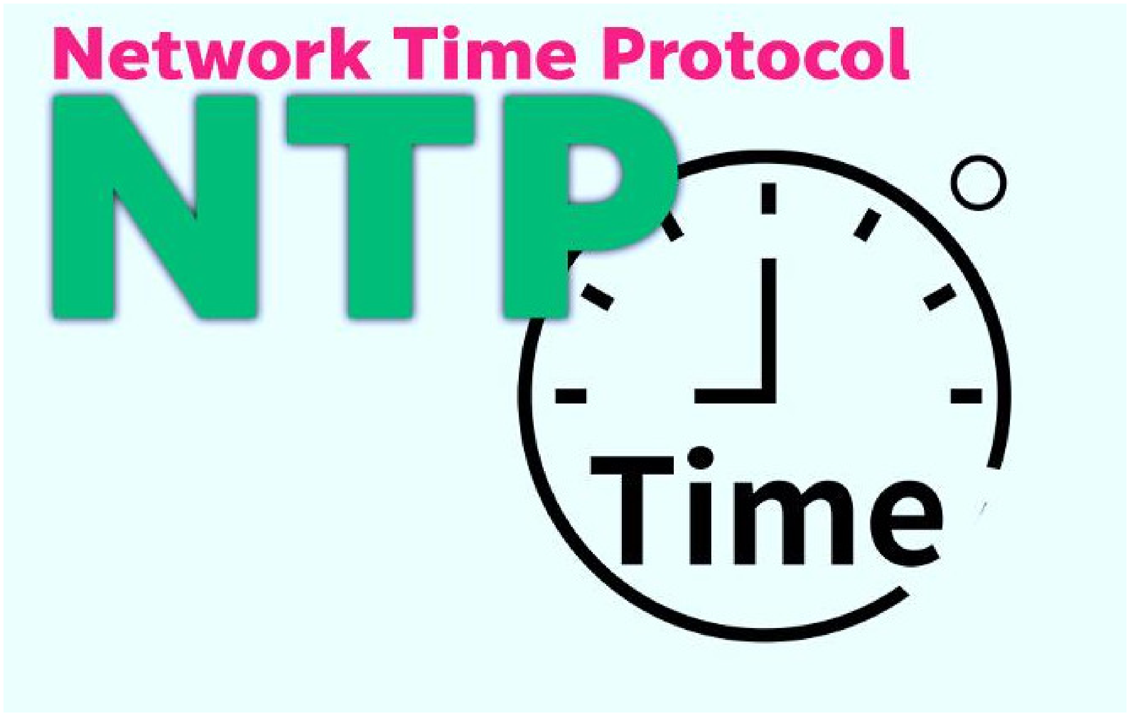
Pico-NTP-Module



Firmware Version 4.00
User Guide
Updated May 18th, 2025

Add-on C-language module to merge with your existing (C-language) project, to retrieve current time from a Network Time Protocol (NTP) server.

IMPORTANT :

This User Guide is about Pico-NTP-Module Firmware Version 4.00 from Andre St. Louys. To get the full potential of this module, you must NOT simply clone the GitHub repository. Read this User Guide and carefully follow the indicated procedure.

Join our Pico-NTP-Module discussion group on:
https://github.com/astlouys/Pico-NTP-Module/discussions

Index

Introduction

The Pico-NTP-Module is a C-Language add-on module that will allow your existing C-language Pico program or project to retrieve current local time from a Network Time Protocol (NTP) server over the Internet. Since the Pico microcontroller has an integrated real-time clock, you'll be able to keep track of the time to display a clock and / or to take specific actions based on the date and / or time.

To help you figure out the details on how to use the Pico-NTP-Module, an example is included in the repository: « Pico-NTP-Example ». This is a basic C-Language program making use of the Pico-NTP-Module. This simple program is also described later in this User Guide.

To get the full potential of the Pico-NTP-Module, make sure you carefully follow the instructions given in this User Guide.

1) The « Setup part 1 » covers the steps required to install the Pico-NTP-Example to your system. Make sure everything is installed as required and that you can build and run the program without problem. You can them go on with the following step to merge the add-on module to your own program.

2) The « Setup part 2 » covers the steps to follow when you want to add the Pico-NTP-Module to one of your own existing C-Language program.

NOTE: « Pico » is used throughout this User Guide to describe the microcontroller used but of course, since we are using the Internet to retrieve current time, a PicoW is required (the « plain Pico » does not have the circuitry to allow Wi-Fi communications). I didn't have the chance to work with the Pico2W so far, but I assume there should not be major changes, if not for the CMakeLists.txt where I assume you must specify which microcontroller is used and which cores. I'm interested in hearing from your experience in adapting this Firmware on the new Pico2W...

