**644/1284 Wide & Power versions**

#### **User's manual**

[Presentation of the boards 2](#_Toc73174354)

[Board specifications 4](#_Toc73174355)

[Pinout 5](#_Toc73174356)

[Installation 7](#_Toc73174357)

[Arduino programming environment 7](#_Toc73174358)

[Automatic install under Windows 7](#_Toc73174359)

[CH340 driver 7](#_Toc73174360)

[Board definition / MightyCore 8](#_Toc73174361)

[Arduino IDE configuration when using the MightyCore 10](#_Toc73174362)

[Programming 11](#_Toc73174363)

[SPI programming 11](#_Toc73174364)

[USB programming 11](#_Toc73174365)

[Programming examples 12](#_Toc73174366)

[Blink 12](#_Toc73174367)

[2nd Serial port 12](#_Toc73174368)

[0.96” OLED display 12](#_Toc73174369)

[1024 samples FFT 13](#_Toc73174370)

[Schematics 14](#_Toc73174371)

[Troubleshooting 15](#_Toc73174372)

## Presentation of the boards

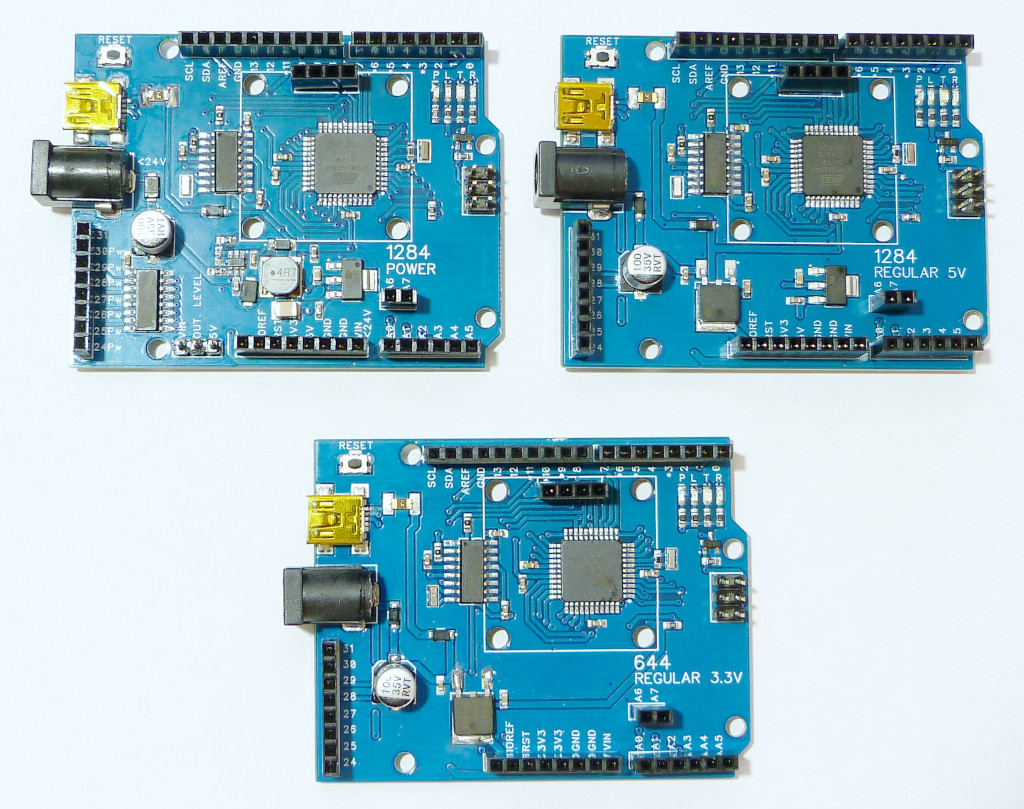
These boards have the same form factor as Arduino UNO/Leonardo boards but offer much more memory and a choice of powering configurations.

Complementary to the [644/1284 Narrow boards](https://github.com/mrguen/644-1284-Narrow/tree/master/doc), this is a large choice of boards based on Atmega644 and Atmega1284 microcontrollers with different powering circuits because:

* Many recent and performing components are running at 3.3v
* Mcu boards are also often used not only to manage signals but also to drive power components.

