iMX 6/7/8 Boot Time and Optimization

Embedded Artists AB
Jörgen Ankersgatan 12
SE-211 45 Malmö
Sweden

https://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: www.embeddedartists.com/contact.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1 Document Revision History

Revision	*Date*	*Description*
A	2020-10-26	First release

2 Introduction

This document shows the Linux kernel boot time for Embedded Artists i.MX 6/7/8 based COM boards. There is also a section exploring **boot time optimization** for the iMX8M Mini uCOM board.

2.1 Additional documentation

Additional documentation you might need is.

- *Bootlin*: Boot time optimization
  https://bootlin.com/doc/training/boot-time/boot-time-slides.pdf
- *Embedded Linux Wiki: A pragmatic guide to boot-time optimization*
  https://elinux.org/images/6/64/Chris-simmonds-boot-time-elce-2017_0.pdf

2.2 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

`Constant width text` – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the
development workstation, i.e., on the workstation where you edit,
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a
document.
```

3 Boot times

3.1 Summary

The table below lists the boot times measured for the Embedded Artists iMX 6/7/8 COM boards. The value is number of milliseconds since the processor starts executing after a reset. Each row represents a time point / measurement point during boot and section 3.2 explains these in more detail.

Please note that this is the times measured during one test run for one specific build for the specific target. The boot time can naturally change between different runs as well as for different builds. Only one boot time optimization has been applied and that is the U-boot boot delay set to zero, see section 4.1 below. This optimization was applied to be able to better compare the boot times between the different boards since the default boot delay can be different for different boards.

Build details:

- Date: 2020-09-17
- Linux: 5.4.24
- U-boot: 2020.04
- meta-ea commit: d5dcc59ee673ad99e99da892f79ce0966f1c3b61

Time point	iMX8M Mini uCOM	iMX8M Nano uCOM	iMX8M Quad COM	iMX6 Quad COM	iMX6 DualLite COM	iMX6 SoloX COM	iMX6 UltraLite COM	iMX7 Dual COM	iMX7 Dual uCOM	iMX7ULP uCOM
SPL: board_init_f	153	168	275	402	103	127	115	102	93	-
u-boot: board_ early_init_f	1364	1030	1250	851	534	504	447	419	418	-
u-boot: board_init	1597	1243	1518	1024	632	590	529	484	485	889
u-boot: Starting kernel	2959	2682	2846	2044	1627	1570	1485	1419	1418	1697
Linux: Init process	6084	5792	6034	6434	5502	5820	5235	4622	5543	5447
Linux: Basic service	8021	7760	7925	10809	10595	12664	17235	9419	10808	17728
Linux: Login prompt	10459	12479	10643	13419	13470	15961	22532	12044	12965	23509

Table 1 - Boot times for COM boards (in milliseconds)

3.2 Time points explained

3.2.1 SPL: `board_init_f`

This is the time when entering into the `board_init_f` function in SPL (Secondary Program Loader). This is basically the first board specific code being executed after a reset. If you need to do really early initialization of your hardware this is where you can add that code.

3.2.2 U-boot: board_early_init_f

This is the time when entering into the board_early_init_f function in U-boot. This function is located in the board specific code, for example, board/embeddedartists/mx8mmea-ucom/mx8mmea-ucom.c for the iMX8M Mini uCOM board.

3.2.3 U-boot: board_init

This is the time when entering into the board_init function in U-boot. This function is located in the board specific code, for example, board/embeddedartists/mx8mmea-ucom/mx8mmea-ucom.c for the iMX8M Mini uCOM board.

3.2.4 U-boot: Starting kernel…

This is the time point where U-boot begins to prepare for booting the Linux kernel. In the console you will see the output "Starting kernel …". This output is available in the function announce_and_cleanup in the file arch/arm/lib/bootm.c.

3.2.5 Linux: Init process

This is the time point where the user-space process 'init' is started. In the console you can look for the message "Freeing unused kernel" since this is the last message before the init process is started.

This is basically the earliest time where you could have your application started. If you know what you are doing and your application is independent of the rest of the system you could start your application here. In general, it is however not recommended to exchange the init process with your application.

3.2.6 Linux: Basic service

This is the time point where a systemd service is started that have been configured to start after the basic target.

For more information about systemd see: https://www.freedesktop.org/wiki/Software/systemd/.

