# Application Note Early Splash Screen With Flicker-Free Transition



Aparna Patra, Devarsh Thakkar, Nikhil Jain, Soumya Tripathy

Sitara MPU

#### ABSTRACT

This application note provides information on how to display a splash screen as fast as possible on AM62P and perform a glitch-free transition from the splash screen to the full system user interface (UI). The steps in this document are demonstrated using the Universal Boot Loader (U-Boot) flow as well as custom Slim Bootloader (SBL) along with the Linux<sup>®</sup> Kernel.

## **Table of Contents**

1 Introduction	2
2 Hardware Used	2
2.1 AM62Px Processor	2
2.2 SK-LCD1	3
2.3 Display Subsystem on AM62P	3
3 Early Splash-Screen Architecture	3
3.1 Boot Stages on AM62P	3
3.2 Flicker-Free Transition	4
4 Flicker-Free Transition From SPL to U-Boot	4
4.1 Steps to Test	4
4.2 Measurements	5
5 Flicker-Free Transition From SBL to Linux <sup>®</sup> Kernel	6
5.1 Steps to Test	6
5.2 Measurements	11
6 Results	12

### **List of Figures**

Figure 2-1. AM62P Block Diagram	2
Figure 2-2. DSS Block Diagram	3
Figure 3-1. U-Boot	3
Figure 4-1. Splash Flow	4
Figure 5-1. Splash Timings	6
Figure 5-2. Code Changes 1	8
Figure 5-3. Code Changes 2	. 9
Figure 5-4. Code Changes 3	10
Figure 5-5. SysConfig Settings	11
Figure 6-1. Hardware Setup	13

### List of Tables

Table 6-1. Spla	ash Display	Time Comparison	. 12
-----------------	-------------	-----------------	------

#### Trademarks

Sitara<sup>™</sup> is a trademark of Texas Instruments. Linux<sup>®</sup> is a registered trademark of Linus Torvalds. FreeRTOS<sup>®</sup> is a registered trademark of Amazon Technologies, Inc. Arm<sup>®</sup> and Cortex<sup>®</sup> are registered trademarks of Arm Limited. All trademarks are the property of their respective owners.



## **1** Introduction

Increasing numbers of automotive, industrial, and robotic use-cases require enabling the display as early as possible in the boot cycle. The operating systems involve multiple stages for boot up and a bootloader is the first software component to come up and initialize the system before booting up to the full system. This application report explains how to enable the display at the bootloader stage, display a splash-screen or animation at this stage, and have a flicker-free transition to the system UI. The splash screen enablement is explained for U-Boot which is a widely used open-source bootloader, as well as a Secondary Bootloader (FreeRTOS®-based) which can perform a fast boot up to Linux using the Falcon mode boot flow. This guide then explains how the splash screen display context can be preserved while the system is booting up and a flicker-free transition to the system UI can be achieved.

## 2 Hardware Used

#### 2.1 AM62Px Processor

The AM62Px (P = Plus) is an extension of the existing Sitara<sup>™</sup> AM62x low-cost family of application processors built for high-performance embedded 3D display applications. Scalable Arm<sup>®</sup> Cortex<sup>®</sup>-A53 performance and embedded features, such as: multiscreen high-definition display support, 3D-graphics acceleration, 4K video acceleration, and extensive peripherals make the AM62Px well-designed for a broad range of automotive and industrial applications, including automotive digital instrumentation, automotive displays, industrial Human Machine Interfaces (HMI), and more.



### Figure 2-1. AM62P Block Diagram



## 2.2 SK-LCD1

The 1920 × 1200 Open LVDS Display Interface (OLDI) display or Low Voltage Differential Signaling (LVDS) Liquid Crystal Display (LCD) kit is an add-on accessory for the starter kit AM62x processor Evaluation Modules (EVM) to add touch and display functions for the evaluation of HMI, industrial PC, and other use cases requiring display. This model is composed of a Thin Film Transistor (TFT) LCD panel, a driving circuit, a backlight system and a projected capacitive touch panel.

#### 2.3 Display Subsystem on AM62P

The Display Subsystem (DSS) is a flexible, multipipeline subsystem that supports high-resolution display outputs. DSS includes input pipelines providing multilayerblending with transparency to enable on-the-fly composition. Various pixel processing capabilities are supported, such as color space conversion and scaling, among others. DSS includes a Direct Memory Access (DMA) engine, which allows direct access to the framebuffer (device system memory). Display outputs can connect seamlessly to an Open LVDS Display Interface transmitter (OLDITX), or can directly drive device pads as a Display Parallel Interface (DPI).



