Application note

STM32MP1 Series system power consumption

Introduction

STM32MP1 Series devices are built on an Arm® Cortex®-A7 with single or dual-core MPU subsystem combined with an Arm® Cortex®-M4 CPU.

STM32MP1 datasheets present the devices power consumption values calculated using bare metal software (not using Linux operating system). The values are available for basic Run modes and for various low-power modes.

This application note provides power consumption values on various use cases measured on a STM32MP157C device mounted on a STM32MP157C-EV1 evaluation board and running on STM32 MPU OpenSTLinux Distribution.

The values provided in this document are indicative typical values measured on one sample at room temperature. User may measure different values depending on device characteristics (slow, typical, fast) and ambient temperature.

For further information on STM32MP1 Series devices, refer to the following documents and deliverables available on *www.st.com* and http://wiki.st.com/stm32mpu :

- STM32MP1 Series reference manuals (see Table 1. Configuration of the lines of the STM32MP1 Series for details)
- STM32MP1 Series datasheets
- STPMIC1x datasheet
- *STM32MP1 Series using low-power modes* application note (AN5109)
- *Getting started with STM32MP1 Series hardware development* application note (AN5031)
- *STM32MP15x and STPMIC1x HW and SW integration* application note (AN5089)
- STM32CubeMP1
- STM32MP15 Series embedded software
- STM32MP1 wiki article *Getting started with ST boards section*

1 Overview

This document is applicable to all the devices of the STM32MP1 Series. The table below describes the main characteristics of all the product lines of the STM32MP1 Series.

Depending on the device part number, the system includes an Arm® Cortex®-M4 and either a single-core or a dual-core Arm® Cortex®-A7. In this document, the Arm® Cortex®-A7 is called MPU and the Arm® Cortex®-M4 is called MCU.

The full featured system (see the table below) is partitioned in:

- One MPU subsystem: dual Arm® Cortex®-A7 with L2 cache
- One MCU subsystem: Arm® Cortex®-M4 with associated peripherals clocked according to CPU activity

The present document assumes a full featured device (for example STM32MP157).

Table 1. Configuration of the lines of the STM32MP1 Series

Lines	Reference manual	Cortex-A7 configuration	Cortex-M4	GPU	DSI	FDCAN
STM32MP151	RM0441	Single-core	Yes	No	No	No
STM32MP153	RM0442	Dual-core	Yes	No	No	Yes
STM32MP157	RM0436	Dual-core	Yes	Yes	Yes	Yes

This document applies to Arm®-based devices.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 STM32MP157x-EV1 evaluation board

The STM32MP157C-EV1 evaluation board is composed of four key components:

- Motherboard (MB1262)
- Daughterboard (MB1263)
  - STM32MP157C 18 × 18, PMIC (power management integrated circuit), DDR3
- Daughterboard camera (MB1379)
- Daughterboard DSI display serial interface (MB1230)
  - MIPI(R) standard, 720p display

Detailed description of the board can be found in the STM32MP1 wiki web page *http://wiki.st.com/stm32mpu/* on the article “STM32MP157C-EV1 - hardware description”.

2.1 Measurement points

On the STM32MP157C-EV1 evaluation board we cannot directly measure the consumption of the various voltage domains (VDD, VDDCORE, VDD_DDR).

The measurements provided in this document have been done after modification of the daughter board (MB1263). Some 0Ω shunt resistors (R80, R81, R82) present on the back side of the board have been replaced by 0.1Ω resistors on VDDCORE, VDD, VVDQ_DDR.

Wires have been soldered to the resistors shown in the figure below in order to measure the voltage across them and deduce current.

Figure 1. Resistors to be replaced for current measurement

Unless otherwise specified, measurements are done using both mother and daughter boards (MB1262 and MB1263).

2.2 STM32MP1 OpenSTLinux Distribution

Unless otherwise specified all measurements have been done using STM32MP1 OpenSTLinux Distribution starter package

openstlinux-4.19-weston-mp1-19-02-20 ‘ev1’ version.

3 Power consumption on STM32MP1 Series

This section provides some use cases of power consumption measurements using STM32MP157C devices mounted on a STM32MP157C-EV1 evaluation board.

3.1 CoreMark® on Linux MPU

First step: prepare CoreMark® executable

The following commands must be run on the Linux station:

1. Download CoreMark® from EEMBC (embedded microprocessor benchmark consortium)

```
>>mkdir ~/Coremark
>>cd ~/Coremark
>>git clone https://github.com/eembc/coremark.git
```

2. Create specific STM32MP1 folder

```
>>mkdir STM32MP1_linux
>>cp_linux/* STM32MP1_linux
```

3. Update STM32MP1_linux/core_portme.mak

   Comment the lines below:

   # CC = gcc

   #LFLAGS_END += -lrt

   Update the lines with IP@ for Ethernet connection:

   LOAD = scp $(OUTFILE) root@<IP@>:/home/root/

   RUN = ssh root@<IP@> -c /home/root/

   Comment the lines below:

   #LOAD = echo Loading done

   #RUN =

Note: <IP@> can be found running ifconfig command.