```
[Unit]
Description=EA after basic.target
After=basic.target

[Service]
StandardOutput=tty
Type=oneshot
ExecStart=/home/root/myoutput.sh "after basic.target"

[Install]
WantedBy=basic.target
```

3.2.7 Linux: Login prompt

This is the time point where the login prompt is available and you can login to the console.

```
NXP i.MX Release Distro 5.4-zeus imx8mmea-ucom ttymxc1

imx8mmea-ucom login:
```

3.3 How was it measured

3.3.1 U-boot

A logic analyzer was used to detect when a GPIO (pin) was toggled to measure the elapsed time to a specific point in the U-boot code. The RESET_OUT signal was used as a starting point for the

measurement. The signals were measured on the expansion board connected to the COM Carrier board V2.

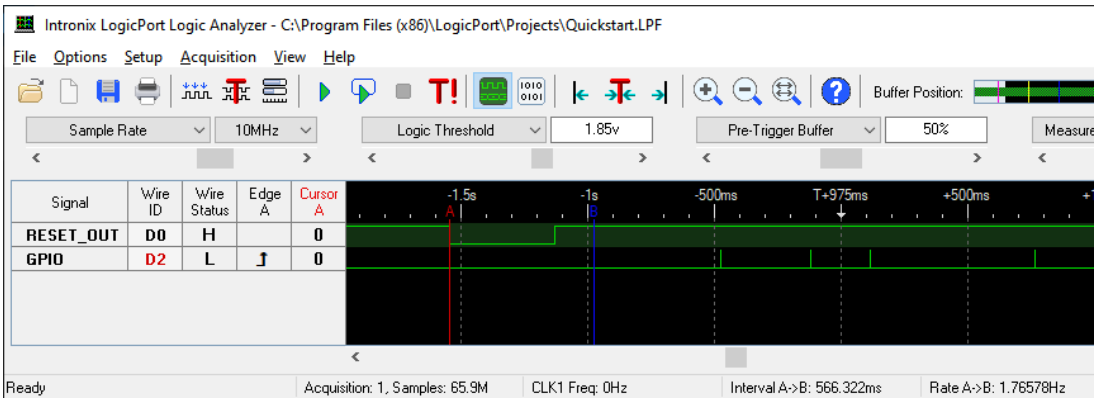

Figure 1 - Logic Analyzer

Table 2 below specifies which GPIO has been used for a specific board when measuring elapsed time.

	Processor pin name	Carrier board pin name	Expansion board
iMX8M Mini uCOM	GPIO 2.10	GPIO_AJ	J48-9
iMX8M Nano uCOM	GPIO 2.10	GPIO_AJ	J48-9
iMX8M Quad COM	GPIO 3.15	GPIO_AF	J48-84
iMX6 Quad COM	GPIO 4.08	GPIO_AJ	J48-9
iMX6 DualLite COM	GPIO 4.08	GPIO_AJ	J48-9
iMX6 SoloX COM	GPIO 4.27	GPIO_P	J48-91
iMX6 UltraLite COM	GPIO 4.11	GPIO_P	J48-91
iMX7 Dual COM	GPIO 2.18	GPIO_AJ	J48-9
iMX7 Dual uCOM	GPIO 2.18	GPIO_AJ	J48-9
iMX7ULP uCOM	PTE6	GPIO_AJ	J48-9

Table 2 - GPIO's used to measure time

3.3.2 Linux

In Linux, elapsed time has been measured using timestamps in the console application Tera Term.

Below is part of a log captured by Tera Term. Each line begins with a timestamp that can be used to calculate elapsed time between different events. For example, the output “Starting kernel” (which comes from U-boot) is at timestamp 16922 ms and “Freeing unused…” is at 20047 ms meaning that the elapsed time between these two events is 3125 ms. We also know the time from reset to “Starting kernel” from the measurement in U-boot, for example, 2959 ms for iMX8M Mini as shown in Table 1. We can now calculate time from reset to “Freeing unused…” by taking 3125 + 2959 = 6084 ms.

