度は、その動的な消費電力と線形の関係にあります。組込みシステムの消費電力を最適化するには、クロック周波数を低速にして消費電力を低減する機会を見つけます。

4.5 自動クロック ゲーティング

クロック ゲーティングは、SoC の動的 (アクティブ/スイッチング) 消費電力を低減するために利用できるパワー マネージメント技術です。SoC 内のモジュールまたは回路がアイドルのとき、クロック信号をクロック ツリーから除去できます。

AM62L は、CBASS (Common Bus Architecture、共通バス アーキテクチャ)、DMA (データ移動アーキテクチャ)、SRAM など、SoC のさまざまな移動部分に自動クロック ゲーティングを提供します。AM62L は、暗号化モジュール、USB、MMCSD、ディスプレイ サブシステム (DSS) などのペリフェラル用の個別クロック ゲーティング オプションも備えています。

システム設計の自動クロック ゲーティングを評価するときは、コンポーネントのアクティビティレベルを考慮してください。アクティビティが高いコンポーネントでは、自動クロック ゲーティングの省電力を使用する機会は多くありません。

4.6 オート セルフ リフレッシュにおける DDR

DDR は、AM62L プロセッサが HLOS を実行している重要なシステムであるデータの保存とフェッチに使用する外部メモリ IC です。DDR と SoC の間に予期されるトランザクションがない場合、定義されたクロックサイクル数が経過した後、DDR をセルフ リフレッシュに移行できます。

DDR セルフ リフレッシュは、DRAM が定期的に読み取られ、データを維持するために再度書き込まれるプロセスです。DDR は、非アクティブクロック サイクル数が経過した後に低消費電力状態に移行するコントローラを定義することにより、自動的にセルフ リフレッシュに移行できます。より多くの電力を節約するために、システムがアイドル状態にあるときは、DDR 自動セルフ リフレッシュ機能を、できるだけ頻繁に使用する必要があります。

4.7 性能

組込みシステムを設計する場合、主なトレード オフはプロセッサの消費電力と性能です。システム アプリケーション周辺の最適化により、前のセクションで説明した最適化方法を使用して、性能に影響を及ぼすことなく最大の電力削減を実現できます。

5 具体的な最適化のデモ

5.1 LVGL アプリケーション

このデモでは、Light and Versatile Graphics Library (LVGL) を使用して、グラフィック アプリケーションを構築します。LVGL は、低 CPU 使用率とメモリ使用量に焦点を当てながら、組込みシステムでグラフィックス アプリケーションを構築するための機能を提供します。このデモのグラフィックス アプリケーションでは、現在の時刻と TI のロゴを表示する黒い画面が使用されています。

GUI の画像については、[付録 11.1](#) を参照してください。

5.2 最小デバイス ツリー

最小デバイス ツリーは、デモの機能に必要なすべてのノードを削除し、PinMux で設定されていた対応するピンを削除します。最小デバイス ツリーのパッチについては、[付録 11.2](#) を参照してください。

5.3 低いクロック速度

K3Conf は、デバイスの無効化、クロックの無効化、クロック周波数の変更を可能にするデバッグ ツールです。k3conf は TI プロセッサ SDK の一部としてパッケージされています。k3conf --help を使用して、使用できるすべてのコマンドを表示します。コマンド k3conf dump clocks を使用すると、デバイスのステータスを簡単に確認できます。

```
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 |
```

コマンド k3conf set clock 0 <clk_id> <freq> を使用して、クロック周波数を変更します。たとえば、

```
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 '0' 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 |

```

WKUP ドメインのペリフェラルは使用されないため、それらを接続するクロックは、最大限の消費電力削減のために低速化する必要があります。k3conf は WKUP ドメインのクロックを遅くするために使用できます。変更内容の完全なリストについては、[付録 11.3](#) を参照してください。

