Application Report
SPRAC82–March 2017

Linux Boot Time Optimizations on DRA7xx Devices

Venkateswara Rao Mandela

ABSTRACT

This application report provides information on benchmarking and optimizing u-boot and Linux kernel on DRA7xx platform. It also demonstrates how to reach the userspace in less than 3 seconds from reset using generic boot time optimizations.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

Boot time optimization is an area of interest for all automotive customers. This document provides a reference for boot time benchmarking on TI Linux SDK and provides an initial set of boot time optimization patches. This document focuses on bootloader and kernel optimizations only, t2 and t3 in Figure 1. A future application report focuses on user space boot time optimizations with specific usecases.

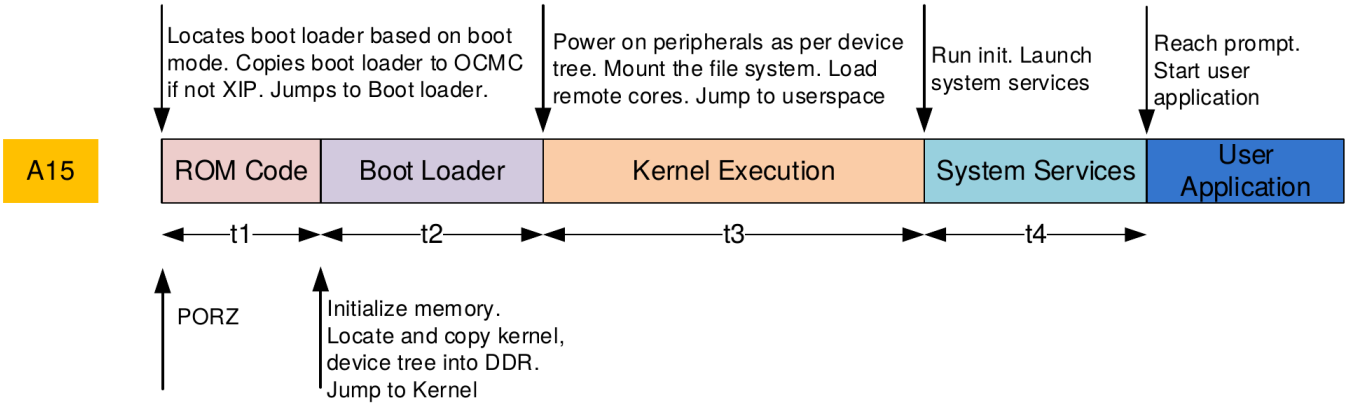

Figure 1. Linux Boot Flow

Table 1. Results

	Unoptimized	Optimized
Time spent in boot loader (t2)	413 ms	355 ms
Time spent in kernel (t3)	6241 ms	2473 ms
Time to reach userspace from PORz	6676 ms	2849 ms

In Section 2 and Section 3, the steps to replicate the results are shown in Table 1. Section 4 explains the optimizations done. Section 5 describes the benchmarking mechanism and ways to add additional benchmarking.

2 Testing the Boot Time Optimizations

2.1 Software Requirements

This document is based on Processor SDK Linux Automotive 3.02. Make sure that you do the following:

- Have a working Processor SDK Linux Automotive 3.02 installation
- Are able to build the U-boot and kernel
- Are able to bring up the EVM with the U-Boot and Kernel images you have built

The Kernel and U-Boot commits corresponding to the 3.02 SDK are shown in Table 2.

Table 2. Kernel/U-Boot Base Commits

Repository	Commit id	Headline
Kernel	89944627d53a	Late Attach: Fix for accessing second level page table
U-Boot	850ffc07ba	defconfigs: dra7xx_hs_evm: Move OPTEE load address to avoid overlaps

The release downloads links and software developers guide can be found at the following link:

Processor SDK Linux Automotive Landing wiki page

2.2 Hardware Requirements

These boot time optimizations were tested on the following:

- Rev H J6 EVM
- 1280 x 800 LG LCD

The patches should work on J6 Eco and J6 Entry as well. However, they have not been benchmarked on J6 Eco and J6 Entry at this point.

