

Home
Embedded Systems
Embedded Tutorials
STM32 ARM
Microchip PIC
Articles
Embedded Projects
IoT
ESP32
Electronics
Electronics Tutorials
Electronics Projects
Courses
Shop
Blog
About



DeepBlue

FOLLOW:



ESP32 / IoT

MORE

SEARCH THE BLOG

Search ...

Search

CATEGORIES

Select Category

PCBWay HIGH-QUALITY PCB
ONLY \$5 FOR 10 PIECES
• Rogers, HDI, aluminum and rigid-flex PCB are available now
• Production time 24 hours

PCB ASSEMBLY
Free shipping + Free stencil
ONLY \$30
• Component sourcing
• Quality assurance

ADVERTISING

JLCPCB
1-4 Layer PCBs \$2
SMT Assembly Fee \$0
Get Coupons

ESP32 Change CPU Speed (Clock Frequency)

Home
Embedded Systems
Embedded Tutorials
STM32 ARM
Microchip PIC
Articles
Embedded Projects
IoT
ESP32
Electronics
Electronics Tutorials
Electronics Projects
Courses
Shop
Blog
About



important in future work in this series of tutorials.

We'll be using this in power consumption control, enhancing RTOS tasks execution time, maybe getting a higher ADC sampling rate, and much more possibilities. If not for that, let it just be for the sake of getting the most out of the CPU. Without further ado, let's get right into it!

ESP32 External Interrupts Pins in
Arduino – GPIO Interrupt
Examples

X

CATEGORIES

- Electronics
 - Electronics Projects
 - Electronics Tutorials

- Embedded Systems
 - Articles
 - Embedded Projects
 - Embedded Tutorials
 - Microchip PIC
 - STM32 ARM

ESM

FAQs

FPGA

IoT

ESP32

Projects Ideas

Tech Reviews

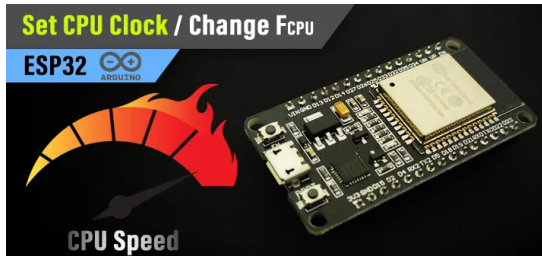


eZOIC

report this ad

Set CPU Clock / Change F_{cpu}

ESP32



Tutorial Contents [\[show\]](#)



Work in Chrome

Google

LEARN MORE

Requirements For This Tutorial

Prior Knowledge

- Nothing

Software Tools

- [Arduino IDE For ESP32 \(Setup Guide\)](#)

Hardware Components

You can either get the complete course kit for this series of tutorials using the link down below. Or just refer to the table for the exact components to be used in practical LABs for only this specific

ESP32 CPU Speed (Frequency)

eZOIC

report this ad



As we've stated earlier in this series of tutorials, the ESP32 is a dual-core system with two Harvard Architecture Xtensa LX6 CPUs. All embedded memory, external memory, and peripherals are located on the data bus and/or the instruction bus of these CPUs.

CPU Clock

Both CPUs can be clocked from various sources internally or externally. The most important circuitry is the PLL (Phase-Locked Loop) which multiplies the input clock frequency (whether it's an external crystal or internal oscillator). This results in a much higher frequency signal that's derived from the main clock source (orders of magnitude in terms of frequency).

The PLL_CLK is an internal PLL clock signal with a frequency of 320 MHz or 480 MHz. Note that this PLL clock is divided /2 or /4 before it's fed to any CPU. The table down below from the datasheet shows the possible configurations for the clock.

PLL_CLK (320 MHz)	1	0	CPU_CLK = PLL_CLK / 4 CPU_CLK frequency is 80 MHz
PLL_CLK (320 MHz)	1	1	CPU_CLK = PLL_CLK / 2 CPU_CLK frequency is 160 MHz
PLL_CLK (480 MHz)	1	2	CPU_CLK = PLL_CLK / 2 CPU_CLK frequency is 240 MHz

Peripherals Clock

The APB bus clock (APB_CLK) is derived from CPU_CLK, the division factor depends on the CPU_CLK source as shown in the table down below. The peripherals clock signals are APB_CLK, REF_TICK, LEDC_SCLK, APLL_CLK, and PLL_D2_CLK.