```
[0 00:00:16.0750] 39710 bytes read in 11 ms (3.4 MiB/s)
[0 00:00:16.0922] 28293632 bytes read in 143 ms (188.7 MiB/s)
[0 00:00:16.0922] ## Flattened Device Tree blob at 43000000
[0 00:00:16.0922]    Booting using the fdt blob at 0x43000000
[0 00:00:16.0922]    Loading Device Tree to 000000007d514000, end
[0 00:00:16.0922]
[0 00:00:16.0922] Starting kernel ...
[0 00:00:16.0922]
...
[0 00:00:20.0047] [    2.801712] Freeing unused kernel memory: 2944K
```

```
[0 00:00:20.0047] [    2.823429] Run /sbin/init as init process
...
[0 00:00:24.0422] imx8mmea-ucom login:
```

Capturing a log with timestamps in Tera Term

In Tera Term go to the File menu and then click Log as shown in Figure 2.

In the Log Dialog select where to save the log, click Timestamp in the Option section and also select Elapsed Time (Logging) in the drop-down menu. This is shown in Figure 3.

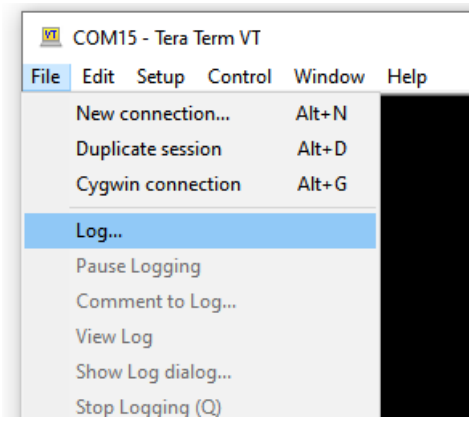

Figure 2 - Tera Term File menu

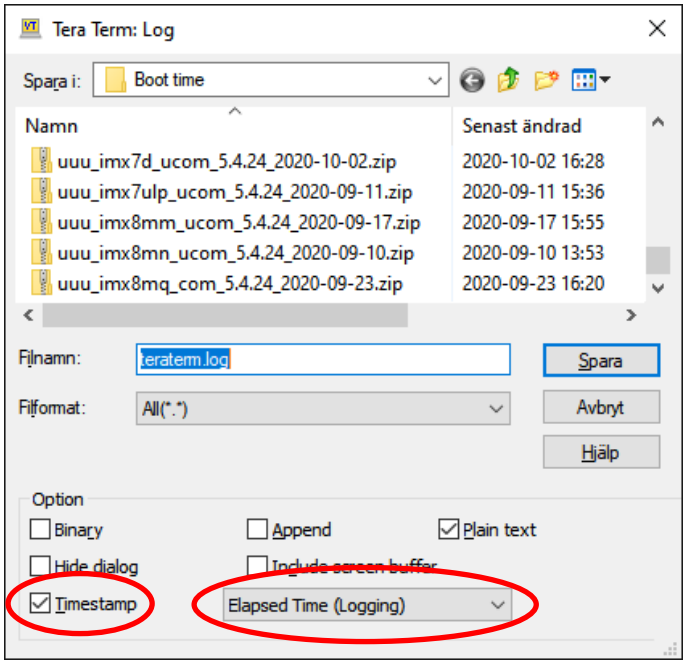

Figure 3 - Tera Term Log dialog

4 Boot time optimization – iMX8M Mini uCOM

In this chapter we explore **some of the methods** of reducing the boot time for the iMX8M Mini uCOM board. The process would be the same for a different COM board. For more ways to reduce the boot time you can look at the documentation mentioned in section 2.1 above.

Please note that the methods are explored in the order described below. Applying a method in a different order could also give a different reduced time for that specific method. For example, if you would disable peripherals in the device tree before silencing the kernel the amount of reduced time due to silencing the kernel would most likely be different. This is also the reason why the total reduced boot time as described in section 4.8 is different from the sum of all times listed in respective section.

4.1 U-boot boot delay

By default, U-boot has been setup with a boot delay of 2 seconds for the iMX8M Mini. The reason why the delay exists is to make it easier to enter into the U-boot console. If you hit any key on your keyboard before the delay expires you will enter into the console.

From within the U-boot console set the boot delay to zero using the commands below.

```
=> setenv bootdelay 0
=> saveenv
```

Reduced boot time: ~2 seconds.

> **Note**: The values specified in Table 1 in chapter 3 above already include this reduction in boot time.

4.2 Silence the kernel

The Linux kernel outputs a lot of messages to the console during boot. Outputting characters on a serial console takes time so reducing the amount will also reduce boot time. It is possible to do this by setting the kernel boot argument 'quiet'.

From within the U-boot console add 'quiet' to `extra_bootargs`.