Figure 2-2. DSS Block Diagram

## 3 Early Splash-Screen Architecture

#### 3.1 Boot Stages on AM62P

This section describes the two boot flow sequences used on AM62P. The Read Only Memory (ROM) code is the first block of code that is automatically run on device start-up or after power-on reset (POR). The ROM bootloader code is hardcoded into the device. This is a very small binary, due to the limited amount of internal memory. After the ROM bootloader comes up, the bootloader either loads the *Secondary Program* loader for a Linux-specific boot flow, or the *Secondary Bootloader* for Real-Time Operating System (RTOS) based boot flow. These boot sequences can also be customized according to the required use case.





 Secondary Program loader mainly serves to initialize the external Double Data Rate (DDR) memory and sets up the boot process for the next bootloader stage, U-Boot. U-Boot, running out of DDR, provides wider functionalities, such as command (CMD) line support, device driver infrastructure and Kconfig infrastructure. U-Boot then loads the Kernel image, to start a High-Level Operating System (HLOS) like Linux on a A53 core.



2. Secondary Bootloader is a FreeRTOS based bootloader, responsible for performing device-specific initialization, loading of respective binaries to initialize the subsequent cores, and to ultimately start the application. On AM62P, the Secondary Bootloader is divided into two stages, namely stage1 and stage 2. Stage 1 initializes DDR and loads the SBL stage 2 and the device manager binary to DDR. The SBL stage 2 has two threads running in parallel, one SBL stage 2 thread to run the boot sequence to start the Hardware Security Module (HSM) M4 core, MCU-R5 core, and Linux on the A53 core, another thread loads the device manager that opens the drivers required for application.

#### 3.2 Flicker-Free Transition

To achieve flicker-free transition from U-Boot Secondary Program Loader (SPL) to the U-Boot stage, preserve the image framebuffer and do not close the Display Subsystem (DSS) driver. To pass the framebuffer from the SPL stage to the U-Boot proper stage, a region in the memory is reserved and the same region is passed from the SPL stage to U-Boot proper using the bloblist, in the video\_post\_probe function. At the SPL stage, various parameters like framebuffer region, size, number of pixel columns (xsize), and the number of pixel rows (ysize) are all stored in a blob. The blobs are reserved memory regions which contain information to be passed from one stage to another. When the initial setup sequence is running in the U-Boot proper stage, reserve\_video API is called, which determines if video blob is present. If video blob is present, the Application Programming Interface (API) uses blob data from the previous stage which makes sure the same framebuffer region and parameters are set, thus keeping the splash image intact without any flicker across the stage. If blob is found, the DSS driver is not probed again, which prevents screen refresh.

## 4 Flicker-Free Transition From SPL to U-Boot

#### 4.1 Steps to Test

This section describes the steps required to achieve an early splash screen, along with flicker-free transition from the SPL stage to U-Boot. The splash screen support from A53 SPL is provided out-of-the-box in the Linux version 9.0 Software Development Kit (SDK) onwards. By default, the splash screen is only enabled at A53 SPL. The default splash source was set to an SD card and displays a gzip TI logo .bmp image. The SPL splash screen features are compiled in the tispl.bin which is built during U-Boot compilation. Any changes made to the SPL splash screen feature requires recompiling tispl.bin. Use the new tispl.bin to boot the board to see the splash screen at the SPL stage.



Figure 4-1. Splash Flow

All information related to splash screens is stored in a board.env file, board/ti/am62px.env contains all the splash-related variables for the AM62P device:

1. Instructions for displaying a custom logo:

To display a custom logo update the splash file with the file name of the custom logo. In a case where the AM62P splash source is MMC (this refers to SD card boot media) to replace the image copy the new splash image in the boot partition of the SD card.

#### Note

U-Boot only supports .bmp and compressed .bmp images.

splashfile=ti\_logo\_414x97\_32bpp.bmp.gz
splashimage=0x80200000
splashpos=m,m
splashsource=mmc



2. Instructions for changing boot media:

The splash\_source struct defined in board/ti/am62px/evm.c defines the different sources from which the splash image can be displayed.

To change the splash source, update the splashsource variable in board.env file with the variable name of the sources defined in the default\_splash\_locations struct. AM62P supports two boot medias; "sf" refers to Octal Serial Peripheral Interface (OSPI) and "mmc" refers to an SD card. Use one of the boot media from the sources defined. To use a different boot media, add the information in the struct and update the splashsource with the name of the new boot media.

#### 4.2 Measurements

The time taken from POR to when the display comes up is noted to be approximately 680ms. The General Purpose Input-Output (GPIO) pin settings, by default, are in Off state. The direction and value of this GPIO pin is set to high, just after the video\_bmp\_display() function, which displays a .bmp file onto the panel. OSPI NOR is used as boot-media for testing purposes.