CPU_CLK Source	APB_CLK
PLL_CLK	80 MHz
APLL_CLK	CPU_CLK / 2

With that being said, we can conclude that setting the PLL_CLK as a clock source will help us choose the 480MHz PLL clock output as a clock source for the CPU (the CPU_CLK will be 480/2 MHz). While the APB bus will be clocked @ 80MHz due to this option being selected.

Should we be concerned about the peripherals clock and APB_CLK? Of course, especially if you're trying to optimize for power consumption or maximizing the performance. You need to know & control this parameter. The bus clock rate dictates the speed at which you can access control registers, read a peripheral, GPIO pin states, etc. If it's a bottle-neck, then a change has to be made indeed.

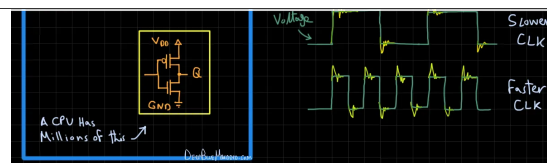
Why Would You Need To Change ESP32 CPU Clock?

You should consider changing the clock rate for your microcontroller, in general, under certain circumstances. Number one being the power consumption optimization. And then, we can say enhancing the performance of the CPU to meet certain timing requirements, boosting a specific peripheral, etc.



Power Consumption Optimization

Generally speaking, it's a rule of thumb, all digital logic circuits consume more power the higher the clock rate is. The faster it goes, the more power it draws. Here is a simplified (approximation) for



We can divide the reasoning behind this into two pieces: Static Power Consumption & Dynamic

Power Consumption. The static power consumption is due to the manufacturing of the transistors being very small in size and there is uncontrollable leakage current even without being switched ON and OFF. That's the static part.

The dynamic power consumption, on the other hand, is what happens while switching the gates as shown in the diagram above. Which has its own reasons as well. First of which is the gates' capacitance, which requires a high surge of current to switch ON and OFF so quickly that we keep following the clock signal without being distorted.

And also the transistor technology has the inherent feature of not being completely turned OFF or ON at each edge of the clock. This means we have a dead-short between Vdd & GND for a very short period of time. With both transistors conducting, called "Shoot-through" current spikes.

Therefore, the more clock edges you have in a finite period of time, the more current spikes there are. In other words, the higher the clock frequency, the higher power drawn by the CPU. ($P = I \times V$)

Enhancing The Performance

The other, more obvious, the reason for willing to change the clock rate is to enhance the performance of the ESP32 in terms of computational power. To achieve some required timing behavior or any other application-dependent reasoning. Let's now move to how we can change the ESP32 CPU clock frequency using the Arduino Core API functions.

ESP32 Change CPU Speed (in Arduino)

LEARN MORE

This is the function we'll be using to set the ESP32 CPU clock frequency.

```
1. //function takes the following frequencies as v
2. // 240, 160, 80   <<< For all XTAL types
3. // 40, 20, 10    <<< For 40MHz XTAL
4. // 26, 13        <<< For 26MHz XTAL
5. // 24, 12        <<< For 24MHz XTAL
6. bool setCpuFrequencyMhz(uint32_t cpu_freq_mhz);
```

As we can see in the documentation of this function, it can expect 80, 160, or 240 as valid inputs as well.

In case you're willing to use a lower CPU clock for power consumption optimization, you'll need first to check the XTAL crystal frequency using the function down below. Then, use the corresponding valid options shown in the comment section in the code snippet above.

```
1. uint32_t getXtalFrequencyMhz(); // In MHz
```


After making a change to the CPU clock frequency, you can double-check the CPU clock rate value by reading the following function's return value.

```
1. uint32_t getCpuFrequencyMhz(); // In MHz
```


To read the APB bus clock rate, use this function.

```
1. uint32_t getApbFrequency(); // In Hz
```

And now, let's make some tests!