```
=> setenv extra_bootargs quiet
=> saveenv
```

Reduced boot time: ~3.3 seconds.

Below you can see the amount of time that has been reduced at certain time points.

- Linux: Init process: ~1.9 seconds
- Linux: Basic service: ~3.0 seconds
- Linux: Login prompt: ~3.3 seconds

> **Note**: Silencing the kernel with 'quiet' might also suppress error and warning messages.

4.3 Simplify boot command

The default boot command is quite complex and involves checking if devices and files are available. If they are not trying alternative boot methods. It also involves loading a boot script from eMMC flash. For a product where we only want to boot from eMMC the boot command can be simplified as below.

Note: By doing this simplification we lose the possibility to use `extra_bootargs`. We can however use the variable `args_from_script` instead to set the kernel boot argument 'quiet'.

```
=> setenv bootcmd 'mmc dev \${mmcdev}; if run loadimage; then run
mmcboot; fi;'
=> setenv args_from_script quiet
=> saveenv
```

Reduced boot time: ~200 milliseconds.

4.4 Increase I2C speed in SPL

SPL is responsible for initializing the DDR RAM. The configuration data used during initialization is stored in EEPROM accessed via the I2C bus. Reading data from the EEPROM takes some time.

At the time of writing this document the default I2C speed in SPL is 100 kHz. Increasing this to 400 kHz will reduce the time of reading the configuration data.

To be able to increase the I2C speed you have to make two changes.

- `CONFIG_SYS_I2C_SPEED`: Set this define to 400000 in `include/configs/mx8mmea-ucom.h`.
- `CONFIG_SYS_MXC_I2C1_SPEED`: Set this configuration to 400000 in `configs/mx8mmea-ucom_defconfig`.

Reduced boot time: ~350 milliseconds.

4.5 Disable U-boot functionality

One more way of reducing boot time is by removing functionality that you don't need. In this example the following was removed from U-boot.

- Fastboot: This functionality is mainly used by the Universal Update Utility (UUU) when flashing / programming a target. You can use a separate u-boot used by UUU that still has the fastboot functionality.
- Video: If your product doesn't have a display or you don't need U-boot to show anything on the display the video functionality can be removed
- Network. If you don't need network functionality from U-boot this can be removed.
- MMC Env: There is a call to `board_late_mmc_env_init` in the board specific code that dynamically updates the `mmcdev` and `mmcroot` variables. This exists to be more flexible of choosing the mmc device used by U-boot and Linux kernel. In most cases the device is fixed and cannot be changed so the call to this function can be removed.

The following changes was made in `configs/mx8mmea-ucom_defconfig`. The configurations below were previously enabled, that is, set to 'y'.

```
CONFIG_FASTBOOT=n
CONFIG_USB_FUNCTION_FASTBOOT=n
CONFIG_CMD_FASTBOOT=n
CONFIG_ANDROID_BOOT_IMAGE=n
CONFIG_FASTBOOT_UUU_SUPPORT=n
CONFIG_VIDEO_IMX_SEC_DSI=n
CONFIG_DM_VIDEO=n
CONFIG_VIDEO_ADV7535=n
```

```
CONFIG_SYS_WHITE_ON_BLACK=n
CONFIG_NET=n
```

The following configurations were removed in `include/configs/mx8mmea-ucom.h`

```
/*
 #define CONFIG_CMD_USB_MASS_STORAGE
 #define CONFIG_USB_GADGET_MASS_STORAGE
 #define CONFIG_USB_FUNCTION_MASS_STORAGE
*/
```

Remove the call to `board_late_mmc_env_init` in `board/embeddedartists/mx8mmea-ucom/mx8mmea-ucom.c`. The call is done in the function `board_late_init`.

```
...
int board_late_init(void)
{
#ifdef CONFIG_ENV_IS_IN_MMC
//      board_late_mmc_env_init();
#endif
...
```

Reduced boot time: ~480 milliseconds.

4.6 Disable peripherals in kernel device tree (dts)

As for the U-boot, functionality can also be removed from the Linux kernel. The easiest way is to disable peripherals in the device tree (dts). If a peripheral is disabled the corresponding driver won't be fully initialized and thereby reducing the boot time. We are not showing the completely modified device tree file. It was the following functionality that was disabled.

- Audio-related
- Display-related
- Network interface
- FlexSPI
- Unused UART devices
- Unused USDHC devices
- PWM
- GPIO buffer

You disable a peripheral by setting the status in the device tree node to "disabled". In the excerpt below you can see how the network interface (fec1 node) is disabled.