• GPIO0\_39 on the user expansion connector-pin 18, is used for time measurements. To set the direction of this pin to output, and the state of this pin to high, use the following code:

```
- a/common/bmp.c
+++ b/common/bmp.c
@@ -19,6 +19,7 @@
#include <splash.h>
#include <splash.h>
#include <asm/byteorder.h>
+#include <asm/byteorder.h>
+#include <asm/io.h>
/*
Allocate and decompress a BMP image using gunzip().
@@ -142,6 +143,11 @@ int bmp_display(ulong addr, int x, int y)
ret = video_bmp_display(dev, addr, x, y, align);
}
+ writel(0x00050007, 0x000F40A0);
+ writel(0xFFFFF7F, 0x00600038);
+ writel(0x80, 0x00600040);
+
if (bmp_alloc_addr)
free(bmp_alloc_addr);
```

 To enable the GPIO driver, if not done already, use the following config options in the am62px\_evm\_r5\_defconfig file:

+CONFIG\_SPL\_GPIO=y +CONFIG\_GPIO=y +CONFIG\_DM\_GPIO=y +CONFIG\_DA&XX\_GPIO=y +CONFIG\_CMD\_GPIO=y

To set the pin multiplexer (pinmux) of the GPIO pin number GPIO0\_39, use the device tree pinmux settings
as mentioned in the following code:

```
diff --git a/arch/arm64/boot/dts/ti/k3-am62p5-sk.dts b/arch/arm64/boot/dts/ti/k3-am62p5-sk.dts
index 4b8e7964ca4d..7dbf5e9b9c2b 100644

    a/arch/arm64/boot/dts/ti/k3-am62p5-sk.dts

+++ b/arch/arm64/boot/dts/ti/k3-am62p5-sk.dts
@@ -232,6 +232,10 @@ hdmi_connector_in: endpoint {
 &main_gpio0
     bootph-all;
+
     status = "okay";
pinctrl-names = "default";
+
+
+
     pinctrl-0 = <&test_gpio_default>;
 }:
 &main_gpio1 {
@@ -446,6 +450,12 @@ AM62PX_IOPAD(0x0078, PIN_OUTPUT, 1) /* (AC24) GPMC0_AD15.VOUT0_DATA23 */
             AM62PX_IOPAD(0x009c, PIN_OUTPUT, 1) /* (AD24) GPMC0_WAIT1.VOUT0_EXTPCLKIN */
         >:
     };
+
+
     test_gpio_default: test-gpio {
+
         pinctrl-single, pins = <
             AM62PX_IOPAD(0x00a0, PIN_INPUT, 7) /* (P24) GPMC0_WPn.GPI00_39 */
+
+
         >:
+
     };
 };
&main_i2c0 {
@@ -789,6 +799,7 @@ &mcu_r5fss0_core0 {
 &main_uart0 {
     pinctrl-names = "default"
     pinctrl-0 = <&main_uart0_pins_default>;
test-gpios = <&main_gpio0 39 GPI0_ACTIVE_HIGH>;
+
     interrupts-extended = <&gic500 GIC_SPI 178 IRQ_TYPE_LEVEL_HIGH>,
```

 Next, connect the GPIO pin and MCU\_PORz to the logic analyzer and measure the time difference between the two to get the accurate timestamp.

### 5 Flicker-Free Transition From SBL to Linux<sup>®</sup> Kernel

#### 5.1 Steps to Test

This section describes the steps required to achieve an early splash screen, along with flicker-free transition from the SBL stage to the Linux Kernel. The DSS sharing example in the MCU+ SDK integrates early splash of image along with SBL on the OSPI boot media, Device Manager, and Inter-processor communication functionality. The bootloader, IPC, and Display run on separate tasks. The Display task displays a splash image with alpha blending and finally switches to the display sharing task, where telltale frames quickly move back and forth. Falcon boot is used in the example, which means the intermediate U-Boot stage is skipped and SBL directly boots the Linux image. This DSS example, with a few modifications is used for demonstration.





- Download the processor SDK and use the ti-linux-kernel directory under /board-support to make the modifications as described in the following steps. Generate the Device Tree Blob (DTB) and image file after the modifications to ti-linux-kernel. These files are later used to create the linux.appimage, which is used in the RTOS example to run Linux on an A53 core.
- The linux.appimage is built using Falcon boot mode. Hence, include the bootargs information in the k3-am62p5-sk.dts file under the chosen node:

bootargs = "console=ttyS2,115200n8 earlycon=ns16550a,mmio32,0x02800000 root=/dev/mmcblk1p2 rw
rootfstype=ext4 rootwait";