2.3 Source Code Changes

Apply the patches shown in Table 3 in the order described below on the U-boot commit shown in Table 2.

Table 3. U-Boot Patches

S. No	Headline	URL	Category
1	fastboot: update linux partition table	http://review.omapzoom.org/38255	Flashing
2	fastboot: flash: add buffer overflow check for cmd	http://review.omapzoom.org/38256	Flashing
3	fastboot: erase QSPI boot areas only when necessary	http://review.omapzoom.org/38257	Flashing
4	fastboot: add more partitions to QSPI	http://review.omapzoom.org/38258	Flashing
5	dra7xx: add functions for timestamping	http://review.omapzoom.org/38259	Benchmarking
6	spl: dra7xx: timestamp various points in execution	http://review.omapzoom.org/38260	Benchmarking
7	spl: dra7xx: add boot time measurements to dtb	http://review.omapzoom.org/38261	Benchmarking
8	dra7xx_evm: minor change to config option	http://review.omapzoom.org/38262	Optimization
9	dra7xx_evm: spl: disable env support	http://review.omapzoom.org/38263	Optimization

Apply the patches shown in Table 4 in the order described below on the kernel commit shown in Table 2.

Table 4. Kernel Patches

S. No	Headline	URL	Category
1	dra7xx: add functions for timestamping execution	http://review.omapzoom.org/38264	Benchmarking
2	config_fragments: choose lzo as the compression format	http://review.omapzoom.org/38266	Benchmarking
3	dra7xx: timestamp various points in execution	http://review.omapzoom.org/38265	Benchmarking
4	arch: arm: omap2: optimize hwmod lookup during init	http://review.omapzoom.org/38268	Optimization
5	ti_fragments: add configuration options to reduce boot time.	http://review.omapzoom.org/38267	Optimization
6	dra7: dts: disable mmc4 to save on boot time	http://review.omapzoom.org/38269	Optimization

2.4 Build Instructions

1. Build U-Boot after applying the patches shown in Section 2.3. Copy MLO and the u-boot.img into the FAT partition of the SD card.
2. Reconfigure the kernel after applying the patches shown in Section 2.3. Ensure that the config fragment, `ti_config_fragments/boot_opt.cfg` added by the patches provided in Table 4, is included in the kernel configuration.
3. Build the kernel. Update the file system with the newly built kernel image, kernel modules and device tree files.
4. Build a uImage from the zImage built by the kernel. Using the uImage format is necessary for single stage boot.

```
host $ mkimage -A arm -O linux -C none  -T kernel -a 0x80008000 \
-e 0x80008000 -n 'Linux uImage' -d zImage uImage
```

5. Update the chosen node in the device tree with boot arguments.

   The boot arguments are

```
elevator=noop console=ttyS0,115200n8 cma=64M omapdrm.num_crtc=1 consoleblank=0
snd.slots_reserved=1,1 fixrtc loglevel=0 root=/dev/mmcblk0p4 rootfstype=ext4 rw
                rootwait
```

   The following is a bash script to set the boot arguments.

```
BOOTARGS="elevator=noop console=ttyO0,115200n8 cma=64M "
BOOTARGS+="omapdrm.num_crtc=1 consoleblank=0 snd.slots_reserved=1,1 "
BOOTARGS+="fixrtc "
BOOTARGS+="loglevel=0 "
BOOTARGS+="root=/dev/mmcblk0p4 rootfstype=ext4 rw rootwait "
fdtput -v -t s "dra7-evm-lcd-lg.dtb" "/chosen" bootargs "$BOOTARGS"
```

   This step is mandatory in single stage boot as there is no other way to pass the boot arguments to the kernel. Two items to note in the bootargs are:

   - loglevel is set to 0. This reduces the boot time by eliminating UART print bottle neck.
   - root is set to /dev/mmcblk0p4, which is the eMMC rootfs partition.

   Choose the right device tree file for your setup by using the documentation at:

   http://processors.wiki.ti.com/index.php?title=Processor_SDK_Linux_Automotive_Software_Developers_Guide#Choosing_the_correct_device_tree wiki page.