These various power requirements are not easy to address with only one standard board. So, why not have all the board variants needed? Here is the 644/1284 Wide & power offer:

* 644 Wide 3.3V
* 644 Wide 5V
* 644 Power 5V
* 1284 Wide 3.3V
* 1284 Wide 5V
* 1284 Power 5V

****

As compared to the Arduino Uno board they feature:

* 4KB/16KB instead of 2 KB of RAM
* 64KB/128KB instead of 32 KB of Flash memory
* 2KB/4KB instead of 1 KB of EEPROM
* One more USART port (Atmega1284)
* 12 more digital I/O

Even though the line of product of Atmega644/Atmega1284 is different, they remain part of the latest 8 bits MCU from Atmel with quite the same architecture as that of the Atmega328. So, if you are familiar with the Atmega328 based boards (Uno, Nano etc …) you will not have any problem using the Wide & Power boards. Programming in the Arduino environment is the same.

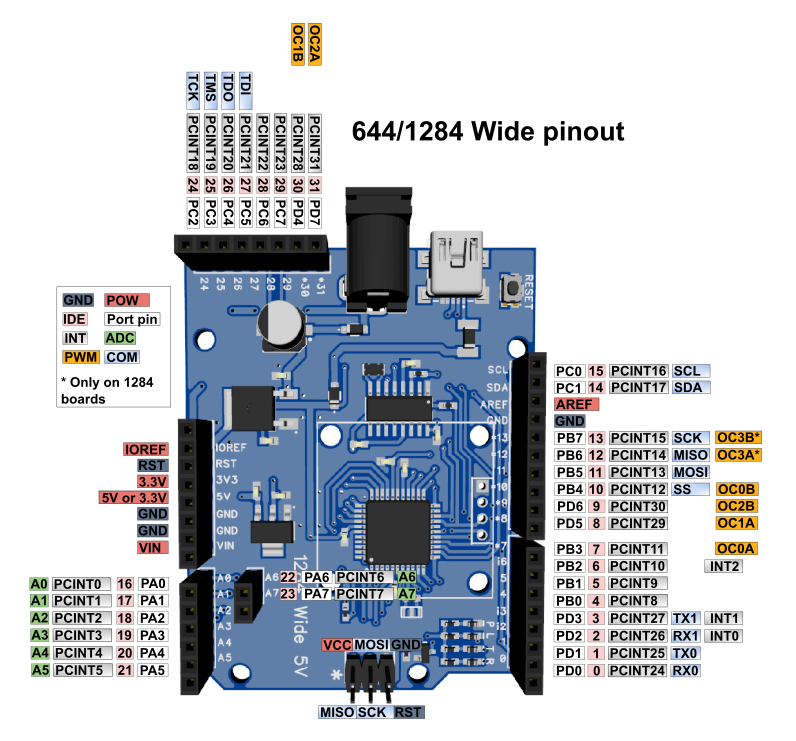
[You can download the Atmega644/1284 data sheet from this link](http://ww1.microchip.com/downloads/en/devicedoc/atmel-8272-8-bit-avr-microcontroller-atmega164a_pa-324a_pa-644a_pa-1284_p_datasheet.pdf)