You can join other users of the Pico-NTP-Module on the discussion group on:
https://github.com/astlouys/Pico-NTP-Module/discussions
I can also be reached with my personal email at:
astlouys@gmail.com

Pico-NTP-Module setup

Setup part 1: Installing / copying the Pico-NTP-Module to your system.

NOTE: The instructions below assume that your development system is a Raspberry Pi running the Raspberry Pi O.S. (previously called "Raspbian") and that you're using Visual Studio Code. It also assumes that you followed the recommended naming and structure conventions for the directory structure on your development system. If this is not the case, you will have to adapt the instructions accordingly.

1) First of all, since we must access the Internet to retrieve local time from a Network Time Protocol server, you must install the Pico-WiFi-Module on your system. Carefully follow the instructions provided in the Pico-WiFi-Module User Guide in my repository. Once your went through all the steps of « Setup part 1 » and « Setup part 2 » of the Pico-WiFi-Module User Guide, go on with the steps below.

2) Create the project directory on your system:
   ```
   mkdir  /home/pi/pico/Pico-NTP-Module
   ```

3) Clone / copy all files from the GitHub repository to the « /home/pi/pico/Pico-NTP-Module » project directory created in step 1 above.

4) While in the « /home/pi/pico/Pico-NTP-Module » project directory create a symbolic link pointing to the pico_sdk_import.cmake file. This way, if / when you update the Pico SDK, your project will always use the latest version when it is built:
   ```
   ln  -s  /home/pi/pico/pico-sdk/external/pico_sdk_import.cmake
   ```

5) It is assumed that you created a « UTILITIES » directory in the pico directory as indicated in the Pico-WiFi-Module User Guide. The file « baseline.h » from the Pico-WiFi-Module project directory should have already be moved to this directory. As mentioned in the Pico-WiFi-Module User Guide, the baseline.h file contains basic definitions that I almost always use in most projects. You may want to take a look at it and add your own pieces of code that you use often. When you update the « baseline.h » file in the UTILITIES directory, it is automatically updated in all your projects through the symbolic link.

6) Create a symbolic link from the project directory to the « baseline.h » file:
   ```
   ln  -s  /home/pi/pico/UTILITIES/baseline.h
   ```

7) This completes the first part of the setup. At this point, make sure you are able to build and use the Pico-NTP-Example application. Refer to the section of this User Guide on how to use it.

Setup part 2: Integrating the Pico-NTP-Module to one of your projects.

NOTE: The instructions below assume that you development system is a Raspberry Pi running the Raspberry Pi O.S. (previously called « Raspbian ») and that you're using Visual Studio Code. It also assumes that you followed the recommended naming and structure conventions for the directory structure on your development system. If this is not the case, you will have to adapt the instructions accordingly.

Make sure you carefully followed the instructions in the « Setup part 1 » above, since they are a prerequisite for this section.

1) From your project directory: « /home/pi/pico/Pico-My-Project », create a symbolic link pointing to both the « .c » and « .h » files of the Pico-NTP-Module:
```
ln  -s  /home/pi/pico/Pico-NTP-Module/Pico-NTP-Module.c
ln  -s  /home/pi/pico/Pico-NTP-Module/Pico-NTP-Module.h
```

2) From your project directory: « /home/pi/pico/Pico-My-Project », create a symbolic link pointing to the « baseline.h » file:
```
ln  -s  /home/pi/pico/UTILITIES/baseline.h
```

3) In your project file: « /home/pi/pico/Pico-My-Project.c », add the include file « Pico-NTP-Module.h » file:
```
#include "Pico-NTP-Module.h"
```

4) In your project file: « /home/pi/pico/Pico-My-Project.c », declare the structure defined in the Pico-NTP-Module.h file:
```
struct  struct_ntp  StructNTP;
```

5) You have to set a few definitions in the « Pico-NTP-Module.h » file as indicated in the next step.

6) The language must be specified in the « #define FIRMWARE_LANGUAGE ». Languages are specified on the lines above the #define, but be aware that only French and English are currently supported. You should not too much problem adding a new language by looking at the « ntp-lang-french.h » file.

7) You also need to adjust a few « #define » at the top of your C-Language program while referring to the « Pico-NTP-Example.c » program as indicated in the next steps below...

8) You must specify your own country in the « #define WIFI_COUNTRY ». See Appendix A in this User Guide to see what are the available definitions. Be aware that the definition you select may have an impact on the frequencies used by the Wi-Fi circuit on the Pico.

Selecting the bad country may restrict the bandwidth match between your Pico and your Wi-Fi router.

9) You must also set the « #define DST_COUNTRY » corresponding to your country. This time, this definition is used to set the rules used in your country to know when to toggle from « summer time » to « winter time » and vice-versa. See Appendix B to select the write identification for your country.