2.5 Hardware Setup Instructions

1. Modify switch settings on EVM to:

```
SW2[7:0] 0000 0111
SW3[7:0] 0000 0001

SW5[9:0] 00 0001 0100

SW8[1:0] 11
```

2. Connect a USB cable from P2/USB1 to the host PC. This is used for flashing the EVM that is using fastboot or USB mass storage (ums).
3. Connect a USB cable from the USB-UART adapter on the EVM to the host PC.
4. Ensure that the 10" 1280x800 LG LCD is connected to the EVM. If you are using a different LCD, choose the corresponding device tree file instead of `dra7-evm-lcd-lg.dtb`.

2.6 Flashing Instructions

1. Insert the SD card with MLO and U-Boot into EVM. Place EVM in SD boot mode.

```
SW2[7:0] 0000 0111
SW3[7:0] 1000 0001
```

   Reboot and stop at the U-Boot prompt.

2. Run the following commands to clear any old env settings and reboot.

```
=> env default -f -a
=> env save
```

Stop again at the U-Boot prompt and enter fastboot state using the following command:

```
=> fastboot 0
```

3. Run the following commands. These commands flash the QSPI with MLO, u-boot.img, kernel and device tree. These commands also create the required partition table in eMMC.

```
host $ fastboot oem spi
host $ fastboot flash xloader MLO
host $ fastboot flash bootloader u-boot.img
host $ fastboot flash kernel uImage
host $ fastboot flash environment dra7-evm-lcd-lg.dtb
host $ fastboot oem mmc
host $ fastboot oem format
```

4. Reboot the target and stop at U-boot prompt. Enter USB Mass storage mode to transfer the file system from host to target.

```
=> ums 0 mmc 1
```

Mount the partitions on the host PC and copy the target file system to the eMMC on the EVM. Here, it is assumed that the EVM is detected as `/dev/sde`.

```
host $ cd /tmp
host $ mkdir -p emmc
host $ sudo mount /dev/sde4 emmc
host $ sudo rsync -av --delete /home/user/targetfs/3_02_00_03/ emmc/
host $ sudo umount emmc
```

Due to the size of the file system, the initial file copy might take 10-15 minutes. rsync is being used so that the same command can be used for copying the initial file system as well as the modified files after a file system update.

5. Once the above commands are complete, change SW2 setting to the following:

```
SW2[7:0] 0011 0111
```

Reboot the EVM.

2.7 Measurement Instructions

Once the command prompt is reached, run the following commands to read the boot time.

```
target # readproc; sh /etc/visualization-scripts/list-boot-time.sh
m-boardinit-time,98
m-entry-time,21
m-image-load-dur,157
m-kernelstart-time,376

k-cust-machine-dur,43
k-hwmod-dur,102
k-init-call-dur,636
k-mm-init-dur,794
k-rest-init-time,1429
k-root-wait-dur,219
k-start-time,425
k-user-space-entry-time,2849

u-prompt-time,14451

Kernel Decompression time,49
Kernel Exec time,2473
```

Section 3 describes how to interpret these results in brief. For a more detailed description, see Section 5.