5.4 低速の A53 コア クロック周波数

AM62L には、通常動作時に 1.25GHz で動作するように構成された 2 つの A53 CPU コアが搭載されています。システムの安定性を維持しながら消費電力を低減するには、コアの速度を 200MHz に合わせて下げます。A53 コアのクロック速度を変更するには、クロックに使用する PLL 分周器を変更する必要があります。Linux 内の CPUFreq フレームワークを使用して、A53 クロックをユーザー空間から簡単に変更できます。CPUFreq は、使用するガバナに基づいて CPU の周波数を変更します。ユーザー空間 ガバナでは、ユーザー空間からクロック周波数を変更できます。使用可能なクロック周波数はユーザー空間で確認することもできます。

```
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 1 つの A53 コアをオフにする

システムの動作に必要な A53 コアは 1 つだけなので、そのうち 1 つをオフにしてください。Linux の CPU ホットプラグは、いずれかのコアをオフにするために使用されます。Linux コンソールでコアをオフにするには、次のコマンドが必要です。

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

5.6 遅い DDR 転送速度

DDR 転送速度は 1600 MT/s から 800 MT/s の半分に遅くすることができます。これにより、同じ時間内に発生する転送数が少なくなるため、消費電力が少なくなります。DDR 転送速度は、DDR クロック速度によって決定されます。DDR のメモリ速度は PLL の 2 倍で、データレートはメモリの 2 倍です。DDR クロック速度は 400MHz です。DDR クロック速度が 400MHz の場合、メモリ速度は 800MHz、データレートは 1600MT/s です。データレートを半分にするために、DDR のクロック速度を 200MHz に半分にします。Arm Trusted Firmware で DDR を低速にするパッチについては、[付録 11.4](#) を参照してください。

5.7 DDR オート セルフリフレッシュ

AM62L 評価基板は LPDDR4 を使用しており、自動セルフリフレッシュが可能です。DDR コントローラは、3 つの低消費電力状態をサポートしています。低消費電力状態と、自動的に開始および終了します。各状態にできる限り入るアイドル時間を定義します。DDR のオートセルフリフレッシュ構成を設定するには、[付録 11.5](#) の Arm Trusted Firmware パッチを参照してください。

5.8 自動クロック ゲーティング

AM62L は、複数のコントローラ向けに複数の自動クロック ゲーティング機能を搭載しています。すべてのコントローラで自動クロック ゲーティングをイネーブルにするには、WKUP_CTRL_MMR_CFG5_CLKGATE_CTRL0 レジスタに 0 を書き込みます。Arm Trusted Firmware で自動クロック ゲーティングを有効にする方法については、[付録 11.6](#) を参照してください。

5.9 遅いパネル FPS

ディスプレイの FPS (フレーム/秒) は、1 秒あたりに表示されるフレームの量です。FPS を遅くすると、ディスプレイの更新時間を短くすることで、省電力を実現できます。場合によっては、ツール モード テストを使用して、ディスプレイの FPS を減らすことができます。

RPI ディスプレイは、ラスタ モードで 60 FPS で動作する 800×480 の解像度のディスプレイです。RPI のディスプレイでは、FPS を下げるために、Linux パネル スインプルドライバでピクセル クロックの速度を落とす必要があります。Linux パネル スインプルドライバには RPI パネル用のデータ構造があり、クロック変数にはピクセル クロックが動作している速度が格納されています。デフォルトでは、RPI パネル 60 FPS、パネルのピクセル クロックを半分にすることができます 30

FPS を実現します。Linux のパネル ピクセル クロック周波数を遅くするパッチについては、[付録 11.7](#) を参照してください。

また、使用中のディスプレイ パネルに RAM が内蔵されている場合は、DSS をコマンド モードにして自動セル フリフレッシュにし、さらに省電力化することもできます。

6 電力データ

以下の電源データは、SDK 11.00.15 を使用する AM62L 固有です。他の仕様を使用して同じデモを作成すると、異なる結果が得られます。表示されている結果は、説明した電力最適化手順と標準 OS アイドル状態の間の潜在的な違いを示しています。

以下の電力測定は、説明した電力最適化の適用と LVGL のデモを示す RPI タッチ ディスプレイを使用して行われました。測定値は、15 分間 1 秒間に 1 回測定された平均値です。

レール名	レール電圧 (V)	電力 (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
