

# AM335x PSP User's Guide

AM335x PSP  
User's Guide

Translate this page to [cs - Český](#)

 

## AM335x PSP User Guide

Linux PSP

### Contents [\[hide\]](#)

- 1 [About this Manual](#)
- 2 [AM335X EVM Configuration](#)
- 3 [Installation](#)
  - 3.1 [Prerequisites](#)
  - 3.2 [PSP Package Contents](#)
  - 3.3 [Install U-Boot](#)
  - 3.4 [Install Linux Kernel](#)
  - 3.5 [Filesystem](#)
  - 3.6 [Toolchain](#)
    - 3.6.1 [Environment Setup](#)
- 4 [Flashing Tools](#)
- 5 [U-Boot](#)
- 6 [Linux Kernel](#)
- 7 [Booting Linux Kernel](#)
  - 7.1 [Boot from MMC/SD](#)
  - 7.2 [Boot from NAND](#)
  - 7.3 [Boot from SPI](#)
  - 7.4 [Boot over Network \(Ethernet\)](#)
- 8 [Various Module User's Guide](#)
  - 8.1 [GPIO Driver](#)
  - 8.2 [DCAN Driver](#)
  - 8.3 [EDMA3 Driver](#)
  - 8.4 [Audio Driver](#)
  - 8.5 [USB Driver](#)
  - 8.6 [MMC/SD driver](#)
  - 8.7 [McSPI Driver](#)
  - 8.8 [Watchdog Timer \(WDT\)](#)
  - 8.9 [CPSW Driver](#)
  - 8.10 [LCD Frame Buffer Driver](#)
  - 8.11 [Touchscreen Driver](#)
  - 8.12 [Power Management](#)
  - 8.13 [NAND Driver](#)
  - 8.14 [PWM Driver](#)
  - 8.15 [EVM On-Board Components Drivers](#)
- 9 [Modifying Pin Mux settings](#)
  - 9.1 [Method I - Modify Linux Kernel, rebuilt and use](#)
  - 9.2 [Method II - Modify pin-mux from arguments passed to Kernel](#)
  - 9.3 [Method III - Modify pin-mux from Linux Console](#)
- 10 [Clock Management details](#)
  - 10.1 [Clock tree from debugfs](#)
  - 10.2 [Parent change](#)
  - 10.3 [Set rate support](#)

## About this Manual

This document describes how to install and work with Texas Instruments Platform Support Package (PSP) for the AM335x platform. This PSP provides a fundamental software platform for development, deployment and execution on AM335x EVM. It abstracts the functionality provided by the hardware. The product forms the basis for all application development on this platform.

In this context, the document contains instructions to:

- [Install the release](#)
- [Build the sources contained in the release](#)

The document also provides detailed description of drivers and modules specific to this platform - as implemented in the PSP.

### Navigation

- [Main Page](#)
- [All pages](#)
- [All categories](#)
- [Popular pages](#)
- [Popular authors](#)
- [Popular categories](#)
- [Category stats](#)
- [Recent changes](#)
- [Random page](#)
- [Help](#)
- [Google Search](#)

### Print/export

- [Create a book](#)
- [Download as PDF](#)
- [Printable version](#)

### Toolbox

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Permanent link](#)
- [Browse properties](#)

## IMPORTANT

This release has been tested on AM335x EVM, BeagleBone board

# AM335X EVM Configuration

For more information regarding EVM Combinations & Configurations, Please refer to [EVM reference manual](#).

## Installation

### Prerequisites

Before Starting the installation of the package please make sure below system requirements are met:

- Host machine running a version of Windows OS such as Windows XP SP3, Linux such as Ubuntu.
- AM335x BeagleBone or EVM board

The Windows machine is used for:

- Running CCSv5.1, which will be used to build flash writers.
- Burn the boot images (U-Boot) onto the flash using the flash writers provided.

The Linux host is used for the following:

- Recompiling U-Boot and the Linux kernel.
- Hosting the NFS server to boot the EVM with NFS as root filesystem.