 To make sure that the splash screen remains persistent while the Linux Kernel boots up, ti-u-boot dynamically updates the Linux Kernel device-tree with framebuffer region meta-data, marking the region as reserved in the Linux device tree as follows:

```
framebuffer: framebuffer@ff700000 {
    reg = <0x00 0xff700000 0x00 0x008ca000>;
    no-map;
};
```

• Set the status of the simple-framebuffer node to "okay" by manually modifying the board device-tree file under the chosen node as shown in the following code:

- To keep the boot animation alive until the display server starts up, the *Direct Rendering Manager (DRM)* framebuffer device emulation feature needs to be manually disabled by removing the following config option in arch/arm64/configs/defconfig like mentioned below: # CONFIG\_DRM\_FBDEV\_EMULATION is not set
- After the previously shown changes in Linux directory, build the Linux Kernel, to create the DTB and Kernel image.
- Apply the overlay file k3-am62p5-sk-microtips-mf101hie-panel.dtbo onto the DTB file to support display on an OLDI panel. Use the following command:

```
fdtoverlay -i ./arch/arm64/boot/dts/ti/k3-am62p5-sk.dtb ./arch/arm64/boot/dts/ti/k3-am62p5-
sk-microtips-mf101hie-panel.dtbo -o ./../../board-support/prebuilt-images/am62pxx-evm-display-
cluster/k3-am62p.dtb
```

- Copy the following two files into the /board-support/prebuilt-images/am62pxx-evm-displaycluster folder:
  - 1. k3-am62p.dtb (created in the previous step)
  - 2. Image (arch/arm64/boot/)
- Remove the usage and definition of DispApp\_splashThread() and DispApp\_displayShareThread() in the examples/drivers/dss/dss\_display\_share/dss\_display\_share.c file



Digioner (Carl+Shitt+			All Design and a second s	
O OPDITUDIOIS	in the subscription of the	Cumples 111	2 BYMES J AGG / BGS (BASER) SHEE / K. KO DORARI SHEE (BASER) SHEE (	
C do the	RZ harge/divection depty care by		Road Links ( Links and Links a	Des_Isside/cells
20 × C #11,55we	ex - as dated shared exercicities to		<pre>stic trict triangl_parterment(bits from the sector sector); stic trict outs of under initiating hereat these before the sector); stic trict outs of under initiating hereat these before the sector);</pre>	<pre>stors: unit tuised_picture/metric/metric/metric/stors/ tuised_store/ tuised_st</pre>
C Malloc or			ietic vedd Displey updetelpiant/ramefarfer(vedd +framefarf, sivi32_t xpostion,	staric vois Stopley updatesimationation (vois "framedur, vietz_t apostine,
rs 6 att, clipte			wints_t yeolition, wints_t winth,	udetsz_t paposition, udetsz_t winte,
E. NO NO SOKA	MN27X,25,01,00,25		stic void Display updetfelltalerreenffreviewing (until t postion,	usta_t engre usta_t teger, usta_t eger,
62 V 84 (94 10	In a matching			
> 501,0001	Uirun, staget		uint32_t beight, uint32_t beight, uint32_t be);	udin12_t height, uin12_t beight, uin12_t bepij;
<ul> <li>stillestill</li> </ul>			ant: void unspegi-splant interest www.close_orget most, void marget; rele void Unspegi-splant interest i	there was accepted plant amount on vision _coper *eep, vos *ergs; there was accepted plant amount on vision _coper *eep, vos *ergs;
> Serric				
E esample	Heavyschig			
e manc			CURAL ACCOUNTS 77	
3 90 90 000				
> still set i i				* Ginais ta Null Fis the */
> columnum				source growthether, partmenticing, macount
× dis:				
> dis_colorb	tarjet			
- dis_display	el trace el			
C An Ant	New concrete h		int_t gfirstPipelineFrameNuffCONTE_D5_NEW_FAMPS_PER_PIPEINE((D159_FRAME_STRL_WX + D159_FRAME_STRL_UD50_UWXattribute((Lection (*.stsframeNuffer*), alignet (4044))	winit girtsipeliedrameluf(CONTECENS HAR FRANKS FAR FIREDELINDE)(DIPLOWE_STRE_ANN + NOP_FONE_STRE_ANN + NOP_FONE_STRE_ANN + NOP_FONE_STRE_ANN + NOP_FONE_STREEANN + NOP_FONE
e andipi	day draws to			
C decision				<pre>- public (tribut end output) South Virtual State (1,1); tatts: uitht; []; []; []; []; []; []; []; []; []; []</pre>
> dis_dista				
> dis_safety				
> gpia			static Taxa, object glisolassilastiassi	tell Text of the pipersolettes:
				NUCLE TOOM OF A DISPOSING SHARES AND
1 1000				
> mass			Static Sampbers_Coject glplashimeoutiem;	that is feepbore (bjoct galantiteouties)
✓ mcapi				
<ul> <li>mcspi_loop</li> </ul>				(a) Construction (Construction) (
<ul> <li>amégas a</li> </ul>				
> nca.6	shoul of seedos			
) mea-dd	statute room			
)	n-Garg	164		
5 many	pikoyadg			
C mains		166 4		
C mop()or				
> mopigen c	sformance_001			
> most per	stometice_1200		dispap_in(thranes())	D(shop_in(0*rees())
2 000			Discher init(AdDaObjects(CONTO DSST)):	Disalge init(adaptorets(CMP20 0000));
> safety				
> stident				
> uet			retVel = Display_runlet(&gbsubbjects(CONFD6_DSS0));	
> udma				
> empty			Bisphpp_deTrit(AgDcodeSects(ComFIG_D650)))	
- Indian de				
> mea-choi	x3-0 tractics			
> meu-choo	s0-0_nortos			
< wkup-chis				
> Sam ci				
E etarpit	A systdig			
C India and				
> help world o				
> kerrel		190 1		
> atp				displop_slashifteed(vaid *args)
2.50				
> security				be (Dist the local + (Dist (Dist the local + )
v tool				
> 6/2/2				Public Francist Frantist
> bin2teader				
~ boot				
> HSMAppins				intil_t retnil = retni = retni = retni = retni
> linux/ppins	naget ven			
y authrees				Semphord"_Bend(List(Doj))ywc2em, System/JW11_(DoctV3);
> qralippina				
✓ sbi_prebuilt				
O default sol	of energina to footp	200-		Life and the second sec
0 detaut do	a nu ni ni nica hi han			PV002_TPREAT_000();
0 0000000	bi, ospi, Jimar, hs. fs. spikeh, soveen da			
O default sta				
0 00001,00				
O defauit abl				Display_seditisibili reduction (sediment sediment sedim
O default all	direct work high clo			
E GLASSE	c inux chari debus Sirrace			
E CLANKS	Cinuc dage Lelease to Actinage	215-		/* Queue the rest frame puffer // indextda. Results. 801-
E sid errore				s((VVDL_dok i= rettia))
E staljevil.de				
E dipute	ebaştimaşe	223-	because long, statute queue feiled))(v/v/);	best to the set of the
E dijulai	Conta Distance			
E SE ASH	ing Secletatis Minar			
E 50 050 10	Inversitage1 debug timage	223-		actor of Change Contraction of Contraction
F idi, and Ji				
F stijasajje				
E SUSTIN	undebug/is/Salirage			
# 59(34)ha	nandetnig formage			
E stal yd jing	a releve firmer			
F 10 cart in	Inux debug ha hutimage	256		
(in Folgerie				
e columnation				Tail@_setwork(Editionardelabilitation)
222 > OUTLINE				
NULTRACE STREET				
10 294 A 1				