3 Interpreting the Results

3.1 Optimized Boot Results

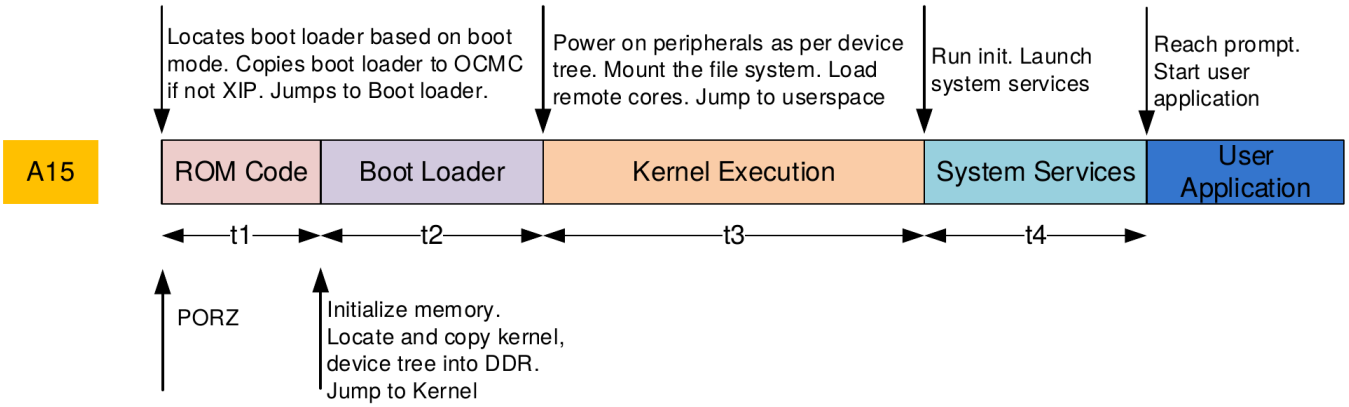

Figure 2. Linux Boot Flow

Below are the raw boot time measurements from the optimized kernel boot on Rev H DRA75x EVM. Figure 2 shows how to map the measurements to the t2, t3 shown in Figure 2.

```
m-boardinit-time,98
m-entry-time,21
m-image-load-dur,157
m-kernelstart-time,376

k-cust-machine-dur,43
k-hwmod-dur,102
k-init-call-dur,636
k-mm-init-dur,794
k-rest-init-time,1429
k-root-wait-dur,219
k-start-time,425
k-user-space-entry-time,2849

u-prompt-time,14451

Kernel Decompression time,49
Kernel Exec time,2473
```

`m-entry-time` corresponds to the time spent in the boot ROM, such as t1 shown in Figure 2. `m-kernelstart-time` corresponds to the time at which MLO hands off execution to the kernel. The time spent in bootloader is 355 ms.

```
t2 = m-kernel-start-time - m-entry-time
   = 376 - 21
   = 355 ms
```

`m-kernelstart-time` corresponds to the time at which MLO hands off execution to kernel. `k-user-space-entry-time` corresponds to the time at which Kernel starts user space execution. The time spent in the kernel is 2473 ms.

```
t3 = k-user-space-entry-time - m-kernel-start-time
   = 2849 - 376
   = 2473 ms
```

The time spent in kernel decompression can be obtained from `m-kernelstart-time`, which is the instant when MLO handed off execution to the kernel and `k-start-time` when execution reached the `init/main.c:start_kernel()` function.

```
decompress_time = k-start-time - m-kernelstart-time
                = 425 - 376
                = 49 ms
```

Table 5. Optimized Boot Time Results

	Optimized
Time spent in boot loader (t2)	355 ms
Time spent in kernel (t3)	2473 ms
Kernel Decompression time	49 ms

The complete operating parameters for these measurements are shown in Table 6.

Table 6. Operating Parameters

		Remarks
Hardware	DRA7xx Rev. H EVM with LG 1280x800 10 inch LCD	
A15 frequency	1 GHz	
Kernel size	3508 KB	Stored in QSPI
Device tree size	112 KB	Stored in QSPI
First stage bootloader(MLO) size	104 KB	Stored in QSPI
Second stage bootloader(u-boot.img) size	736 KB	Stored in QSPI
File system		Stored in eMMC

3.2 Unoptimized Boot

The raw results for the unoptimized boot are shown below. To obtain the results for the unoptimized boot, revert the following:

- patches 4,5,6 on the kernel in the table "Kernel Patches"
- patch 9 on u-boot in the table "U-Boot Patches"

Rebuild kernel and U-boot and update the EVM as per the instructions in Section 2.6. The flashing should be faster (approximately 1 min) as only a few folders are updating in the root file system. Reboot and take the boot time measurements.

```
m-boardinit-time,98
m-entry-time,22
m-image-load-dur,158
m-kernelstart-time,435

k-cust-machine-dur,44
k-hwmod-dur,206
k-init-call-dur,2981
k-mm-init-dur,810
k-rest-init-time,3105
k-start-time,2086
k-user-space-entry-time,6676

u-prompt-time,19271

Kernel Decompression time,1651
Kernel Exec time,6241
```

Use the same steps described in Section 3.1 to get results show in Table 7.