Work in Chrome



LEARN MORE

Components For This Tutorial's LABs

QTY.	Component Name	Buy Links
ESP32 Devkit v1 DOIT		
1	Board	Amazon.com - eBay.com -
	or Any Other ESP32 Dev Board	Banggood.com
2	BreadBoard	Amazon.com - eBay.com - Banggood.com
1	Jumper Wires Pack	Amazon.com / Amazon.com - eBay.com / eBay.com - Banggood.com
1	Micro USB Cable	Amazon.com - eBay.com - Banggood.com

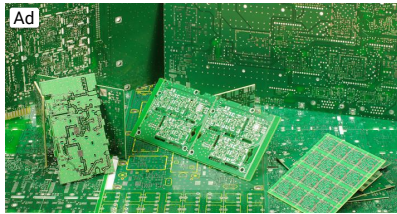
*Affiliate Disclosure: When you click on links in this section and make a purchase, this can result in this site earning a commission. Affiliate programs and affiliations include, but are not limited to, the eBay Partner Network (EPN) and Amazon.com, Banggood.com. This may be one of the ways to support this free platform while getting your regular electronic parts orders as usual at no extra cost to you.

ESP32 Read Default CPU Clock – LAB



LAB Number 7


LAB Name Based ESP32 CPU Default Clock Data



Ad

RF Synthesizer Modules

The HSM Series offers Output Power Dynamic Range of +20dBm to -60dBm.

 holzworth.com

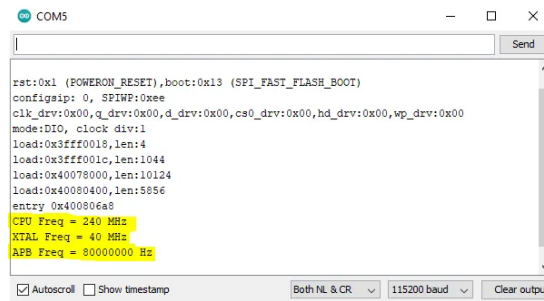
VISIT SITE



```
1.  /*
2.  * LAB: 7
3.  * Name: ESP32 Read Default Clocks
4.  * Author: Khaled Magdy
5.  * For More Info Visit: www.DeepBlueMbedded.com
6.  */
7.
8.  #define GPIO_pin 5
9.
10. uint32_t Freq = 0;
11.
12. void setup()
13. {
14.     pinMode(GPIO_pin, OUTPUT);
15.     Serial.begin(115200);
16.     Freq = getCpuFrequencyMhz();
17.     Serial.print("CPU Freq = ");
18.     Serial.print(Freq);
19.     Serial.println(" MHz");
20.     Freq = getXtalFrequencyMhz();
21.     Serial.print("XTAL Freq = ");
22.     Serial.print(Freq);
23.     Serial.println(" MHz");
24.     Freq = getApbFrequency();
25.     Serial.print("APB Freq = ");
26.     Serial.print(Freq);
27.     Serial.println(" Hz");
28. }
29.
30. void loop()
```



Screenshot For The Result



```
COM5
Send

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config:spi: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
CPU Freq = 240 MHz
XTAL Freq = 40 MHz
APB Freq = 80000000 Hz

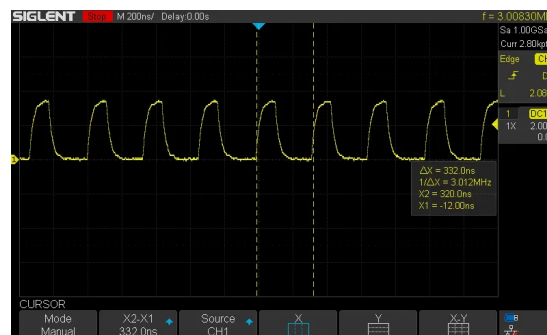
[Autoscroll] [Show timestamp] [Both NL & CR] [115200 baud] [Clear output]
```

Well, I had no idea about the XTAL crystal frequency, it turned out to be 40MHz, that's OK. The CPU turned out to be clocked at the Max rate by default (which is 240MHz). And the APB frequency is 80MHz as expected, Remember this table?