Figure 5-2. Code Changes 1





Figure 5-3. Code Changes 2



~ OPEN EDITORS						
C dog_dopiny_shares_examples/dover/doubles_do_ 3+						
X C the share a desident share accordents. As						
C mains exercisioneri doi doo dooler aheriaria 7+	001 Dis static inti2 t Disate putter(Des Delert terret)	202 203 static (st32 t Disclos context(Das Drist tangBtd)				
C the theolog shore completibline deploy the	no {	394 (				
10 <sup>2</sup> - MCU PLUS 20K AM 20 PK (09 (01 (00 30))	107 int32_t retVal = fVID2_50K;	395 int32_t retVal = FVT00_50K;				
> se,nui	10 int32,t thutus = System <sup>2</sup> _SUCCES; Does nutbeinet System <sup>2</sup> _SUCCES;	306 inf13_t status = System_JAKIESS; 307 Des Institutet = Hintelfait				
E v stol ospi jinus mutatage	40 W Tasit Forms taktheres;	3 30+ Task_From taskforms;				
<ul> <li>Sociospipinius stager</li> <li>Sociospipinius stager</li> <li>Sociospipinius stager</li> <li>Sociospipinius stager</li> </ul>						
> ti-am-date						
E eargiesystig	64 Disket (arCbi);	42 Diake create (code):				
C main.c						
> stil_sd_linux	66 Debug log("Starting display (IVrW")) Debug log("Starting display (IVrW"))	(b) Debug log(Starting display Hittory); Debug log(Starting display Hittory);				
> sbjuert						
> social units and the second states						
v da		401+ Task? Parama, Init(RiaskParamas);				
> dss.colorbar.test		4104 task/arean.stackSize = DISPLAY TASK SIZE:				
<ul> <li>dis_display_plane</li> </ul>						
> anticipa de la	10- Wrasservers. priority - TASC HELSPILSH, THERADI, Manual Annual Annual Annual Annual Annua Annual Annual Annua Annual Annual Annu	412+ taskParama.priority = TASK PHI_SPLASH_TMERAD;				
C dss_display_sample.h	The advances and a construction of the second	41/4 taidymm.astherin - Blacker missification				
C os distay granes 34						
C designation of the		416+ status = TaskP_construct(AgBisplaySplashTask, AtaskParans);				
7 035_03049_IB1		9.7 A 4124 status as Searchered constructionary behavior from 201				
> 0010		49				
> 10c						
> loc						
> ncan						
> recep						
V hogi	A CHECKEN AND A CHECKEN A WAY SHALL THE CHECKEN	Control Control and Control an				
v mogu opporte	49- McClcofferenc, callect + Nisplag_splastificouttering	4274 clackPersons.collbeck = 0159400_splassTimeoutClfvn;				
> mcu rifetiõ ferritos						
> mourstade on notes		<pre>status = ClockP_construct(#giplaihClockDtj, &amp;clockParems);</pre>				
<ul> <li>wbup r5fss3.0, freetos</li> </ul>	If (status as Sustant SUSS)	All Britstatus an Water Wittens				
> ti-am-clang						
F example.sysclg						
C mainc	Dispuss_results/pinore results/fer(costs*)kg/srstPipe/inef(art(s), 0155, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 562, 581, 581, 581, 581, 581, 581, 581, 581	c.s. Usgade_upsit/sepiad/read/ar/free/(uside/jag/inset/sepiad/read/uside/sepiad sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sepiad/sep				
C most loopbedu	DISP_STAND_UNKD_VELOWT, DISP_STREE_PROL_PICE();	456 005 group (stable) (sta				
> mospi_performance_(bit						
> morp (performance ) sont	20 Display_spidteSplanhfreeNoffer(void*)&gfirstPipelineFreeNof(1), DISP_SPLAS(_DMAD_DDOSTION, \	438 DispApp_updetr5plashfreeSeffer((void/MgfirstPipelinefreeSeff)), DISP_SPLAS_INADE_SPDSTEDN, \				
> minop	41 0159 SPLAS_DWAL_WOTTON, 0159 SPLASE_DWAL_QUDN, V 20 DTFS 9013051 MAGE METAL. DTS MEDITIS ME METALIN	4/7 OSST-20104-UNAS-HUDOL (USE STUDIE) (USE STUDIE) ( 6/9 OSST-20104-UNAS-HUDOL (				
> safety						
> scicilent	iii inst0sj = &app0sj->inst0sj[0];	442 instObj = RappObj-sinstObj(0);				
> uet						
> udma	Ac // Serie tradgety devices // retwal = fold strengt (institution of strangt), multiple	www. // start clapped praver //				