Either of Windows or Linux host can be used for:

- Hosting the TFTP server required for downloading kernel and file system images from U-Boot using Ethernet.
- Running a serial console terminal application

## PSP Package Contents

### IMPORTANT

The values of MM, mm, pp and bb in this illustration will vary across the releases and actually depends on individual component versions.

Extract the contents of release package on Linux host with the following command:

```
$ tar -xvzf AM335x-LINUX-PSP-MM.mm.pp.bb.tgz
```

This creates a directory AM335x-LINUX-PSP-MM.mm.pp.bb with the following contents:

```
\---AM335x-LINUX-PSP-MM.mm.pp.bb
|---AM335xPSP_Software_Manifest
+---docs
|   |---AM335x_PSP_MM.mm.pp.bb_Release_Notes.pdf
|   |---AM335x_PSP_User_Guide.pdf
|   |---AM335x_Audio_Driver_User_Guide.pdf
|   |---AM335x_EDMA_Driver_User_Guide.pdf
|   |---AM335x_PSP_U-Boot.pdf
|   |---AM335x_PSP_McSPI_Driver_Guide.pdf
|   |---AM335x_PSP_MM.mm.pp.bb_Feature_Performance_Guide.pdf
|   |---AM335x_PSP_MMC_SD_Support.pdf
|   |---AM335x_USB_User_Guide.pdf
|   |---AM335x_PSP_WDT_Support.pdf
|   |---AM335x_PSP_Flashing_Tools_Guide.pdf
+---host-tools
|   |---nand-flash-writer.out
|   |---mkspd-am335x.sh
+---src
|   |---nandflash-MM.mm.pp.bb.tar.gz
+---images
|   +---examples
|   |   +---am335x
|   |   |   |---edma_test.ko
|   +---kernel
|   |   +---am335x
|   |   |   |---uImage
|   +---u-boot
|   |   |---am335x
|   |   |   |---u-boot.img
|   |   |   |---MLO
+---src
|   +---kernel
|   |   |---ChangeLog-MM.mm.pp.bb
|   |   |---diffstat-MM.mm.pp.bb
|   |   |---kernel-patches-MM.mm.pp.bb.tar.gz
|   |   |---linux-MM.mm.pp.bb.tar.gz
|   |   |---ShortLog
|   |   |---Unified-patch-MM.mm.pp.bb.gz
|   +---u-boot
|   |   |---ChangeLog-MM.mm.pp.bb
|   |   |---diffstat-MM.mm.pp.bb
|   |   |---ShortLog
|   |   |---u-boot-patches-MM.mm.pp.bb.tar.gz
|   |   |---u-boot-MM.mm.pp.bb.tar.gz
|   |   |---Unified-patch-MM.mm.pp.bb.gz
|   |   |---Readme.txt
+---examples
|   |---examples-MM.mm.pp.bb.tar.gz
```

Instructions to extract U-Boot, Linux & Example source code from tar archive

#### NOTE

Below steps assumes that release package is extracted inside directory represented as \$AM335x-PSP-DIR:

## Install U-Boot

- U-Boot source tarball is u-boot-MM.mm.pp.bb.tar.gz needs to be extracted on Linux build host. This will create U-Boot source base for AM335x

```
$ cd $AM335x-PSP-DIR/AM335x-LINUX-PSP-MM.mm.pp.bb/src/u-boot
$ tar -xvfz u-boot-MM.mm.pp.bb.tar.gz
```

## Install Linux Kernel

- Similarly kernel source tarball linux-MM.mm.pp.bb.tar.gz needs to be extracted to have kernel source directory setup for building kernel and/or modules

```
$ cd $AM335x-PSP-DIR/AM335x-LINUX-PSP-MM.mm.pp.bb/src/kernel
$ tar -xvfz linux-MM.mm.pp.bb.tar.gz
```

- Various example sources are included inside \$AM335x-PSP-DIR/examples/examples-MM.mm.pp.bb.tar.gz

## Filesystem

To boot-up Linux, a target filesystem is needed. A filesystem is not included in PSP package. It should be obtained from [SDK release](#).