When the CPU is being Clocked from the PLL_CLK, the APB bus will have an 80MHz clock rate. Which we've already validated in the results above.

CPU_CLK Source	APB_CLK
PLL_CLK	80 MHz
APLL_CLK	CPU_CLK / 2
XTAL_CLK	CPU_CLK
RTC8M_CLK	CPU_CLK

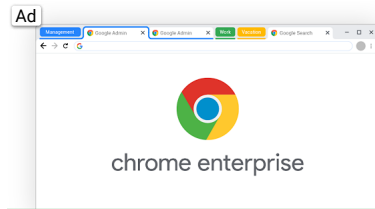
Let's now see how fast the GPIO pin is being switched ON & OFF on my DSO.



It turned out to be 3MHz as you can see. It's not a good-looking square wave due to the drive capability of the GPIO pin output driver and also maybe due to propping attenuation for High-frequency components. All in all, we have got an estimate for this parameter.

ESP32 Change CPU Speed (Frequency) – LAB

LAB Name **ESP32 CPU Frequency Change (For Power Consumption)**



Work in Chrome

Move faster with Chrome sync and cross-platform compatibility.



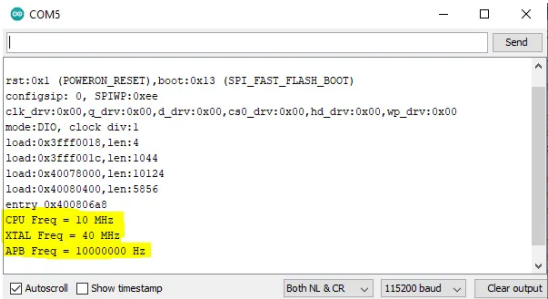
LEARN MORE

```
1. //function takes the following frequencies as v
2. // 240, 160, 80   <<< For all XTAL types
3. // 40, 20, 10    <<< For 40MHz XTAL
4. // 26, 13       <<< For 26MHz XTAL
5. // 24, 12       <<< For 24MHz XTAL
6. bool setCpuFrequencyMhz(uint32_t cpu_freq_mhz);
```

```
1. /*
2.  * LAB: 8
3.  * Name: ESP32 Read Default Clocks
4.  * Author: Khaled Magdy
5.  * For More Info Visit: www.DeepBlueMbedded.com
6.  */
7.
8. #define GPIO_pin 5
9.
10. uint32_t Freq = 0;
11.
12. void setup()
13. {
14.     pinMode(GPIO_pin, OUTPUT);
15.     Serial.begin(115200);
16.     setCpuFrequencyMhz(10);
17.     Freq = getApbFrequency();
```

```
24.     Serial.println(" MHz");
25.     Freq = getApbFrequency();
26.     Serial.print("APB Freq = ");
27.     Serial.print(Freq);
28.     Serial.println(" Hz");
29. }
30.
31. void loop()
32. {
33.     digitalWrite(GPIO_pin, 1);
34.     digitalWrite(GPIO_pin, 0);
35. }
```

Screenshot For The Result



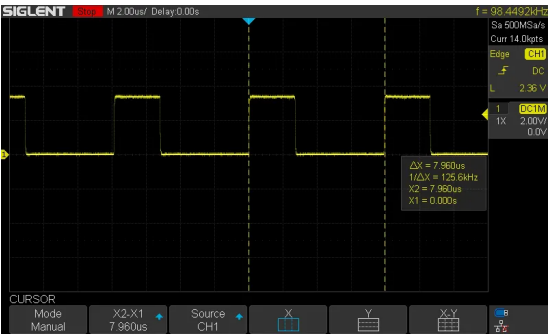
The CPU turned is now clocked @ 10MHz. And the APB frequency is 10MHz as well. Recall this table?

When the CPU is being Clocked

from the XTAL_CLK, the APB bus will have the same clock rate as the CPU. Which we've already validated in the results above.