> empty						
✓ helo_world	iii) if(retival := IVID2_SOK)	447 if(retVal. I= FVID2_SOK)				
✓ amtGpo-sk						
> mou-cites0-0_freetos						
> mou-chan-dynamos						
<ul> <li>witcp-rotation greenes</li> </ul>						
E correlegada	55 // Schr Spiller (Red) Calcular //	Control of the C				
C maine						
C helo,world.c						
> helo_world_cop	10- Composition of Company of	AS74 Semphoref_perd(appliablimeoutiem_system/gAIT_ORIVER);     Emphoref_perd(appliablimeoutiem_set);     Emphoref_perd(appliablimeoutiem);     Emphoref_perd(appliablimeoutiem);     Emphoref_perd(appliablimeoutiem);     Emphoref_perd(appliablimeoutiem);     Emphoref_perd(appliablimeoutiem);     Emphoref_perd(appliablimeoutiem);     Emphoref_perd(appliablimeoutiem, system);     Emp				
> kernel						
> otp						
> ad	64 //// Semptore/.perof81st00j:rsynSem, System/.pdff_F0895K); /// USAdd arteroid and an an and an and an an and an and an	61.4 Sessionerf_peed(ddirstop)-souncies_stytem_salT_f0REvEs); (62) Tavid Antrover(ddirduodsolandirstop)				
) 20070		40 b				
v tools						
> bin2c		→ cos in(retVal == PVIDL_SOR)				
> bin2header		467+ // Create display starse threas */				
~ boot						
> HSM3ppimageGen	taskbarami name - TBispipp displayShareThread";	Construction and a "Display display there thread";				
> InuxAppinageGen	// task@arms.stark = gg(abs/shortEasStarks(0))	Contract and a second and				
> cu/2rprc						
> gnu#ppimageGen	testbares args = (0.147) (0.0715) (0.0715) (0.050)     testbares for (0.147) (0.050) (0.0116) (0.050)	27.54 ExistManama, ang) = (volum lagosobjects)(CHP 10_00001); 27.54 ExistManama, ang) = (volum lagosobjects)(CHP 10_00001);				
v sbi_prebuilt\art62ps-sk		65				
0 default_sbl_ennic_linar_hs_fs.clg		ATC- status = Taxidconstruct(kgDisplayShareTaxk, AtaxidFarams);				
O default_std_nult_in_fuztg		AV7				
O delaut_sol_copi_inus_talcon_ins_ts.ctp		2794 (ALCE) (ALCE)				
O defeut stijoppijinus ha huspieh screen.dg						
O default set osoi one hs facto						
0 defect_bb_uet_hi_b.dp						
0. default still wart, linux his fischy						
Stolenme_Inue_stage1.debug.ht_httimage						
E stijenne, inwystage Ldebug timage		+ 407+ Stifstatus (+ System SUCCES)				
E stijening inzujstaget release to jis timage						
# stojenimi, inuo stage trelesie timage El chi o il debuo te de timo a		Carde_Sentruct(AsplantCardod);				
E Stindigtuntinge	Table destruct (definition) interval	2011 Tast destruct (Antionalanda)				
E stijnulizeosetis Stimage		402+ TastP_destruct(&g0isplayShereTask);				
E sbi_ru/intege						
E strjospijinur, stage Ldebug to jis Limage						
E stij ospij linuk staget debugstimage						
E stij ospijinus staget release his fistimage						
# sbi_ospi_linux_stage1 release.timage	Not a static set for the second for	400 control on the Newton Intelligent American American Control (Control (Contro) (Control (C				
E sti st ina detautimate		50 ( Sector Construction of the Construction o				
E shi si fina menerin fi timare	00 0cs.0ctrivideness huteres:	501 Oss Octr10starass *sptarass;				
E sti st inucreiese finance	DA DELOCICIÓN/MARAMENTES	502 Day, Detro Ad-Vighersens * ad-Vighersens :				
E stijuert inus debug in tstimage		100 USA UNITER AND A UNITE				
F stijuert_inuxdebugstimage	N7 Des DetriblichelDerem vglobelDerferen;	185 Des DitriflioselDesPerent "globalDesPerent;				
If stijuart_inuurelesselts_fstimage	Verfanses = Reverfel-instances	500 507 vitheraus = Asseth's tvitheraus -				
) omaa Sile overlayhuus; soodi soosaa soodi soo soo soo soo soo soo soo soo soo so						
TBRELINE	200     > taketike       510     LayerParans + Asp00(s)-SlayerParans;					
N 03429						