## Toolchain

GNU toolchain for ARM processors from Arago is recommended. Arago Toolchain can be found in the linux-devkit directory of the SDK [here](#).

## Environment Setup

After installing the toolchain, the environment in the Linux host needs to be setup.

- Set the environment variable PATH to contain the binaries of the Arago cross-compiler tool-chain.

For example, in bash:

```
$ export PATH=/opt/toolchain/arago/arago-2011.09/bin/:$PATH
```

- Add the location of U-Boot tools directory to the PATH environment variable (required for mkimage utility that is built as part of U-Boot build process and is needed to generate ulmage when building the kernel)

For example, in bash:

```
$ export PATH=/opt/u-boot/tools:$PATH
```

#### NOTE

Actual commands to be used for setting the environment variables will depend upon the shell and location of the tools.

#### NOTE

To help get started quickly, the PSP package comes with pre-built binaries. However, after making any changes to U-Boot and/or Linux Kernel, they have to be cross-compiled and the new binaries that are generated should be used.

## Flashing Tools

On AM335x EVM, the Cortex-A8 core boots up first. On boot-up, the ARM core runs the U-Boot image which need to be present in the flash memory of the EVM.

The flash-writers are a part of the PSP package. Instructions on how to use the flash-writers in CCSv4/v5 can be found in [AM335x Flashing Tools Guide](#).

## U-Boot

For details about AM335x U-Boot, refer [AM335x U-Boot User's Guide](#).

## Linux Kernel

For details about AM335x Linux, refer [AMSDK Linux User's Guide](#).

## Booting Linux Kernel

Kernel along with root filesystem can either be booted from on board storage device or can be fetched over the Ethernet or UART to RAM using TFTP or serial protocols like YMODEM and booted from there. Also, the root filesystem can be formatted as JFFS2 and flashed, which, then can be mounted as MTD root block device. Please refer to the U-Boot User Guide for details about flashing and supported storage devices.

Following sections describe various kernel boot options possible.

#### NOTE

The offsets and MTD partition numbers used in examples below may vary depending upon actual partition layout used on particular storage device. Also, selecting multiple storage device support in kernel (e.g., NAND & SPI) may change the effective partition

number to be used as root partition.

## Boot from MMC/SD

Make sure the Boot Mode/Configuration Select Switch is set for the MMC/SD boot mode as described in [SD boot switch settings](#) section.

Power on EVM with SD card inserted and wait for U-Boot prompt of the 2nd stage (U-Boot#) to come up on the serial console.

Setup the SD card as described in [Setting Up Boot Environment on SD Card](#) and When Kernel Image and File system are placed in SD card.

```
U-Boot# setenv bootargs 'console=tty00,115200n8 root=/dev/mmcblk0p2 mem=128M rootwait'
U-Boot# setenv bootcmd 'mmc rescan; fatload mmc 0 0x82000000 uImage; bootm 0x82000000'
```

## Boot from NAND

Make sure the Boot Mode/Configuration Select Switch is set for the NAND boot mode as described in [NAND boot switch settings](#) section.

Power on AM335x EVM and wait for U-Boot prompt of the 2nd stage (U-Boot#) to come up on the serial console.

When kernel ulmage and JFFS2 filesystem are flashed on the NAND device:

```
U-Boot# nand read.i 0x81000000 280000 500000
U-Boot# setenv bootargs 'mem=128M console=tty00,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2 ip=dhcp'
U-Boot# bootm 0x81000000
```

When kernel image is flashed on the NAND device, and NFS mounted filesystem is being used:

```
U-Boot# nand read.i 0x81000000 280000 500000
U-Boot# setenv bootargs 'console=tty00,115200n8 root=/dev/nfs nfsroot=172.24.179.98:/nfs_root,noLOCK rw mem=128M'
U-Boot# bootm 0x81000000
```

## Boot from SPI

Make sure the Boot Mode/Configuration Select Switch is set for the SPI boot mode as described in [SPI boot switch settings](#) section.