4. Update STM32MP1_linux/Makefile

   Remove space between LOAD and OUTFILE in Makefile

   $(LOAD)$(OUTFILE)

5. Compile CoreMark® for one core

```
>>cd ~/Coremark
>>make PORT_DIR=STM32MP1_linux
>>cp coremark.exe coremark_1core.exe
```

6. Update STM32MP1_linux/core_portme.mak for two cores

   Change below line

   *XCFLAGS=-DMULTITHREAD=**2** -DUSE_FORK*

7. Compile CoreMark® for two cores

```
>>cd ~/Coremark
>>make PORT_DIR=STM32MP1_linux
>>cp coremark.exe coremark_2core.exe
```

8. Make *.exe executable

```
>>chmod +x coremark_1core.exe coremark_2core.exe
```

9. Copy CoreMark® executable onto the MB1263 SDCard:
   Mount the SDCard onto the Linux PC or virtual machine (using SDCard slot or USB SDCard reader key)

```
>>sudo cp coremark_*.exe [path to /userfs on SDCard]
>>sync
```

Note: Administrator rights are needed on the machine in order to run 'sudo'.

Second step: power measurements

To run CoreMark® on the board run the commands below:

```
>>cd /usr/local
>>./coremark_1core.exe 0x0 0x0 0x66 0 7 1 2000 > run1.log
or
>>./coremark_2core.exe 0x0 0x0 0x66 0 7 1 2000 > run2.log
```

The table below shows the consumption measurements done on CoreMark® with 1 and 2 MPU cores.

Table 2. CoreMark power consumption on STM32MP157C + MB1263

-	Power VDDCORE VDDCORE = 1.2 V (mW)	Result Iteration per second
CoreMark® on 1 MPU core (650 MHz)	420	1887
CoreMark® on 2 MPU cores (650 MHz)	512	3774

3.2 Graphical use cases

The graphical use cases are built on two main components:

- glmark2-es2-wayland
  purpose: run graphical interface on DSI screen
  AVC_High@L4.0_
  1440x1080_25fps_
  MP1L2_10mn30s_TC_AVD_58
- memtester
  purpose: generate a heavy load on the DDR traffic

The clock configurations used are:

- MPU clock: 650 MHz
- GPU clock: 533 MHz
- DDR clock: 528 MHz
- MCU clock: off

Run use cases

1. glmark2-es2-wayland

```
>>glmark2-es2-wayland --off-screen --size 1280x720 -b texture:duration=60.0:texture-filter=nearest --run-forever &
```

   Measure the consumption.
2. glmark2-es2-wayland + 2 memtester

```
>>memtester 5 12 > /dev/null &
>>memtester 5 12 > /dev/null &
```

Measure the consumption.

Table 3. Graphical use cases: power consumption on STM32MP157C + MB1262/MB1263/MB1379/MB1230

-	Power VDD VDD = 3.3 V	Power VDDCORE VDDCORE = 1.2 V	Power VDD_DDR VDD_DDR=1.34 V
glmark2-es2-wayland	100	500	440
glmark2-es2-wayland + 2 memtester	100	500	576

3.3 MCU data acquisition with/without MPU

This use case provides some highlight on possible power consumption gain when setting the MPU in CStop mode while keeping the MCU in Run mode to perform some data conversion on ADC.

The software used to run this use case is included in the OpenSTLinux starter kit and can be accessed by choosing: ‘’*3: stm32mp157c-ev1-m4-examples-sdcard*” during boot process. This is a non Weston validation software.

Then at Linux prompt run:

```
>>cd /usr/local/Cube-M4-examples/STM32MP157C-EV1/Applications/OpenAMP/OpenAMP_TTY_echo_wakeup
>>./fw_cortex_m4.sh start  **start OpenAMP_TTY_echo_wakeup on MCU**
>>stty -onlcr -echo -F /dev/ttyRPMSG0
>>cat /dev/ttyRPMSG0 & **create virtual Uart connection between MPU and MCU /dev/ttyRPMSG0**
```

Then below commands can be used:

```
>>echo "*stop" > /dev/ttyRPMSG0 **set MCU in CStop**
>>echo "*standby" > /dev/ttyRPMSG0 **set MCU in CStop with PDDS = 1**
>>echo "start" > /dev/ttyRPMSG0 **set MCU in CRun**
```

The configuration used is:

- MCU Clock ~ 209 MHz
- ADC clock source = CK_PER (HSI) = 64 MHz
- Prescaler DIV2
- Regular conversion
- Resolution 16 bytes
- Sampling rate from Trigger TIM2_TRGO ~ 1 KHz

Run use cases

1. MPU + MCU running ADC acquisition
   Measure the consumption.
   (After command ./fw_cortex_m4.sh was started, OpenAMP_TTY_echo_wakeup was started on MCU).
2. MPU CStop + MCU running ADC acquisition

   ```
   >>echo mem > /sys/power/state **set MPU in CStop**
   ```

   Measure the consumption.