```
&fec1 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_fec1>;
        phy-mode = "rgmii-id";
        phy-handle = <&ethphy0>;
```

```
        fsl,magic-packet;
        status = "disabled";

        mdio {
                #address-cells = <1>;
                #size-cells = <0>;

                ethphy0: ethernet-phy@0 {
                        compatible = "ethernet-phy-ieee802.3-c22";
                        reg = <0>;
                        at803x,led-act-blind-workaround;
                        at803x,eee-okay;
                        at803x,vddio-1p8v;
                };
        };
};
```

Reduced boot time: ~130 milliseconds.

4.7 Reduce kernel size

Loading the kernel from eMMC to RAM takes time. Reducing the size of the kernel can therefore reduce the boot time. This was tested by removing some functionality in the kernel. The kernel size shrunk from about 26 Mbyte to about 21 Mbyte. The reduced boot time was in this case only about **50 milliseconds**. This option wasn't further explored and is **not included** in the summary in section 4.8 below.

4.8 Summary of reduced boot time

In Table 3 below we compare the boot time for the unmodified distribution with the one where the above described optimizations have been applied. As you can see the total boot time from reset to login prompt has been reduced with about 5.5 seconds (from almost 10.5 seconds to about 5 seconds).

Time point	Unmodified	Optimized	Reduced time
SPL: board_init_r	153	153	0
u-boot: board_ early_init_f	1364	1013	351
u-boot: board_init	1597	1248	349
u-boot: Starting kernel	2959	1896	1063
Linux: Init process	6084	2802	3282
Linux: Basic service	8021	3583	4438
Linux: Login prompt	10459	4927	**5532**

Table 3 - Summary of reduced boot time for iMX8M Mini uCOM (in milliseconds)

5 Sleep mode to running

Besides the boot time it is also of interest to know the time it takes to wake up from a sleep mode (power-saving mode). This is for example useful if you don't want to power down the system completely, but instead set the system into a sleep mode in order to save power.

More information about sleep modes:

https://www.kernel.org/doc/html/v5.4/admin-guide/pm/sleep-states.html

5.1 Summary

Table 4 below lists the time it takes to wake up from a certain sleep mode for a standard, unmodified `ea-image-base` build. The value is number of milliseconds since a trigger point (key pressed on keyboard) until an application toggling a GPIO starts toggling the GPIO again.

Build details:

- Date: 2020-09-17
- Linux: 5.4.24
- U-boot: 2020.04
- meta-ea commit: d5dcc59ee673ad99e99da892f79ce0966f1c3b61

Sleep mode	iMX8M Mini uCOM	iMX8M Nano uCOM	iMX8M Quad COM	iMX6 Quad COM	iMX6 DualLite COM	iMX6 SoloX COM	iMX6 UltraLite COM	iMX7 Dual COM	iMX7 Dual uCOM	iMX7ULP uCOM
Standby	-	-	-	88	87	95	89	98	94	-
Suspend-to-Idle	82	15	1254	78	82	90	94	83	81	88
Suspend-to-RAM	96	29	1274	94	94	99	89	97	97	283

Table 4 - Sleep mode to running (in milliseconds)

In Table 5 below the **USB interface** connected to the USB hub on the carrier board has been **disabled** (in the device tree file). As you can see the wakeup time is much lower so the USB hub is causing delays.

Sleep mode	iMX8M Mini uCOM	iMX8M Nano uCOM	iMX8M Quad COM	iMX6 Quad COM	iMX6 DualLite COM	iMX6 SoloX COM	iMX6 UltraLite COM	iMX7 Dual COM	iMX7 Dual uCOM	iMX7ULP uCOM
Standby	-	-	-	24	24	26	35	21	25	-
Suspend-to-Idle	17	-	11	12	17	18	17	9	14	-
Suspend-to-RAM	31	-	21	27	30	30	28	26	28	-

Table 5 - Sleep mode to running without USB hub (in milliseconds)

5.2 How was it measured

A shell script was created that toggles a GPIO. The board was then configured to wake up via the TTY (UART) used by the console and then put to sleep with one of the commands described in section 5.2.3 below. When sending a character (by pressing a key on the keyboard) the board was brought up from sleep and the shell script started running again.

A logic analyzer was used and setup to trigger on a character being sent on the TTY. The time from this trigger point until the GPIO started to toggle was measured. See section 5.2.5 for information about where the UART signal can be measured on the expansion connector.

5.2.1 Shell script toggling GPIO

A simple shell script was created that toggles a GPIO as fast as possible. The example below is for the iMX8M Mini uCOM board and the script begins by setting up the GPIO to use. Table 6 below lists GPIO's and TTY's for all the boards.