Power on AM335x EVM and wait for U-Boot prompt of the 2nd stage (U-Boot#) to come up on the serial console.

Assuming kernel image is flashed on the SPI flash @ 0x62000 and NFS based root filesystem is used:

```
U-Boot# sf read 0x81000000 0x62000 0x200000
U-Boot# setenv bootargs 'console=tty00,115200n8 root=/dev/nfs nfsroot=172.24.179.98:/nfs_root,noLOCK rw mem=128M'
U-Boot# bootm 0x81000000
```

## Boot over Network (Ethernet)

### NOTE

When setting a MAC address please ensure that the LS-bit of the 1st byte is not 1 i.e. when setting the MAC address: **y** in **xy:ab:cd:ef:gh:jk** has to be an even number. For more info this refer to the wiki page [http://en.wikipedia.org/wiki/MAC\\_address](http://en.wikipedia.org/wiki/MAC_address)

When kernel image and ramdisk image are fetched from a TFTP server:

- Ensure that the EVM is connected to network with DHCP and TFTP server set up
- If the TFTP server supports negotiation between client and server, Disable it
- Copy kernel image and ramdisk to TFTP server's root directory.
- Set 'ethaddr' U-Boot environment variable with proper ethernet address in format 'xx:xx:xx:xx:xx:xx' (replace 'xx' with proper hexadecimal values)
- Execute following commands at U-Boot prompt. Assuming kernel image name as 'ulmage' and ramdisk file name as 'ramdisk.gz'

```
U-Boot# setenv autoload no
U-Boot# dhcp
U-Boot# setenv serverip <Server IP Address>
U-Boot# tftp 0x81000000 uImage
U-Boot# tftp 0x82000000 ramdisk.gz
U-Boot# setenv bootargs 'mem=128M console=tty00,115200n8 root=/dev/ram0 initrd=0x82000000,40M ramdisk_size=32768 ip=dhcp'
U-Boot# bootm 0x81000000
```

- Alternatively, kernel can be made to use the same IP address as assigned to U-Boot instead of doing DHCP request again by setting U-Boot parameters as follows:

```
U-Boot# print ethaddr
U-Boot# setenv ethaddr <unique-MAC-address> <-- Check if MAC address is assigned and is unique
U-Boot# setenv bootcmd 'dhcp;run addip; tftp 81000000 uImage;bootm' <-- Set only if not present already, format xn:yy:zz:aa:bb:cc
U-Boot# setenv hostname <unique-hostname>
U-Boot# setenv addip 'setenv bootargs ${bootargs} ip=${ipaddr}:${nfserver}:${gatewayip}:${netmask}:${hostname}:eth0:off'
U-Boot# setenv autoload no
U-Boot# setenv nfserver <nfs-server-ip>
U-Boot# setenv bootargs 'console=tty00,115200n8 root=/dev/nfs nfsroot=<nfs-server-ip>:<path-to-nfs-share>,noLOCK rw mem=128M' <-- Make sure the same NFS server IP is used below
U-Boot# setenv serverip <tftp-server-ip>
U-Boot# saveenv
U-Boot# boot
```

- After saving the environment variables, they need not be setup again on reboot unless a change is required.
- Note that the above example uses NFS mounted root file system accessed over 'eth0' interface

### NOTE

'ethaddr' need not be set for devices having valid MAC IDs set. In such cases, U-Boot will automatically detect and set the ethernet address (should show message like "Detected MACID:...").

## Various Module User's Guide

### GPIO Driver

AM335X has four GPIO modules provides 32 dedicated general-purpose pins with input and output capabilities, total 0 - 127 pins are available for usage.