Table 7. Unoptimized Boot Time Results

	Unoptimized
Time spent in boot loader (t2)	413 ms
Time spent in kernel (t3)	6241 ms
Kernel Decompression time	1651 ms

The complete operating parameters for these measurements are shown in Table 8.

Table 8. Operating Parameters

		Remarks
Hardware	DRA7xx Rev. H EVM with LG 1280x800 10 inch LCD	
A15 frequency	1 GHz	
Kernel size	5228 KB	Stored in QSPI
Device tree size	128 KB	Stored in QSPI
First stage bootloader(MLO) size	108 KB	Stored in QSPI
Second stage bootloader (u-boot.img) size	736 KB	Stored in QSPI
File system		Stored in eMMC

Section 4 discusses the optimization steps used to achieve these results.

4 Optimization Steps

4.1 Boot Media Selection

The DRA7xx SoC supports booting from various boot media. However, from a fast boot perspective, only QSPI NOR and eMMC are relevant. One of the decisions involved in optimizing the boot time is the location of the bootloader, kernel and device tree. Based on the initialization time and the binary sizes, it was decided to boot from QSPI. eMMC has a higher initialization time, which makes it unsuitable for small image sizes.

4.2 Single Stage Boot Mode

When optimizing for boot time, the system integrator has already decided on the boot media and the locations where various binaries are stored. In this case, the flexibility offered by U-Boot is not needed and can boot from MLO to kernel directly in a single stage. This provides a saving of at least a second in boot process.

4.3 U-Boot Optimizations

The optimizations in MLO are limited to env support in MLO/SPL as it unnecessarily increases the time spent in the kernel.

U-Boot does not have any boot time optimizations as the single stage boot mode is used. However, customizations for easy flashing of QSPI and eMMC using fastboot are included. The partitions to flash the files required for usecases are defined, such as early splash (logo partition), early video (data partition), remotecore partitions (for loading remotecores early).

4.4 Kernel optimizations

4.4.1 Kernel Compression

Kernel compression is a tradeoff between the following:

- The size of the binary that increases the time read the kernel binary into DDR
- The binary decompression time on target

Based on our experiments, we determined that lzo compression offers the best tradeoff among the various supported formats.

4.4.2 Kernel Configuration

The default kernel configuration includes built-in functionality for a wide range of usecases. This increases the size of the kernel as well as the initialization time. To reduce the kernel initialization time, we remove unused functionality and convert some of the functionality not required for the early use features into modules.

The following lists some of the major kernel configuration options that can be customized to reduce the boot time. The kernel configuration modifications for boot time, `ti_config_fragments/boot_opt.cfg`, can be found in the kernel source tree.

1. Convert all file systems except the one used for the root file system into a module.

```
CONFIG_EXT2_FS=m
CONFIG_EXT3_FS=m
CONFIG_JBD=m
CONFIG_FAT_FS=m
CONFIG_MSDOS_FS=m
CONFIG_VFAT_FS=m
CONFIG_CRAMFS=n
```

2. Convert the Memory Technology Device (MTD) support into modules. Memory Technology Devices are flash, RAM and similar chips, often used for solid state file systems on embedded devices. Also, convert the associated file systems into modules.

```
CONFIG_MTD=m
CONFIG_MTD_NAND=m
CONFIG_SPI_TI_QSPI=m
CONFIG_JFFS2_FS=m
CONFIG_UBIFS_FS=m
```

3. Convert SCSI, ATA and Controller Area Network (CAN) into modules. Disable PCI, if not required.

```
CONFIG_SCSI=m
CONFIG_ATA=m
CONFIG_SATA_AHCI_PLATFORM=m
CONFIG_CAN=m
CONFIG_CAN_RAW=m
CONFIG_PCI=n
CONFIG_PCI_DRA7XX=n
```

4. Set the default frequency governor to performance. This ensures that the A15 is running at the maximum supported frequency.