### Board specifications

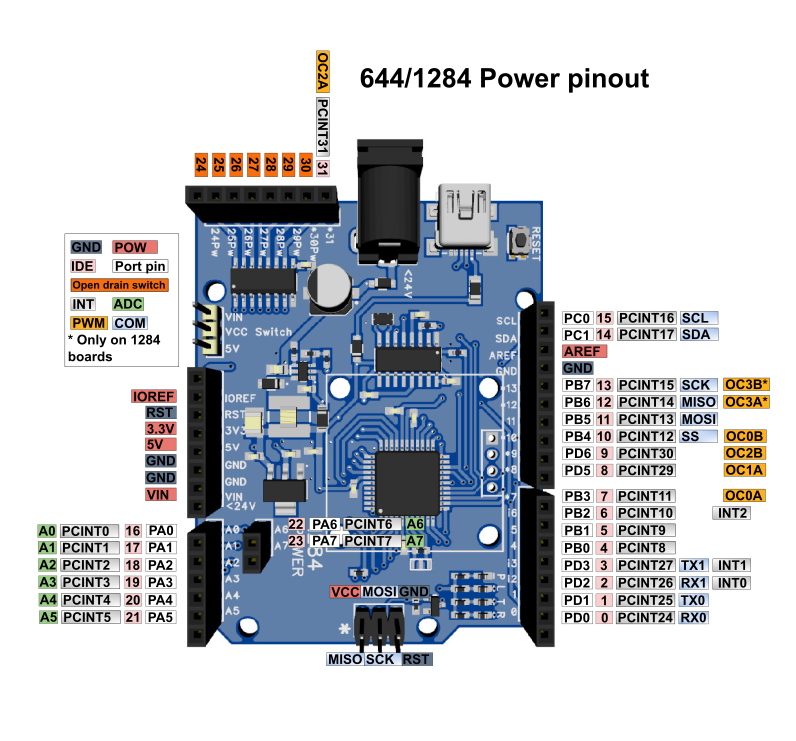
|  | **644 Wide / Power** | **1284 Wide / Power** |
| --- | --- | --- |
| **MCU** | ATMEGA644PA-AU | ATMEGA1284P-AU |
| **Operating voltage** | Vin: 6.2 to 15 volts or 6 to 24 volts (Power version)  USB: 4.75 to 5.25 volts  **5V version**: On board 5V (800 mA) and 3.3V (250 mA) regulators.  **3.3V version**: On board 3.3V (800 mA) regulator  **Power version**: DC/DC circuit up to 1200 mA | Vin: 6.2 to 15 volts or 6 to 24 volts (Power version)  USB: 4.75 to 5.25 volts  **5V version**: On board 5V (800 mA) and 3.3V (250 mA) regulators.  **3.3V version**: On board 3.3V (800 mA) regulator  **Power version**: DC/DC circuit up to 1200 mA |
| **Max current** | USB: 500 mA limited by a PTC.  Vin: 800 mA @ 6.2V, 25°c (1)  Or 1200 mA@any input voltage (Power version) | USB: 500 mA limited by a PTC.  Vin: 800 mA @ 6.2V, 25°c (1)  Or 1200 mA@any input voltage (Power version) |
| **Flash** | 64 Kb | 128 Kb |
| **RAM** | 4 Kb | 16 Kb |
| **EEPROM** | 2 Kb | 4 Kb |
| **Digital I/Os** | 24 / 17 (Power version) | 24 / 17 (Power version) |
| **Open drain switches** | 0 / 7 (Power version) | 0 / 7 (Power version) |
| **Analog I/Os** | 8 | 8 |
| **PWM** | 6 | 8 |
| **SPI** | 1 | 1 |
| **USART** | 1 | 2 |
| **I2C** | 1 | 1 |
| **USB** | 1 | 1 |
| **Size** | ~3600 mm² | ~3600 mm² |
| **Weight** | 1. g | 1. g |

1. **Note that the higher VIN you use the less current you can draw. Practically the effective max current when using the Wide version is around 200 mA@Vin=12V.**

### ****Pinout****



### 



## Installation

### Arduino programming environment

Of course, if you intend to develop your program in the Arduino Environment you must have the Arduino IDE installed. You can download it from <https://www.arduino.cc/en/main/software>

The following installation procedure was tested with the **Arduino IDE version 1.8.13**

### Automatic install under Windows

Windows users can simplify the installation process:

1. Install the MightyCore. Follow the installation process: <https://github.com/MCUdude/MightyCore#how-to-install>.
2. Download and execute

<https://github.com/mrguen/MightyCore/blob/master/MightyCore-Pandauino-WINDOWS-installer.exe>

The installer is not signed so it is normal if you get a warning.

It will:

* Install the boards definitions in your local sketch/hardware folder
* Install the CH340 driver (close the driver installer if it is already installed)

You can skip the next explanations on “CH340 driver” installation and “Board definition”, but if you need to troubleshoot the automatic installation, and have a look at the [Arduino IDE configuration](#_Arduino_IDE_configuration).

### CH340 driver

The USB/Serial adapter is the well know CH340G chip. It is a Chinese chip that is implemented on most Nano clones and works well.