[GPIO Driver User Guide](#) have more details of driver usage

### DCAN Driver

[DCAN Driver User Guide](#)

### EDMA3 Driver

[AM335x EDMA Driver's Guide](#)

### Audio Driver

The audio driver in the PSP package conforms to the ASoC framework in the Linux kernel. The current driver supports audio capture and playback using the AIC3106 codec on the EVM. For more details on the audio driver refer to the following page: [AM335x Audio Driver's Guide](#)

### USB Driver

The USB subsystem includes two USB (Mentor Graphics USB2.0 OTG) controller.

[AM335X USB Driver User Guide 04.06.00.](#)

### MMC/SD driver

MMC/SD Driver supports MMC/SD/SDHC/uSD cards. HSMMC peripheral (and driver) has support for 4 data lines at the max operating frequency of 48MHz. HSMMC is a slave DMA peripheral and uses EDMA to move data between SD card and system memory.

Please refer [AM335x MMC/SD Driver's Guide](#) for more details.

### McSPI Driver

Please refer AM335x\_PSP\_McSPI\_Driver\_Guide.pdf in the release package or refer online [AM335x McSPI Driver's Guide](#).

### Watchdog Timer (WDT)

The following document covers the details regarding the watchdog timer in AM335x:

Please refer [AM335x PSP WDT Driver User Guide](#).

### CPSW Driver

CPSW (Common Platform Switch) is a ethernet switch consisting of 3 MAC ports.

Please refer [AM335x CPSW \(Ethernet\) Driver's Guide](#) for more details.

### LCD Frame Buffer Driver

LCDC (LCD controller) is updated version of LCDC found on DA850.

Please refer [AM335x LCD Controller Driver's Guide](#) for more details.

### Touchscreen Driver

The touchscreen controller is an 8 channel general purpose ADC, with optional support for touchscreen conversions for a 4/5/8-wire resistive panel.

Please refer [AM335x Touchscreen Driver's Guide](#) for more details.

### Power Management

Refer to [AM335x Power Management User guide](#) for more details.

### NAND Driver

Please refer [AM335x NAND Driver's Guide](#) for more details.

### PWM Driver

Please refer [AM335x PWM Driver's Guide](#) for more details.

### EVM On-Board Components Drivers

AM335x EVM supports components like temperature sensor and ambient light sensor. Usage information on drivers for these components

is available in [EVM On-Board Components Drivers Guide](#)

## Modifying Pin Mux settings

On AM335x devices, the pins are tri-stated and set to Mode 0 or any other mode as required for specific module (e.g., MMC) in U-Boot. If a particular pin needs to be used for any other function than Mode 0 or override any other pin mode which was already set in U-Boot, there are 3 methods to do it.

### Method I - Modify Linux Kernel, rebuilt and use

Default mux mode can be changed by adding specific mux entry in the beginning of `board_mux` array in `arch/arm/mach-omap2/board-am335xevm.c` or calling `omap_mux_init_signal()` during initialization (e.g., in device specific initialization function called from `omap2_init_devices()` in `arch/arm/mach-omap2/devices.c`).

The names of multiplexed signals are specified in `arch/arm/mach-omap2/mux33xx.c` file in kernel source directory.

e.g., for setting `xref_clk0` pin (mode 0) to `usb1_drvvbus` (mode 7 or FUNCTION 8), add

```
AM33XX_MUX(XREF_CLK0, OMAP_MUX_MODE7)
```

to `board_mux` structure in board file or call

```
omap_mux_init_signal("xref_clk0.usb1_drvvbus", 0)
```

#### NOTE

The string passed above should be `mode0_name.desired_mode_name` format.