10) The « #define DELTA_TIME » specifies the time difference (in minutes) between your local time and UTC time and this, when you are in « normal » or « winter time ». For example, since I live in Quebec, Canada, my local time (in winter) is -5 hours from UTC time. So, the delta time for me will be « -300 » (-5 hours X 60 minutes = -300 minutes).

11) Depending on your program, you may also want to copy-past other parts / functions of « Pico-NTP-Example.c » to your own program to use them like « display_human_time() », « get_pico_unique_id() », « input_string() », « log_info() »... Those functions are used in « Pico-NTP-Module », so either you copy-paste them to your own program, either you get rid of / replace them in the Pico-NTP-Module.

Pico-NTP-Example

The « Pico-NTP-Example » application doesn't do much... Its main purpose is to show how to integrate the « Pico-NTP-Module » to your current project. Nonetheless, here is how to use it.

Build the Pico-NTP-Example application and download it to your Pico. The application will automatically be launched when uploaded to the Pico.

Once started, the application gives you some time to start a terminal emulator program and make a connection with the Pico. Since the purpose of the application is to synchronize the Pico with a NTP server over the Internet and display the time on the terminal, it is not very useful if you do not connect such a terminal.

Once this is done, you will see a few messages over the screen and should be able to follow what is going on.

The Firmware finally ends up in an endless loop updating current time on the terminal screen. If you have another type of display, you may use it to make your real-time clock (small LED or LCD).

If your terminal setting converts the « CR » (carriage return) to « CR-LF » (carriage return / line feed) to follow the Unix convention, you will have to revert to a direct translate from « CR » to « CR » for the time clock to update itself on the same terminal line.

When the time is being displayed, pressing « ESC » will toggle the Pico into upload mode, ready to receive a new Firmware version, if you plan to work on the code yourself.

Please not that I kept this program simple since its main purpose it to show how to integrate the Pico-NTP-Module to another program / project. Consequently, there is no provision to automatically update the clock when reaching the toggling from summer time to winter time and vice-versa. If you want to implement this feature, you may refer the the Pico-Green-Clock or Pico-RGB-Matrix in my other repositories. They do have this automatic clock change. That being said, Version 3.00 of the Pico-RGB-Matrix (to be published soon) will have an improved, simplified and optimized algorithm for this feature. You may want to wait until this Version 3.00 is available and take a look at it.