合計		256.82

比較のため、以下の電力測定値は、ディスプレイを接続した状態で OS アイドル状態ですが、電力最適化は行われていません。測定値は、15 分間 1 秒間に 1 回測定された平均値です。

レール名	レール電圧 (V)	電力 (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
合計		450.56

通常の OS アイドル時と電力最適化 OS アイドル時の合計差は、約 200mW の省電力です。

7 HMI ユース ケース内での低電力モードの実装

電力最適化の次のレベルは、システム レベルの設計に低消費電力モードを搭載することです。低消費電力モードでは、システムのアイドル状態が可能ときに消費電力を低減できます。AM62L は、常時オン ディスプレイ モード (DSS+ DeepSleep と呼ばれる) と呼ばれる独自の低消費電力状態を提供します。これにより、ディスプレイ サブシステムと、オンチップ RAM に保存されたそのサポート フレーム バッファは、SoC の残りの部分が DeepSleep 低消費電力状態に移行している間はアクティブのままになります。この低消費電力モードは、ディスプレイ サブシステムがフレーム バッファ内に静的イメージを保持するように設計されており、ディスプレイ サブシステムがフレーム バッファからフェッチしてパネル上の静的画像をアクティブに維持しながら、消費電力を最小限に抑えることができます。59 秒後に、SoC は常時オン ディスプレイ モードを終了します。これにより、アプリケーション コードはフレーム バッファを更新してから、常時オンのディスプレイ モードに再移行できます。この低消費電力モードは、システムがアイドル状態にあるが、ディスプレイをオンのままにする必要がある場合の HMI アプリケーションに役立ちます。

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

図 7-1. AM62L の低消費電力モード

表 7-1 に、常時オン ディスプレイ モードの消費電力を示します。

表 7-1. 常時オン ディスプレイの消費電力

レール名	レール電圧 (V)	電力 (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
合計		69.99

表 7-2 は、実行されたすべての最適化を直接比較したものです。

表 7-2. 常時稼働のディスプレイと OS アイドル時の消費電力の比較

レール名	レール電圧 (V)	最適化なし (mW)	ディスプレイがアクティブな状態で最適化された OS アイドル (mW)	常時オン ディスプレイ モード (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

表 7-2. 常時稼働のディスプレイと OS アイドル時の消費電力の比較 (続き)

レール名	レール電圧 (V)	最適化なし (mW)	ディスプレイがアクティブな状態で最適化された OS アイドル (mW)	常時オン ディスプレイ モード (mW)
合計		450.56	256.82	69.99

この低消費電力モードは、SDK 12.1 で追加される予定です。詳細については、TI の担当者にお問い合わせになるか、e2e.ti.com に質問を投稿してください。

8 デモ ステップ

8.1 システム要件

- ディスプレイ
 - 7 インチ ラズベリー パイ タッチ パネル
- AM62L
- 32 GB SD カード
- デモ イメージ
 - TI SDK 12.00.00.07 の画像
 - Raspberry PI DSI オーバーレイ
 - OS のアイドル時の電力を削減するスクリプト
 - LVGL デモ イメージ

8.2 SD カードを準備

32 GB SD カードを SDK 12.00.00.07 のデフォルト イメージ <https://www.ti.com/tool/download/AM62L-LINUX-SDK/12.00.00.07.04> でフラッシュします

SD カードを取り出し、もう一度挿入します。SD カードは 2 つのディレクトリでマウントする必要があります。1 つは /media/\$USER/root/ と /media/\$USER/boot/

ビルド済みファイルと SD カードへのインストール方法については、<https://github.com/kwillis01/Beyond-SDK/tree/am62l-hmi-demo/am62l/examples/hmi-demo> を参照してください

8.2.1 必要なファイルをビルド

すべてのファイルの作成は、ホスト マシンから行われます。

次の GitHub [リポジトリ](#)は、必要なすべてのビルドファイルを提供します

次の手順に従って、さまざまなイメージを構築するために必要なツール チェーンをインストールします。

簡単に使用できるように、各ディレクトリのクローン作成時に、次の環境変数を各ディレクトリへのパスに作成します。