Figure 5-4. Code Changes 3

- Compile the images in tools/boot/sbl\_prebuilt/am62px-sk/ default\_sbl\_ospi\_linux\_hs\_fs\_splash\_screen.cfg
- Flash the built images into the OSPI flash using the universal asynchronous receiver-transmitter (UART) TI UniFlash tool.
- For fast rendering of the Linux GUI application, build and install ti-img-rogue-driver using the top-level makefile in SDK. Use the following steps:
  - 1. Run command:

٠

- make ti-img-rogue-driver (Value RGX\_BVNC="36.53.104.796" in Rules.make)
  2. Insert the SD card and run the following command:
- sudo make ti-img-rogue-driver\_install DESTDIR=/media/aparna/root Switch to OSPI NOR boot mode to view the demonstration.

BOOTMODE [ 8 : 15 ] (Sw5) = 0000 0000 BOOTMODE [ 0 : 7 ] (Sw4) = 1100 1110



### **5.2 Measurements**

The time taken from POR to when the display comes up is noted to be approximately 180ms. The measurement was taken by using a GPIO pin, which by default is in active-low state. The pin is set high in CSL\_dssVpSetGoBit() by using the gpio\_set\_high() API. OSPI NOR is used as boot-media for testing purposes.

- The dss\_display share example provided in the default MCU+ SDK is used for experimentation. Modifications to this example are made to implement the flicker-free transition demonstrated through SBL flow.
- GPIO0\_39 on the user expansion connector-pin 18, is used for time measurements. Figure 5-5 shows the options in SysConfig to enable this pin:

LE ABOUT						RE	STAF
\Xi Type Filter Text 🗙 🕊	$\leftrightarrow$ $\rightarrow$ Software $ ightarrow$ GPIO					(i) <> 👜 🤇	Ð
V TI DRIVER PORTING LAYER RAT	GPIO (1 Added)	Œ	ADD	i Problems		×2	× *
Clock 1/1 🔮 🕀	GPIO_LED			There are no problems in the co	urrent design.		
MPU ARMv7 12/16 ♥ ⊕ TIMER ⊕	Name PIN Direction	GPIO_LED Output	*	♦ Generated Files		2	* ×
TI DRIVERS (16)	Trigger Type	None	*	Filter: all			-
BOOTLOADER 3 🥑 🕀	Use MCU Domain Peripherals			File name	Category	Include in build	
DSS 1/1 🛇 🕀 EPWM 🕀	GPIO Peripheral Preferred Voltage	GPIO0 Any	▼ 🗗	ti_dpl_config.c	TI Driver Porting Layer (DPL)	-	
GPIO 1 ♥ GTC 1/1 ♥ GTC 1/1 ♥ ⊕	✓ Signals ↑↓	Pins	Pull Up/Down Rx Pull Up 👻 🔽	ti_dpl_config.h	TI Driver Porting Layer (DPL)	-	ĺ
	GPIO Pin(GPI00_39)	GPMC0_WP 👻	🗈 No Pull 🔻 🗹	ti_drivers_config.c	TI Drivers	-	í
MCAN $\oplus$	Other Dependencies		~	ti_drivers_config.h	TI Drivers	-	
MCASP   MCSPI				ti_drivers_open_close.c	TI Drivers	-	
MCU_BIST				ti_drivers_open_close.h	TI Drivers	-	
OSPI 1/1 ♥ ⊕				ti_pinmux_config.c	TI Drivers	-	
UART 2 ♥ ⊕ UDMA 1/2 ♥ ⊕				ti_power_clock_config.c	TI Drivers	-	
TI BOARD DRIVERS (2)				ti_board_config.c	TI Board Drivers	-	
LED ()				ti_board_config.h	TI Board Drivers	-	
FILE SYSTEM (1)				ti_board_open_close.c	TI Board Drivers	-	
				ti_board_open_close.h	TI Board Drivers	-	
				example.syscfg	Configuration Script		
				13 Total Files			

#### Figure 5-5. SysConfig Settings

Set the GPIO pin to high by defining the API in dss\_display\_share.c as follows :

```
void gpio_set_high(void *args) {
    uint32_t gpioBaseAddr, pinNum;
    DebugP_log("GPIO LED started ...\r\n");
    /* Get address after translation translate */
    gpioBaseAddr = (uint32_t) AddrTranslateP_getLocalAddr(GPIO_LED_BASE_ADDR); pinNum =
    GPIO_LED_PIN;
    GPIO_setDirMode(gpioBaseAddr, pinNum, GPIO_LED_DIR);
    GPIO_pinWriteHigh(gpioBaseAddr, pinNum);
    DebugP_log("GPIO LED HIGH!!\r\n");
}
```



Use the API in CSL\_dssVpSetGoBit() as follows: diff --git a/source/drivers/dss/v0/hw\_include/V3/csl\_dssVideoPort.c b/source/drivers/dss/v0/ hw\_include/V3/csl\_dssVideoPort.c index f882d54..ee18b95 100755 --- a/source/drivers/dss/v0/hw\_include/V3/csl\_dssVideoPort.c +++ b/source/drivers/dss/v0/hw\_include/V3/csl\_dssVideoPort.c @@ -183,6 +183,8 @@ void CSL\_dssVpEnable(CSL\_dss\_vpRegs \*vpRegs, uint32\_t enable) CSL\_REG32\_WR(&vpRegs->CONTROL, regVal); } +extern void gpio\_set\_high(void \*args); + void CSL\_dssVpSetGoBit(CSL\_dss\_vpRegs \*vpRegs) { uint32\_t regVal; @@ -192,6 +194,7 @@ void CSL\_dssVpSetGoBit(CSL\_dss\_vpRegs \*vpRegs) DSS\_VP1\_CONTROL\_GOBIT, CSL\_DSS\_VP1\_CONTROL\_GOBIT\_VAL\_UFPSR); CSL\_REG32\_WR(&vpRegs->CONTROL, regVal); + gpio\_set\_high(NULL); } void CSL\_dssVpSetLcdTdmConfig(CSL\_dss\_vpRegs \*vpRegs,

• When the GPIO0\_39 pin is connected to the logic analyzer, the GPIO pin is set high at around 180ms.

### 6 Results

In conclusion, implementing an early splash screen with a flicker-free transition enhances the user experience by providing a seamless and visually appealing entry point into the application. By optimizing the loading sequence and providing smooth transitions, developers can reduce perceived wait times and maintain user engagement. Focus future developments on refining these transitions and exploring further optimization techniques to enhance performance and responsiveness.

Splash Displayed From Boot Flow	Time Taken
Splash from SBL in SBL flow	180ms
Splash from U-Boot in SPL flow	680ms

#### Table 6-1. Splash Display Time Comparison





Figure 6-1. Hardware Setup

#### IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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 will 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 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.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2025, Texas Instruments Incorporated