```
#!/bin/bash

echo "Starting"

# GPIO 2.10 -> (2-1)*32+10=42
echo 42 > /sys/class/gpio/export
echo high > /sys/class/gpio/gpio42/direction

while :
do
        echo 0 > /sys/class/gpio/gpio42/value
        echo 1 > /sys/class/gpio/gpio42/value
done
```

The application was started by.

```
$ chmod a+x myapp.sh
$ ./myapp.sh &
```

5.2.2 GPIO's and TTY

The same GPIO has been used when measuring wakeup time as was used when measuring boot time as described in section 3.3 and Table 2.

	Processor pin name	Linux pin	TTY
iMX8M Mini uCOM	GPIO 2.10	42	ttymxc1
iMX8M Nano uCOM	GPIO 2.10	42	ttymxc1
iMX8M Quad COM	GPIO 3.15	79	ttymxc0
iMX6 Quad COM	GPIO 4.08	104	ttymxc3
iMX6 DualLite COM	GPIO 4.08	104	ttymxc3
iMX6 SoloX COM	GPIO 4.27	123	ttymxc0
iMX6 UltraLite COM	GPIO 4.11	107	ttymxc0
iMX7 Dual COM	GPIO 2.18	50	ttymxc0
iMX7 Dual uCOM	GPIO 2.18	50	ttymxc0

Table 6 - GPIO, pin number and TTY

5.2.3 Commands

The commands below were run in Linux during measurement.

Activate wakeup from TTY. Change the TTY to the one valid for the board being tested. Below is valid for the iMX8M Mini uCOM board.