CPU_CLK Source	APB_CLK
PLL_CLK	80 MHz
APLL_CLK	CPU_CLK / 2
XTAL_CLK	CPU_CLK
RTC8M_CLK	CPU_CLK

Let's now see how fast the GPIO pin is being switched ON & OFF on my DSO.



It turned out to be 0.125MHz (or 125KHz) as you can see. This was nearly expected because in the previous test the CPU clock rate was 240MHz and the max GPIO switching rate was 3MHz. Now, the CPU clock is reduced to 10MHz, so the max GPIO pin switching is expected to fall down to 1/24th of its previous value.

Theoretically, it should have become $(1/24) \times 3 =$

has its role in deciding on the peripherals max data rate as well.

Concluding Remarks

I know the GPIO pin toggling thing maybe a little bit wired way of showing how bus clock rate affects the performance of all peripherals. It just applies to all other peripherals on the bus not only GPIO pins. That's the point of doing this in the first place!

ESP32 CPU clock control is extremely important in many applications, and I hope it's more clear to all of you by the end of this tutorial. I'll keep updating this series of tutorials by adding more applications and techniques that may help you in your projects. Drop me a comment if you've got any questions or suggestions, I'll be glad to help!

You can also check the [ESP32 Course Home Page](#) for more ESP32 tutorials divided into sections based on categories. This may be helpful for you in case of searching for a specific tutorial or application.

Did you find this helpful? If yes, please consider [supporting this work](#) and sharing these tutorials!

Stay tuned for the upcoming tutorials and don't forget to **SHARE** these tutorials. And consider **SUPPORTING** this work to keep publishing free content just like this!

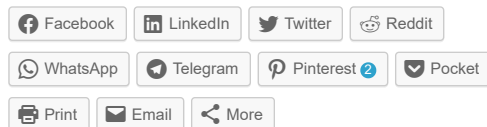
[ESP32 Course Home Page](#)

[Previous Tutorial](#)

Tutorial 4

[Next Tutorial](#)

Share this:



Like this:

Loading...

Related



Getting Started With
ESP32 Programming
Tutorials
April 20, 2021
In "ESP32"



ESP32 External
Interrupts Pins in
Arduino - GPIO
Interrupt Examples
April 29, 2021
In "ESP32"



ESP32 PWM Tutorial &
Examples
(AnalogWrite) -
Arduino
April 25, 2021
In "ESP32"

Share This Page With Your Network!



Tags: ESP32 Arduino



Khaled Magdy

I'm an embedded systems engineer doing both Software & Hardware. I'm an EE guy who studied Computer Engineering, But I'm also passionate about Computer Science. I love reading, writing, creating projects and Technical training. A reader by day a writer by night, it's my lifestyle. You can view my profile or follow me via contacts.

YOU MAY ALSO LIKE...

Installing ESP32
in Arduino IDE –
Step By Step
Guide
APRIL 20, 2021

ESP32 External
Interrupts Pins in
Arduino – GPIO
Interrupt
Examples
APRIL 29, 2021

ESP32 ADC
Tutorial – Read
Analog Voltage in
Arduino
APRIL 27, 2021

1 RESPONSE

Comments 1 Pingbacks 0

Henry V. May 13, 2021 at 12:27 PM

Oh, thanks so much for this detailed explanation!
YOU're the BEST, Keep going

[Reply.](#)

LEAVE A REPLY

[report this ad](#)

Navigate

CATEGORIES

Select Category



— [Home](#)

— [About](#)

— [Contact](#)

— [Blog](#)

SEARCH THE BLOG

Search ...

Search

Resources

[STM32 ARM MCUs Programming Course](#)

[Embedded Systems – PIC Course](#)

[DeepBlue Patreon Page](#)

[PayPal Donation](#)

[Books Recommendation List](#)

Disclosure

DeepBlueMbedded.com is a participant in the Amazon Services LLC Associates Program, an affiliate advertising program designed to provide a means for sites to earn advertising fees by advertising and linking to Amazon.com You can also check my Full [Disclaimer](#) Page For More Information.

Legal Notes

[Disclaimer](#)

[Privacy Policy](#)

[Trademark Info](#)