Appendix A – CYW43 country definitions

```
#define CYW43_COUNTRY_WORLDWIDE         CYW43_COUNTRY('X', 'X', 0)

#define CYW43_COUNTRY_AUSTRALIA         CYW43_COUNTRY('A', 'U', 0)
#define CYW43_COUNTRY_AUSTRIA           CYW43_COUNTRY('A', 'T', 0)
#define CYW43_COUNTRY_BELGIUM           CYW43_COUNTRY('B', 'E', 0)
#define CYW43_COUNTRY_BRAZIL            CYW43_COUNTRY('B', 'R', 0)
#define CYW43_COUNTRY_CANADA            CYW43_COUNTRY('C', 'A', 0)
#define CYW43_COUNTRY_CHILE             CYW43_COUNTRY('C', 'L', 0)
#define CYW43_COUNTRY_CHINA             CYW43_COUNTRY('C', 'N', 0)
#define CYW43_COUNTRY_COLOMBIA          CYW43_COUNTRY('C', 'O', 0)
#define CYW43_COUNTRY_CZECH_REPUBLIC    CYW43_COUNTRY('C', 'Z', 0)
#define CYW43_COUNTRY_DENMARK           CYW43_COUNTRY('D', 'K', 0)
#define CYW43_COUNTRY_ESTONIA           CYW43_COUNTRY('E', 'E', 0)
#define CYW43_COUNTRY_FINLAND           CYW43_COUNTRY('F', 'I', 0)
#define CYW43_COUNTRY_FRANCE            CYW43_COUNTRY('F', 'R', 0)
#define CYW43_COUNTRY_GERMANY           CYW43_COUNTRY('D', 'E', 0)
#define CYW43_COUNTRY_GREECE            CYW43_COUNTRY('G', 'R', 0)
#define CYW43_COUNTRY_HONG_KONG         CYW43_COUNTRY('H', 'K', 0)
#define CYW43_COUNTRY_HUNGARY           CYW43_COUNTRY('H', 'U', 0)
#define CYW43_COUNTRY_ICELAND           CYW43_COUNTRY('I', 'S', 0)
#define CYW43_COUNTRY_INDIA             CYW43_COUNTRY('I', 'N', 0)
#define CYW43_COUNTRY_ISRAEL            CYW43_COUNTRY('I', 'L', 0)
#define CYW43_COUNTRY_ITALY             CYW43_COUNTRY('I', 'T', 0)
#define CYW43_COUNTRY_JAPAN             CYW43_COUNTRY('J', 'P', 0)
#define CYW43_COUNTRY_KENYA             CYW43_COUNTRY('K', 'E', 0)
#define CYW43_COUNTRY_LATVIA            CYW43_COUNTRY('L', 'V', 0)
#define CYW43_COUNTRY_LIECHTENSTEIN     CYW43_COUNTRY('L', 'I', 0)
#define CYW43_COUNTRY_LITHUANIA         CYW43_COUNTRY('L', 'T', 0)
#define CYW43_COUNTRY_LUXEMBOURG        CYW43_COUNTRY('L', 'U', 0)
#define CYW43_COUNTRY_MALAYSIA          CYW43_COUNTRY('M', 'Y', 0)
#define CYW43_COUNTRY_MALTA             CYW43_COUNTRY('M', 'T', 0)
#define CYW43_COUNTRY_MEXICO            CYW43_COUNTRY('M', 'X', 0)
#define CYW43_COUNTRY_NETHERLANDS       CYW43_COUNTRY('N', 'L', 0)
#define CYW43_COUNTRY_NEW_ZEALAND       CYW43_COUNTRY('N', 'Z', 0)
#define CYW43_COUNTRY_NIGERIA           CYW43_COUNTRY('N', 'G', 0)
#define CYW43_COUNTRY_NORWAY            CYW43_COUNTRY('N', 'O', 0)
#define CYW43_COUNTRY_PERU              CYW43_COUNTRY('P', 'E', 0)
#define CYW43_COUNTRY_PHILIPPINES       CYW43_COUNTRY('P', 'H', 0)
#define CYW43_COUNTRY_POLAND            CYW43_COUNTRY('P', 'L', 0)
#define CYW43_COUNTRY_PORTUGAL          CYW43_COUNTRY('P', 'T', 0)
#define CYW43_COUNTRY_SINGAPORE         CYW43_COUNTRY('S', 'G', 0)
#define CYW43_COUNTRY_SLOVAKIA          CYW43_COUNTRY('S', 'K', 0)
#define CYW43_COUNTRY_SLOVENIA          CYW43_COUNTRY('S', 'I', 0)
#define CYW43_COUNTRY_SOUTH_AFRICA      CYW43_COUNTRY('Z', 'A', 0)
```

```
#define CYW43_COUNTRY_SOUTH_KOREA       CYW43_COUNTRY('K', 'R', 0)
#define CYW43_COUNTRY_SPAIN             CYW43_COUNTRY('E', 'S', 0)
#define CYW43_COUNTRY_SWEDEN            CYW43_COUNTRY('S', 'E', 0)
#define CYW43_COUNTRY_SWITZERLAND       CYW43_COUNTRY('C', 'H', 0)
#define CYW43_COUNTRY_TAIWAN            CYW43_COUNTRY('T', 'W', 0)
#define CYW43_COUNTRY_THAILAND          CYW43_COUNTRY('T', 'H', 0)
#define CYW43_COUNTRY_TURKEY            CYW43_COUNTRY('T', 'R', 0)
#define CYW43_COUNTRY_UK                CYW43_COUNTRY('G', 'B', 0)
#define CYW43_COUNTRY_USA               CYW43_COUNTRY('U', 'S', 0)
```

Appendix B – DST country definitions

The available Daylight Saving Time (DST) country definitions are as follow:

DST setting	Country / World area	DST start time	DST end time
0	Any country – No DST support at all. It's like if Summer Time doesn't exist for the clock.	= = =	= = =
1	Australia	1st Sunday October, 2h00	1st Sunday April, 3h00
2	Australia Lord Howe Island (30 minutes shift)	1st Sunday October, 2h00	1st Sunday April, 2h00
3	Chile	1st Saturday September, 24h00	1st Saturday April, 24h00
4	Cuba	2nd Sunday March, 0h00	1st Sunday November, 1h00
5	European Union Akrotin and Dhekelia Albania Andorra Bosnia and Herzegovina Faroe Islands Gibraltar Greenland (except Danmarkshavn and Thule Air Base) Guernsey Isle of Man Jersey Kosovo	Last Sunday March, 1h00 *UTC Time*	Last Sunday October 1h00 *UTC Time*
6	Israel	Friday before last Sunday March, 2h00	Last Sunday October, 2h00
7	Lebanon	Last Sunday March, 0h00	Last Sunday October, 0h00
8	Moldova	Last Sunday March, 2h00	Last Sunday October, 3h00
9	New Zealand	Last Sunday September, 2h00	1st Sunday April, 2h00
10	Bahamas Bermuda Canada (except Yukon and Saskatchewan) Greeland (Thule Air Base) Haiti Nunavut Ontario Quebec U. S. A.	2nd Sunday March, 2h00	1st Sunday November, 2h00
11	Palestine	Saturday before last Sunday March, 2h00	Saturday before last Sunday October, 2h00
12	Paraguay	1st Sunday October, 0h00	4th Sunday March, 0h00