```
$ echo enabled > /sys/class/tty/ttymxc1/power/wakeup
```

Suspend to Standby

```
$ echo standby > /sys/power/state
```

Suspend to Idle

```
$ echo freeze > /sys/power/state
```

Suspend to RAM

```
$ echo mem > /sys/power/state
```

5.2.4 iMX7ULP uCOM

The iMX7ULP processor is different since the Cortex-M4 is the main core while for the other processors it is the Cortex-A core. Because of this the process of measuring the wakeup time is a bit different.

The application, that by default, is running on the Cortex-M4 can wake up the Cortex-A7 core. We will use this feature instead of using wakeup from TTY. See Figure 4 for the menu being presented by the application running on the Cortex-M4 core. Pressing ‘W’ will wake up the Cortex-A7 core.

The wakeup time is measured as the time between pressing ‘W’ (console connected to Cortex-M4) and the time when the first character is outputted from the Cortex-A7 core. See section 5.2.5 for information about where a UART signal can be measured on the expansion connector.

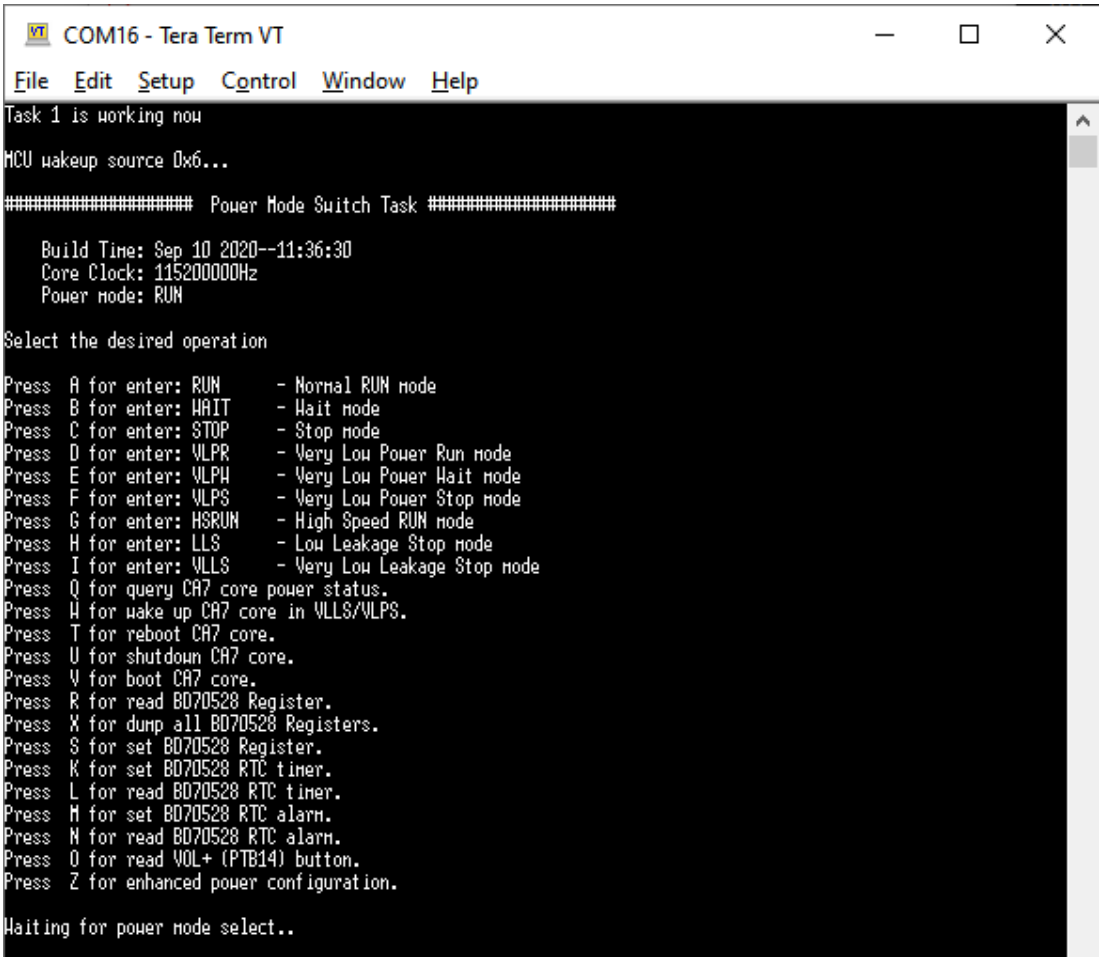

Figure 4 - Application running on Cortex-M4

5.2.5 UART / TTY pins on expansion connector

Table 7 below lists the UART pins that was used during measurements of wakeup times. The last column specifies where the signal can be found on the expansion board connected to the COM carrier board V2.

	Description	Expansion board
UART-A_RXD	Data sent from console application to Linux (Cortex-A core). Used to detect when a user presses a key on keyboard to wake up Linux from sleep mode.	J48-49
UART-A_TXD	Data sent from Linux to console application. For iMX7ULP this pin is used to detect when the Cortex-A core wakes up.	J48-74
UART-C_RXD	UART connected to the Cortex-M4 core on iMX7ULP. Used to detect when pressing 'W' on keyboard to inform the Cortex-M4 application to wake up the Cortex-A core.	J48-72

Table 7 - UART pins available on expansion board

6 Hardware related delays

The times listed in chapters 3 and 4 are all relative to the time when the processor starts executing code. There are delays related to the hardware, such as PMIC power rail sequencing, that occur before the processor starts executing code. Table 8 below lists delays for cold and warm reset for respective board. Note that the delays can vary over temperature and individual boards due to tolerance of on-board reset generators and time bases.

If you need to know the total boot time from a cold/warm reset until the kernel is running you should take the time below and add to the time listed in chapter 3 above.

	iMX8M Mini uCOM	iMX8M Nano uCOM	iMX8M Quad COM	iMX6 Quad COM	iMX6 DualLite COM	iMX6 SoloX COM	iMX6 UltraLite COM	iMX7 Dual COM	iMX7 Dual uCOM	iMX7ULP uCOM
Cold reset	65	65	278	584	584	526	537	500	276	515
Warm reset	350	457	251	252	252	200	238	233	134	645

Table 8 - Hardware reset delays (in milliseconds)

6.1 Cold reset

Cold reset is the same as switching on power to the board. The time is measured from valid level on VIN until positive edge on RESET_OUT.

6.2 Warm reset

Warm reset could be triggered from software or if a reset button is pressed. The time is measured from positive edge on RESET_IN until positive edge on RESET_OUT (except for the iMX8M Mini/Nano uCOM where it is negative edge on RESET_IN until positive edge on RESET_OUT).