### Method II - Modify pin-mux from arguments passed to Kernel

The above API approach is useful if pin-mux needs to setup at run time depending upon the board/hardware detected without need of maintaining separate kernel binaries. Alternatively, particular pins can be setup by passing respective pinmux details to kernel command line in following format

```
omap_mux=<mode0_name>.<signal_name>=<value>,<mode0_name>.<signal_name>=<value>
```

E.g., to set `mmc1_cmd_mux0` (mode 0) pin to `gpio0_0` and enable pull down, append following to kernel command line passed from the boot loader:

```
omap_mux=mmc1_cmd_mux0.gpio0_0=0
```

For pull up (set bit 17):

```
omap_mux=mmc1_cmd_mux0.gpio0_0=0x20000
```

configuration for multiple pins can be passed which are separated by comma.

For details about this boot parameter, refer `Documentation/kernel-parameters.txt` in kernel source directory.

### Method III - Modify pin-mux from Linux Console

Pin-mux can also be setup from kernel console. This requires debugfs support.

1. Make sure kernel is built with debugfs support Configure the kernel to enable debugfs:

```
Symbol: DEBUG_FS [=y]
Type : boolean
Prompt: Debug Filesystem
Defined at lib/Kconfig.debug:89
Location:
-> Kernel hacking
```

2. Boot the target hardware and mount debugfs:

```
mkdir -p /debugfs
mount -t debugfs debugfs /debugfs
cd /debugfs/omap_mux/
```

3. Assume `spi0_d0` pins needs to be set

View current pin-mux setting

```
root@arago-armv7:/debugfs/omap_mux# cat spi0_d0
name: spi0_d0.gpio0_3 (0x44e10954/0x954 = 0x002f), b NA, t NA
mode: OMAP_MUX_MODE7 | AM33XX_PIN_INPUT
signals: spi0_d0 | uart2_txd | i2c2_scl | NA | NA | NA | NA | NA | gpio0_3
```

Change pin-mux value (0x20 is an example for MODE 0, Input mode, PULL Enabled with PULL-DOWN mode,

```
root@arago-armv7:/debugfs/omap_mux# echo 0x20 > spi0_d0
```

View current pin-mux setting

```
root@arago-armv7:/debugfs/omap_mux# cat spi0_d0
name: spi0_d0.gpio0_3 (0x44e10954/0x954 = 0x0020), b NA, t NA
```

```
mode: OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN
signals: spi0_d0 | uart2_txd | i2c2_scl | NA | NA | NA | NA | gpio0_3
```

Another example (0x31 is an example for MODE 1, Input mode, PULL Enabled with PULL-UP mode,

```
root@arago-armv7:/debugfs/omap_mux# echo 0x31 > spi0_d0
```

View current pin-mux setting

```
root@arago-armv7:/debugfs/omap_mux# cat spi0_d0
name: spi0_d0.uart2_txd (0x44e10954/0x954 = 0x0031), b NA, t NA
mode: OMAP_MUX_MODE1 | AM33XX_PIN_INPUT_PULLUP
signals: spi0_d0 | uart2_txd | i2c2_scl | NA | NA | NA | NA | gpio0_3
```

## Clock Management details

### Clock tree from debugfs

Details of clocks such as rate, usecount and flags can be viewed through debugfs entries of the clocks. To access these details one has to mount the debugfs root directory first, follow the steps below to mount debugfs and view clock details:

- Build Kernel with debugfs support (default is with debugfs support), (refer [Modifying Kernel configuration section](#))

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- menuconfig
Select Power management options -> Power Management Debug Support
```

Ensure that Kernel hacking -> Debug Filesystem is selected in Kernel configuration (by default it is selected)

- Mount the debugfs filesystem

```
$ mount -t debugfs none /sys/kernel/debug
```

- Change to clock directory under debugfs, clock entries arranged exactly same as clock tree, ie. root clock -> child clock & sibling clocks, to see the details,

example 1:

```
$ cd /sys/kernel/debug/clock
root@arago-armv7:/sys/kernel/debug/clock# ls -l
drwxr-xr-x  5 root  root      0 Jan  1 1970 clk_32768_ck
drwxr-xr-x  6 root  root      0 Jan  1 1970 clk_rc32k_ck
drwxr-xr-x  2 root  root      0 Jan  1 1970 ehprwm0_tbc1k
drwxr-xr-x  2 root  root      0 Jan  1 1970 ehprwm1_tbc1k
drwxr-xr-x  2 root  root      0 Jan  1 1970 ehprwm2_tbc1k
-r--r--r--  1 root  root      0 Jan  1 1970 summary
drwxr-xr-x  2 root  root      0 Jan  1 1970 tclkin_ck
drwxr-xr-x  2 root  root      0 Jan  1 1970 virt_19_2m_ck
drwxr-xr-x  3 root  root      0 Jan  1 1970 virt_24m_ck
drwxr-xr-x  2 root  root      0 Jan  1 1970 virt_25m_ck
drwxr-xr-x  2 root  root      0 Jan  1 1970 virt_26m_ck
root@arago-armv7:/sys/kernel/debug/clock#
```

example 2:

```
root@arago-armv7:~# cd /sys/kernel/debug/clock/
root@arago-armv7:/sys/kernel/debug/clock# ls
clk_32768_ck ehprwm1_tbc1k tclkin_ck virt_25m_ck
clk_rc32k_ck ehprwm2_tbc1k virt_19_2m_ck virt_26m_ck
ehprwm0_tbc1k summary virt_24m_ck
root@arago-armv7:/sys/kernel/debug/clock# cd clk_32768_ck/
root@arago-armv7:/sys/kernel/debug/clock/clk_32768_ck# ls
flags rtc_fck timer1_fck
rate sysclkout_pre_ck usecount
root@arago-armv7:/sys/kernel/debug/clock/clk_32768_ck# cat rate
32768
root@arago-armv7:/sys/kernel/debug/clock/clk_32768_ck# cat usecount
3
root@arago-armv7:/sys/kernel/debug/clock/clk_32768_ck#
```

### Parent change

Currently changing a parent clock is not supported through clk, this can be done from uboot command prompt as below:

- Load uboot, wait for 1st stage to complete.
- Interrupt the autoboot of 2nd stage (countdown from 3-0).
- Find the clocksel register corresponding to the clk whose parent needs to be changed.
- write the value corresponding to new parent to the register

example:

To write to a reg:

```
# mw 0x481c50c4 0x3
```

To read from a reg:

```
# md 0x481c50c4 1
```

for help:

```
# help
```

## Set rate support

Some clocks support changing clock rate dynamically at runtime, clock rate of a clock may be changed only if:

- The clock is not used by any module or none of its child clock is in use.i.e. `clk->usecount` is zero.
- When setting the rate of a leaf clock, parent is not in use i.e.`parent->usecount` is zero.
- Changing the clock rate will not cause system instability.

To change a clock's rate:

- 1.Get the pointer to `clk` struct by,

```
clk = clk_get(dev_id,clk_name);
```

2. Set the new rate,

```
if(clk->set_rate)
    ret = clk->set_rate(clk);
```



For technical support please post your questions at <http://e2e.ti.com>. Please post only comments about the article **AM335x PSP User's Guide** here.

### Links



[ARM Microcontroller MCU](#)

[ARM Processor](#)

[Digital Media Processor](#)

[Digital Signal Processing](#)

[Microcontroller MCU](#)

[Multi Core Processor](#)

[Ultra Low Power DSP](#)

[8 bit Microcontroller MCU](#)

[16 bit Microcontroller MCU](#)

[32 bit Microcontroller MCU](#)

Categories: [AM335x](#) | [AM35x](#) | [AM37x](#) | [AM1x](#) | [AM18x](#) | [Linux](#) | [PSP](#)

[Leave a Comment](#)

This page was last modified on 25 July 2012, at 01:09.

This page has been accessed 15,416 times.

Content is available under [Creative Commons Attribution-Share Alike 3.0 license](#).

[Privacy policy](#) [About Texas Instruments Embedded Processors Wiki](#) [Disclaimers](#)