   `CONFIG_CPU_FREQ_DEFAULT_GOV_PERFORMANCE=y`

5. Remove options used for debugging.

```
# Disable debug fs
CONFIG_DEBUG_FS=n

# Disable Kprobes debug infrastructure. This is a requirement for
# removing all debug symbols.
CONFIG_KPROBES=n

# Remove debug info from kernel to reduce size
CONFIG_DEBUG_INFO=n

# Disable all debug symbols
CONFIG_KALLSYMS=n
CONFIG_KALLSYMS_ALL=n
```

Beyond the generic customizations shown above, other customizations can be performed on the target hardware with the knowledge of the use case and by profiling the kernel initcalls. In Section 5 describes the benchmarking mechanism used in this document and how additional instrumentation can be added using this benchmarking mechanism.

5 Benchmarking the Boot Time

To benchmark the boot time across kernel and u-boot, patch the kernel and u-boot. The boot time measurements done in the u-boot are passed to the kernel via the device tree.

5.1 Internals of the Boot Time Benchmarking

Boot time benchmarking is performed using the 32 KHz timer part of the DRA7XX SoC. This 32 KHz timer starts within a few ms of the cold reset of the SoC. The 32 KHz timer is 32-bit wide and can be read from the all the cores on the SoC. This timer can be used to benchmark across software transitions, for example, switching from Boot ROM to bootloader or switching from bootloader to kernel or benchmarking interactions across cores.

All the measurements made in the U-boot are passed to the kernel by modifying properties on the chosen node in the device tree. All the measurements made in the kernel are also reported in the same manner. This provides a uniform reporting mechanism across U-boot and Kernel for boot time.

After the boot is complete, the following nodes are visible in the `/proc/device-tree/chosen` directory on the target.

```
root@dra7xx-evm:/proc/device-tree/chosen# ls
bootargs                k-user-space-entry-time  m-ipu1start-time
k-cust-machine-dur      m-boardinit-time         m-ipu2start-time
k-hwmod-dur             m-display-time           m-kernelstart-time
k-init-call-dur         m-dsp1start-time         m-mmc-init-dur
k-mm-init-dur           m-dsp2start-time         m-spi-init-dur
k-rest-init-time        m-entry-time             name
k-root-wait-dur         m-heap-init-dur
k-start-time            m-image-load-dur
```

The prefix of the node names indicate which stage of execution the measurement was made (see Table 9).

Table 9. Node Name Prefixes

Prefix	Interpretation	Example
m-	measurement in MLO	m-entry-time
k-	measurement in kernel	k-start-time

The suffix indicates whether the measurement is a time stamp or a duration measurement (see Table 10).

Table 10. Node Name Suffix

Suffix	Interpretation	Example	Description
-time	Indicates time from PORZ	m-entry-time	Indicates the time at which MLO execution started
-dur	Indicates duration for an operation	m-spi-init-dur	Indicates the amount of taken to initialize the QSPI peripheral

All the measurements are reported in ticks of the 32 KHz timer. One tick is equal to 30.5 µs. The value assigned to the device tree node can be read as a 32-bit big endian integer. To convert the measurements to milliseconds, run the readproc utility included in the file system. This reads each measurement in the device tree nodes, converts it into milliseconds and stores the output in a text file of the same name under `/tmp`.

The script `list-boot-times.sh` included in the target file system can be used to print these times to console.

5.1.1 Measurement Descriptions

Table 11 provides a brief description of the measurements produced by this benchmark mechanism.

Table 11. Boot Benchmark Descriptions

Binary	Device Tree entry	Description
MLO	m-entry	Time at which MLO started execution
MLO	m-boardinit	Time of entry into `common/spl/spl.c:board_init_r()`
MLO	m-heap-init-dur	Duration to initialize heap used in MLO
MLO	m-mmc-init-dur	Time taken by mmc initialization function call.
MLO	m-spi-init-dur	Time taken by QSPI initialization function call.
MLO	m-image-load-dur	Time taken to load kernel and device tree
MLO	m-kernelstart	Time at which MLO jumps to the kernel entry point
Kernel	k-start-time	Time at which execution reached `init/main.c:start_kernel()`
Kernel	k-mm-init-dur	Time taken to initialize memory map
Kernel	k-cust-machine-dur	Time taken in the `customize_machine()` call.
Kernel	k-hwmod-dur	Time taken in powering on the various peripherals specified in device tree
		`arch/arm/mach-omap2/omap_hwmod.c:omap_hwmod_setup_all()`
Kernel	k-rest-init-time	Time at which the `rest_init()` function is called.
		This function spawns the `kernel_init` thread which calls the init process.

