iMX 6/7/8 Power consumption, sleep and wake-up

Embedded Artists

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

https://www.EmbeddedArtists.com

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.

Table of Contents

1 Document Revision History

Revision	***Date***	***Description***
A	2020-11-12	First release

2 Introduction

This document shows the power consumption for Embedded Artists i.MX 6/7/8 based COM boards.

2.1 Additional documentation

- *Get started with the 7-inch HDMI Display Kit* – This document contains instructions for how to use the 7-inch display with a Developer's Kit.
- *iMX 6/7/8 Boot time and optimization* – This document measures boot time, but also the time it takes to wake from a sleep/power-down mode.
- *Getting Started with M.2 modules and iMX6,7,8* - This document shows how to configure Wi-Fi modules

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.
```

This field is used to highlight an important fact or something to take note of.

3 Power consumption

3.1 Summary

The table below lists power consumption measured on Embedded Artists iMX 6/7/8 COM boards. The power consumption is measured during one test run for one specific build for the specific target and this can naturally change for different builds. Measurement has been done on a COM Carrier board V2 with a uSD card inserted in J22 connector and a network cable inserted in J9 connector. The 7-inch HDMI display has been connected to the board for the video playback measurement.

The consumption is rather stable for a low-power mode when nothing is running on the board, but can vary when the processor is in a running state depending on the applications and services that are running.

It is important that you measure consumption on your specific hardware and software setup to get accurate numbers. Use the values below as a starting point when comparing different boards.

The consumption is measured with a multi-meter on JP1 on the COM Carrier board V2 over a 100-milliohm resistor.

Build details:

- Date: 2020-10-27
- Linux: 5.4.24
- U-boot: 2020.04
- meta-ea commit: ff1185db979eb1604c24cef1e81455b4ab28a526

	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**
Running, idle	1295	1212	2556	1201	1051	898	1055	871	1004	385
Suspend-to-idle (freeze)	784	718	1808	874	616	443	619	380	465	213
Suspend-to-ram (mem, deep)	95	120	674	227	200	187	474	195	200	113
Standby	-	-	-	379	323	288	500	179	194	130
Iperf3	1606	1523	2754	2379	2158	1700	1229	1147	1317	-
Memtester	2507	2220	3407	3075	2783	2240	1503	1472	1455	481
Video playback	1872	1614	2840	2033	2006	1496	1484	1166	1207	-

Table 1 – Power consumption for COM boards (in milliwatt, mW)

3.2 Measure points explained

3.2.1 Running, idle

This is the state when Linux has booted and the user has logged in. Measurement has been taken 1 minute after login.

3.2.2 Suspend-to-idle

The system has been set to suspend-to-idle state. This is done from within Linux by running the command below.

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

See the kernel documentation for more information about suspend-to-idle.

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

3.2.3 Suspend-to-ram

The system has been set to suspend-to-ram state which most often is the most energy saving state. This is done from within Linux by running the command below.

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

See the kernel documentation for more information about suspend-to-ram.

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

3.2.4 Standby

The system has been set to standby state. This is done from within Linux by running the command below.

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

See the kernel documentation for more information about standby.

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

3.2.5 Iperf3

Here we use the tool `iperf3` to test the network interface for maximum achievable bandwidth. Power consumption is measured while running this network test.

You need to start an iperf server on for example a Linux host machine.

```
# iperf3 -s
```

The next step is to start the tool on the target. Below is an example of how to run the tool when the server has an IP address of 192.168.1.123.

```
# iperf3 -c 192.168.1.123 -p 5201 -i 1 -P 4
```

3.2.6 Memtester

At this measure point we run the tool `memtester` to stress-test the memory. This tool will put some heavy load on the processor. The tool is started as below.

```
# memtester 100
```

3.2.7 Video playback

Here we measure the power consumption during video playback, that is, decoding a video file and rendering it on a display. Video files can be downloaded from https://www.sample-videos.com/. The video file that has been used has 1280x720 pixels resolution. The display has also been setup to use this resolution.

Start video playback as below. For more information read the document *Get started with the 7-inch HDMI display kit* found on Embedded Artists website.

```
# gst-play-1.0 big_buck_bunny_720p_30mb.mp4
```

4 Reduce power consumption

Reducing power consumption is a project in it self and very much depend on the final application and its requirements. The numbers presented in Table 1 should not be taken as final numbers that will be valid for your specific application. The sections below show some findings that might be of interest when investigating power consumption.

4.1 iMX8M Mini uCOM – Ethernet interface

If you are not using the Ethernet interface you might want to disable it in the device tree file (dts). This will however not necessarily reduce power consumption. What it means to disable the interface is that the driver won't be enabled and initialized. This can in some situations be counter productive when it comes to power consumption.

As an example, disabling the `fec1` interface in `imx8mmea-com.dtsi` while still keeping an Ethernet cable inserted into the J9 connector on the COM carrier board will increase power consumption. The suspend-to-ram state will go from 95 mW to 390 mW for this use-case. Removing the Ethernet cable will however **reduce the consumption** to **83 mW**.

The reason why consumption increase is because the driver will never power-down the Ethernet PHY since the driver will never be initialized and active. Having an Ethernet cable inserted into the connector will put the Ethernet PHY in a state where it is active and therefore consuming power. This behavior might be different for different Ethernet PHY's.

This is an example of where it is **not obvious** how reducing power should be performed.

4.2 iMX7ULP – power down Cortex-M core

The iMX7ULP application processor is different compared to the other processors since the Cortex-M core is the main core and not the Cortex-A core. In the numbers presented in Table 1 the Cortex-M core is still running.

Powering down the Cortex-M core to the "Very Low Leakage Stop Mode" while Linux / Cortex-A core is in suspend-to-ram state will **reduce power consumption** from 113 mW to **59 mW**.

5 Wake-up sources

When putting the processor in a power-down / sleep mode it must be possible to take it out of this mode, that is, wake it up from sleep. There are different ways of doing this.

5.1 UART

It is possible to configure a UART to wake up the processor when data is transmitted on the UART. This is for example useful when testing power-down modes by configuring the UART used as console to wake up the processor. It is then enough to send a character (press any key on keyboard) via the terminal application, such as Tera Term, connected to the console.

First you need to identify which UART (TTY) that is used as console. The example below is for the iMX8M Mini uCOM board where `ttymxc1` is used as console.