```
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 ブート イメージ

ブートファイルのビルドには、U-Boot、OP-TEE、Arm Trusted Firmware (TF-A)、TI-Linux-firmware が必要です。

TF-A のクローン作成と構築の手順:

付録 11.4、11.5、および 11.6 のパッチを TF-A に適用し、TF-A をビルドします。

OP-TEE のクローン作成と構築の手順:

U-Boot のクローン作成とビルドの手順:

ブートイメージ ファイルは、tiboot3.bin、tispl.bin、および u-boot.img です。ホスト コンピュータで、tiboot3.bin、tispl.bin、および u-boot.img ファイルを U-Boot から SD カードのブート パーティションに移動します。

```
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 コマンド

uEnv.txt ファイルを使用すると、U-Boot コマンドを毎回設定することなくセットアップできます。

このファイルは、ディスプレイが接続されているときに表示される自動コンソールを削除し、RPI DSI タッチ ディスプレイのオーバーレイを設定するために使用されます。

uEnv.txt ファイルの内容は [付録 11.8](#) に記載されています。uEnv.txt を SD カードのブート パーティションに移動します。

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

8.2.1.3 GUI ファイル

lvglsim ファイルは、LVGL グラフィックス アプリケーション用のバイナリです。

ti-lvgl-demo のクローンを作成してビルドする手順: <https://github.com/kwillis01/ti-lvgl-demo/blob/hmi-simple-demo/README.md>

まず、[ti-lvgl-demo](#) リポジトリをクローンし、hmi-simple-demo ブランチに切り替えます

```
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
```

この [チュートリアル](#) に従って Docker をセットアップします。LVGL デモをビルドします。

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

lvglsim バイナリを ti-lvgl-demo/lv_port_linux/bin/ から SD カードのルート パーティションのルート ディレクトリに移動します。

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

img_lv_demo_simple_high_res_ti_logo.png は [こちら](#) からダウンロードできます。

フォルダ内のディレクトリ ti-lvgl-demo/assets/ を作成します /media/\$USER/root/usr/share

img_lv_demo_simple_high_res_ti_logo.png を次に移動します /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 カーネル イメージ、モジュール、およびデバイス ツリー

Linux のクローン作成とビルドの [手順](#):

[付録 11.2](#) と [11.7](#) のパッチを Linux カーネルに適用し、カーネル イメージ ファイルとデバイス ツリー ファイルをビルドします。

k3-am6213-evm-hmi.dtb を /media/\$USER/root/boot/dtb/ti/k3-am6213-evm.dtb に移動し、k3-am6213-evm-dsi-rpi-7inch-panel.dtbo を次に移動します /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/
```