Table 4. **Data conversion: power consumption on STM32MP157C + MB1263**

	Power VDD VDD = 3.3 V (mW)	Power VDDCORE VDDCORE = 1.2 V (mW)	Power VDD_DDR VDD_DDR=1.34 V (mW)
MPU + MCU running ADC acquisition	100	391	225
MPU CStop + MCU running ADC acquisition	24	114	203

3.4 Various low-power modes with OpenST Linux

STM32MP1 Series devices can be used in various power modes, their respective expected consumption is provided in the table at the end of this section.

The measurements are provided on mother and daughter board (MB1262 + MB1263) using a “ev1” Weston distribution, which means that some current on VDD supply is dissipated in low-power modes due to components on the mother board (screen, Ethernet, or other) and IO’s connected from the STM32 MPU daughter board to the mother board.

Measurements done on daughter board only (MB1263) are also provided for VDD (but not possible to measure with provided distribution without recompilation to remove Weston graphical manager), also a patch has been applied deactivating eMMC to remove eMMC current on VDD due to the fact that the daughter board design is not optimized for power consumption.

Run use cases

1. CRun

   A ‘For’ loop that takes about 100% of one A7 core is launched using below commands.

```
>>while true; do echo “” > /dev/null;done
```

2. Stop

   This mode is not enabled on the provided distribution. LP-Stop is used instead.

3. LP-Stop

   The activation of a wake-up source that prevents entering into the deeper low-power mode is needed.

   UART wake-up source can be used.

```
>>echo enabled > /sys/devices/platform/soc/40010000.serial/tty/ttySTM0/power/wakeup
>>echo enabled > /sys/devices/platform/soc/40010000.serial/power/wakeup
>>echo mem > /sys/power/state
```

4. LPLV-Stop

   This measurement is not available at this time using the provided distribution.

   It requires to have GPIO wakeup activation implemented in the device tree.

   Measurement provided.

5. Standby DDR in Self Refresh

   Reset the board (to cleanup any unwanted wake-up source settings)

```
>>echo mem > /sys/power/state **A7 in CStop and System in Standby, DDR in SR**
```

6. Standby DDR OFF

   Reset the board or press wakeup button

```
>>shutdown -h 0 **A7 in CStop with PDDS=1 and System in Standby, DDR Off**
```

Table 5. Power consumption in various power mode with STM32MP157C + MB1263

System power modes	Power VDDCORE VDDCORE = 1.2V (mW)	Power VDD VDD = 3.3V (mW)	Power VDD_DDR VDD_DDR=1.34V (mW)
CRun (~100% of 1 A7 core)	460	110	220
CSleep Linux prompt after boot (~CSleep)(1)	400	90 21(2)	220
Stop(3) DDR Self Refresh termination resistors ON	32	32 7(2)	200
LP-Stop DDR Self Refresh termination resistors OFF	32	32 7(2)	75
LPLV-Stop DDR Self Refresh termination resistors OFF	(VDDCORE = 0.9 V) 12	32 7(2)	75
Standby DDR Self Refresh termination resistors OFF	(VDDCORE= 0 V) 0	22 0.5(2)	75
Standby DDR OFF	(VDDCORE= 0 V) 0	11.5 0.5(2)	(VDD_DDR = 0 V) 0

1. *Under Linux, due to kernel scheduler unpredictability, system CSleep requests occur in an asynchronous way versus IP activity (for example internal FIFO in UART or DMA transfer). As a result, only few IPs can be configured to have their clock switched off in CSleep (using bit…LPEN bit RCC). At Linux prompt the system is 90% of the time in CSleep (MPU clock off) and rest of the time distributes IRQ and threads to A7 cores.*
2. *Measurement on MB1263 only "ed1-non Weston" distribution with eMMC de-activation patch.*
3. *Stop mode is not available in the OpenStLinux distribution, a dedicated software was used to do the measurement.*

For more details on power modes of the STM32MP1 Series, refer to application note *"Using low-power modes with STM32MP1 Series"* (AN5109).

Revision history

Table 6. **Document revision history**

Date	Version	Changes
12-Mar-2019	1	Initial release.

Contents

List of tables