```
# dmesg | grep console
[    0.000000] Kernel command line: console=ttymxc1,115200
root=/dev/mmcblk2p2 rootwait rw
```

Enable wake-up on `ttymxc1`.

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

5.2 GPIO

It is possible to configure a GPIO (pin) to wake up the processor when the state of the pin changes. This can be done by defining a gpio-key in the device tree file (dts) and setting that key as a wakeup-source. The example below was added to the dts file for the iMX8M Mini uCOM board. GPIO2.10 is used in the example and this GPIO is available on the expansion board, connector J48-9.

Note: Pin muxing (pinctrl) of GPIO2.10 is performed in pinctrl_exp_conn. If you chose a different GPIO you also must make sure that pin muxing is performed.

```
        gpio-keys {
                compatible = "gpio-keys";

                wakeup_button {
                        label = "Power Button";
                        gpios = <&gpio2 10 GPIO_ACTIVE_LOW>;
                        linux,code = <KEY_POWER>;
                        wakeup-source;
                };
        };
```

When the processor has been put in a power-down mode you can wake it up by pulling pin GPIO2.10 high (3.3V).

5.3 RTC

It is possible to configure the RTC to wake up the processor. The example below is for the iMX8M Mini uCOM board where the RTC is configured to generate a wake-up event after 10 seconds. Please note that the alarm will be triggered 10 seconds after you have issued this command, not 10 seconds after you have put the processor in sleep mode.

```
# echo +10 > /sys/class/rtc/rtc0/wakealarm
```

5.4 Wake on LAN

Another way to wake up the processor is over the network. It can be possible to configure the network interface to wake the processor when a "magic packet" is sent to the targets MAC address.

First of all, the network node in the device tree must have been enabled to accept magic packets. Below is how this looks like for the iMX8M Mini uCOM board (where it is enabled by default)

```
&fec1 {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_fec1>;
    phy-mode = "rgmii-id";
    phy-handle = <&ethphy0>;
    fsl,magic-packet;
    status = "okay";
```

Wake-on-lan must then also be enabled using a tool called `ethtool`. At the time of writing this document this tool is not included in a default `ea-image-base` build. It can be added to the build by modifying the `local.conf` file as below.

```
IMAGE_INSTALL_append = " \
   ethtool \
"
```

The example below enables wake-on-lan on the `eth0` interface.

```
# ethtool -s eth0 wol g
```

You need to know the MAC address of the network interface. Use `ifconfig` to get this information. In this example the MAC address is `00:1A:F1:10:27:B8.`

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1A:F1:10:27:B8
          inet addr:192.168.1.90  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21a:f1ff:fe10:27b8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1402 errors:0 dropped:872 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:95208 (92.9 KiB)  TX bytes:10292 (10.0 KiB)
```

On a Linux host you can use the tool `wakeonlan` to send the magic packet to a target and wake up the processor.

```
# wakeonlan 00:1A:F1:10:27:B8
```

5.5 Wake on Wireless LAN (WoWLAN)

Another way to wake up the processor is over the wireless network. It can be possible to configure the network interface to wake the processor when a "magic packet" is sent to the target's MAC address.

The instructions below have been tested on an iMX8M Mini uCOM board together with a **1MW M.2 module**.

First of all, make sure that the connection to the wireless network is working by following the instructions in *Getting Started with M.2 modules and iMX6,7,8.*

Check which device is in use, here it shows `phy#0`:

```
# iw dev
phy#0
        Unnamed/non-netdev interface
                wdev 0x2
                addr a2:c9:a0:3d:41:a9
                type P2P-device
                txpower 31.00 dBm
        Interface wlan0
...
```

Disable any preexisting wakeup conditions:

```
# iw phy#0 wowlan disable any
```

Check that wowlan is disabled.

```
# iw phy#0 wowlan show

WoWLAN is disabled.
```

Enable detection of the magic packet

```
# iw phy#0 wowlan enable magic-packet

# iw phy#0 wowlan show

WoWLAN is enabled:
 * wake up on magic packet
```

You need to know the MAC address of the network interface. Use `ifconfig` to get this information. In this example the MAC address is `E8:E8:B7:07:FB:BD`. Note that the interface name could be either wlan0 (as below) or mlan0 depending on which wireless module is being used.

```
# ifconfig
wlan0     Link encap:Ethernet  HWaddr E8:E8:B7:07:FB:BD
          inet addr:192.168.5.212 Bcast:192.168.5.255  Mask:255.255.255.0
          inet6 addr: fe80::eae8:b7ff:fe07:fbbd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1070 errors:0 dropped:7 overruns:0 frame:0
          TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:71361 (69.6 KiB)  TX bytes:11934 (11.6 KiB)
```

On a Linux host (make sure it is connected to the same wireless network) you can use the tool `wakeonlan` to send the magic packet to a target and wake up the processor.

```
# wakeonlan E8:E8:B7:07:FB:BD
```