boot ディレクトリの SD ルート パーティションにカーネル イメージとモジュールをインストールします。

```
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 スクリプト

付録 11.3 に記載されている内容を wkup_periph_clk_freqs.sh という名前のファイルに、また付録 11.9 に記載されている内容を hmi-script.sh という名前のファイルにそれぞれ記述してください。

これらのスクリプトには、これまでの電力最適化のすべてが追加されています。また、lvglsim バイナリも開始します。スクリプトは作成するオプションですが、デモを実行するには、スクリプトに記載されているすべてのコマンドを実行する必要があります。

スクリプトを実行可能に設定します。

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

hmi-script.sh を次に移動します /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 デモを実行

1. すべてのファイルをビルドして SD カードにインストールしたら、ホスト コンピュータから取り出します。
2. SD カードを AM62L に挿入します。
3. リボン ケーブルを RPI タッチ ディスプレイに接続します。ポートはディスプレイの背面にあります。コネクタ ピンはディスプレイから離して向く必要があります。
4. AM62L の裏面で、リボン ケーブルを DSI_CONN ポートに接続します。コネクタ ピンは、ポートの白い部分に向いている必要があります。
5. ホスト コンピュータから、それぞれ AM62L の UART ポートと JTAG ポートに 2 本のマイクロ USB コードを接続します。
6. AM62L Power Visualizer ツールを開き、ページに従って「ツールのセットアップ方法」:<https://dev.ti.com/gallery/view/SitaraMPU/AM62L%5FPower%5FVisualizer/ver/1.0.0/>
7. ホスト コンピュータで、Linux UART コンソール用の/dev/ttyUSB0 を開きます。
8. AM62L の USB C ポートに電源を接続します。
9. Linux が起動したら、RPI ディスプレイの電源を入れます。

注

この操作は、起動時に遅すぎないようにしてください。遅れると、ディスプレイが認識されません。

10. AM62L Power Visualizer で電力測定値のストリーミングを開始します。
11. Linux が起動したら、himi-script.sh を実行します。ディスプレイに表示する必要がある内容の例については、[付録 11.1](#) を参照してください。
12. デモをシャットダウンするときは、次を実行します

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

9 まとめ

AM62L ベースの HMI アプリケーションの電力最適化は、HMI アプリケーションの概要や、その使用方法に大きく依存します。この資料では、HMI の使用事例の実践的なデモを活用して、AM62L SoC を利用する際に、グラフィックの最適化、デバイス ツリーの削減、クロック管理、メモリの最適化、ディスプレイの最適化など、システムの消費電力を低減するさまざまな方法を紹介します。HMI アプリケーションを使用している間に一時停止できるシステムについては、常時オンのディスプレイ低消費電力モードの詳細を確認できます。ユーザーは、与えられた情報に基づいて、システムに最適な電力最適化を決定し、実装することができます。

10 参考資料

1. GitHub、<https://github.com/lvgl/lvgl>、Web ページ。
2. テキサス インスツルメンツ、[AM335x 低消費電力設計ガイド](#)、アプリケーション ノート。
3. GitHub、<https://git.ti.com/cgit/k3conf/k3conf/>、Web ページ。
4. テキサス インスツルメンツ、[AM62L テクニカル リファレンス マニュアル](#)、Web ページ。
5. テキサス インスツルメンツ、[AM62L データシート](#)、Web ページ。
6. テキサス インスツルメンツ、[AM62L の消費電力の概要](#)、Web ページ。
7. テキサス インスツルメンツ、[AM62L Power Visualizer](#)、ウェブページ。

11 付録

11.1 画像を表示

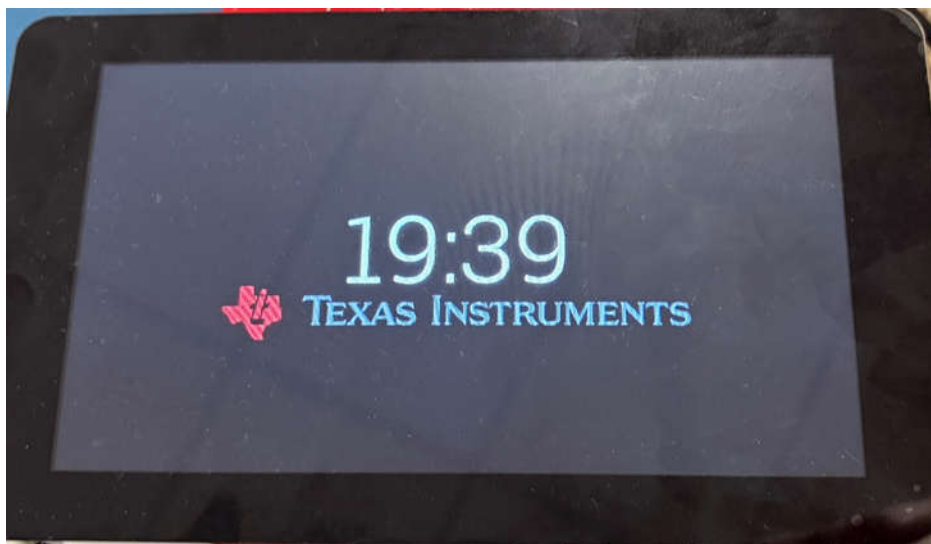


図 11-1. 画像を表示

11.2 最小デバイス ツリー パッチ

Linux カーネルにパッチを適用します。

```
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_SD_CD */
+         >;
+         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 低クロック周波数

カーネルで実行できる Bash スクリプト。

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 速度を 800 MT/s に下げるパッチ

TF-A にパッチを適用します

```

diff --git a/fdts/am62l-skevm-lp4-50-800.dtsi b/fdts/am62l-skevm-lp4-50-800.dtsi
index a498030c5..9daba8d6b 100644
--- a/fdts/am62l-skevm-lp4-50-800.dtsi
+++ b/fdts/am62l-skevm-lp4-50-800.dtsi
@@ -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 DDR のオート セルフ リフレッシュを有効にする

TF-A にパッチを適用します。

```
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 自動クロック ゲーティングを有効にする

TF-A にパッチを適用します。

```
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_scm_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_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_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 ハーフ パッチによる低速パネル ピクセル クロック

Linux カーネルにパッチを適用します。

```
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 コマンド

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 すべての電力最適化を有効にして、lvglsim バイナリを実行する

Bash スクリプト。

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
./lvglsim
```

11.10 RPI タッチ ディスプレイのトラブルシューティング

Linux が RPI タッチ スクリーンを検出できない場合は、AM62L および RPI タッチ ディスプレイの電源を切ってください。[デモを実行]の説明に従って、電源をバックアップします。

同じ問題が発生した場合は、もう一度やり直してください。

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含みいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとし、TI は一切の責任を拒否します。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、[TI の販売条件](#)、[TI の総合的な品質ガイドライン](#)、[ti.com](#) または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

Copyright © 2026, Texas Instruments Incorporated

最終更新日：2025 年 10 月