Table 12. Boot Benchmark Descriptions

Binary	Device Tree Entry	Description
MLO	m-display	Time at which splash screen is enabled in MLO.Currently unused
MLO	m-dsp1start	Time at which DSP1 is started from MLO
MLO	m-dsp2start	Time at which DSP2 is started from MLO
MLO	m-ipu1start	Time at which IPU1 is started from MLO
MLO	m-ipu2start	Time at which IPU2 is started from MLO

5.2 Profiling Userspace Execution

The target file system also includes a binary to read the 32 K timer from user space for benchmarking. The output displayed is in milliseconds.

```
target # read32k_driver
7128
target # read32k_driver
7249
```

5.3 Additional Profiling in Kernel

The following steps profile new locations in the kernel using the 32K timer:

1. Add code to measure the timestamp using `read_fast_counter()`. For example, in `arch/arm/mach-omap2/omap_hwmod.c`.

```
extern u32 read_fast_counter(void);
u32 new_measure_time;
...
static int __init omap_hwmod_setup_all(void)
{
    new_measure_time = read_fast_counter();
```

2. Declare the variable to hold the measurement in `arch/arm/mach-omap2/board-dra7xx.h`.

```
extern u32 new_measure_time;
```

3. Define a new attribute on the "chosen" node to hold the measurement in `arch/arm/boot/dts/dra7.dtsi` in the kernel repository.

```
/ {
    chosen {
        k-new-measure-time = <0x0000000>;
    };
};
```

4. Modify the `kernel_update_dt_with_boottimes()` function in `arch/arm/mach-omap2/board-dra7xx.c` to update the device tree with the measurement.

```
void kernel_update_dt_with_boottimes(void)
{
...
    kernel_set_boottime_vals(bus, "k-new-measure-time", new_measure_time);
```

Modify the code appropriately to measure duration instead of a timestamp. All the measurements are in 32 K timer ticks and are converted to milliseconds by using the readproc userspace utility.

5.4 Additional Profiling in MLO

All the source code modifications below are in the u-boot repository except for the one required device tree modification in step 3.

1. Declare the variable to store the measurement in `include/spl.h`.

```
extern u32 new_measure_time;
```

2. Define the variable and make the measurement using the `read_fast_counter()` function, for example, in `common/spl/spl.c`.

```
u32 new_measure_time;

static int spl_load_image(u32 boot_device)
{
    new_measure_time = read_fast_counter();
```

3. Define a new attribute on the "chosen" node to hold the measurement in `arch/arm/boot/dts/dra7.dtsi` in the kernel repository.

```
/ {
    chosen {
        m-new-measure-time = <0x0000000>;
    };
};
```

Note that this is necessary to avoid resizing the device tree in MLO. Also, note that the convention of using prefix m- for measurements from MLO.

Rebuild and use the updated device tree to obtain the measurements.

4. In the `arch/arm/cpu/armv7/omap-common/boot-common.c` file, modify `spl_fdt_fixup_rom_bench_nums()` to update the device tree with the new measurement.

```
void spl_fdt_fixup_rom_bench_nums(void *fdt)
{
...
    fdt_setprop_inplace_u32(fdt, node, "m-new-measure-time",
    new_measure_time);
```

All the measurements are in 32 K timer ticks and are converted to milliseconds by using the readproc userspace utility.

6 Summary

This document describes a method to benchmark the Linux kernel and U-boot, and provides Kernel and U-Boot patches to enter user space within 2.9 s.

Table 13. Results

	Unoptimized	Optimized
Time spent in boot loader (t2)	413 ms	355 ms
Time spent in kernel (t3)	6241 ms	2473 ms
Time to reach userspace from PORz	6676 ms	2849 ms

These methods and patches will be built on in future application reports on boot time optimizations focused on specific usecases.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265