You will need to install its driver from

<http://www.wch-ic.com/downloads/CH341SER_EXE.html>

You can also refer to <https://sparks.gogo.co.nz/ch340.html>

Once the driver is installed and you connect the board to your computer using a USB plug you should see it listed in the USB peripherals with a name like “CH340”

### Board definition / MightyCore

The Arduino IDE does not include the Atmega644 and Atmega1284 support by default. You can find a couple of boards definitions on GitHub see

<https://github.com/search?q=Atmega644>

<https://github.com/search?q=Atmega1284>

But you will probably like the MightyCore. There are two possibilities to install the MightyCore boards definition including Wide and Power boards:

* + Copy the files from the repository <https://github.com/mrguen/MightyCore> into your personal sketchbook /hardware folder. This is the preferred option to have a coherent configuration.
  + Or modify the automatic install of the MightyCore from the Arduino. It is a possibility if you already have the MightyCore installed but it might fail in future versions of the MightyCore.

#### OPTION A – INSTALL BOARDS DEFINITION IN YOUR SKETCHBOOK DIRECTORY

* Go to <https://github.com/mrguen/MightyCore>
* Click the green button Code – Download zip and save the archive to your computer
* Unzip its content in your sketchbook /hardware folder.

For example, on Windows 10 standard install it is located in

C:\Users\*USER\_NAME*\Documents\Arduino\hardware

Create the \hardware folder if necessary.

#### OPTION B – MODIFY THE MIGHTYCORE AUTOMATIC INSTALL

This procedure is tested for Arduino IDE 1.8.13 and MightyCore 2.1.1.

* Install the MightyCore. Follow the installation process: <https://github.com/MCUdude/MightyCore#how-to-install>.
* Download the variant file from <https://github.com/mrguen/MightyCore/tree/master/avr/variants/wide> and place it inside your MightyCore install directory. For example on Windows 10 standard install it is located in C:\Users\ USER\_NAME \AppData\Local\Arduino15\packages\MightyCore\ hardware\avr\*VERSION*\variants
* Modify the MightyCore install board.txt (in C:\Users\*USER\_NAME* \AppData\Local\Arduino15\packages\MightyCore\hardware\avr\*VERSION* ) by adding these lines where appropriate:

1284.menu.pinout.wide=Wide pinout

1284.menu.pinout.wide.build.variant=wide

1284.menu.pinout.wide.build.bootloader\_led=B7

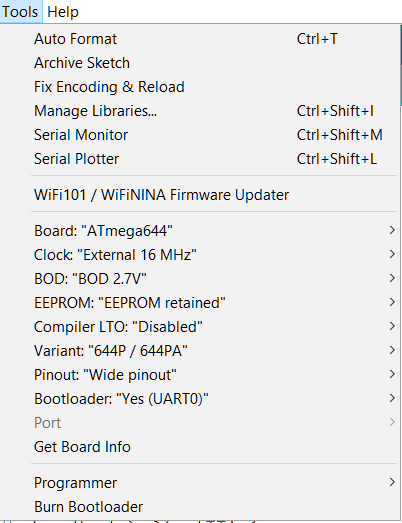
…

644.menu.pinout.wide=Wide pinout

644.menu.pinout.wide.build.variant=wide

644.menu.pinout.wide.build.bootloader\_led=B7

### Arduino IDE configuration when using the MightyCore



In the Arduino IDE **Tools** menu, you should select these options:

**Board:** MightyCore (in sketchbook)/ Atmega644 or Atmega1284

**Clock:** External 16 MHz

**BOD:** BOD 2.7V

**EERPOM**: “EEPROM retained”, or not depending on your application

**Compiler LTO:** disabled but you can also use LTO enabled

**Variant:** P or PA variant

**Pinout:** Wide pinout

**Bootloader:** Yes (UART0)

**Port:** the COM port given by your system (once the bootloader has been installed and the board connected through the USB port)

## Programming

You can program the board directly through the USB connector, since the bootloader is already loaded. But you can also program it through the ISP socket. In this case it will write over the bootloader.

### SPI programming

You can program the board using a programmer like the [USBAsp](https://www.fischl.de/usbasp/) or any of the many other supported programmers.

1. Install your programmer on your system
2. Connect the programmer to the ISP port (2\*3 header) on the device and to the computer. It should be properly listed as a device by your system. **Make sure you connect in the right direction with the pin number 1 identified by \***
3. Select the proper programmer in **Tools/Programmer**
4. Select the menu **Sketch/Upload Using Programmer**.

### USB programming

When the bootloader is installed, you can also program the device through a USB port.

The bootloader has been uploaded on the device already but it might have been written over if you programmed the board using the ISP socket.

In this case, burn the bootloader by uploading it to the board with a programmer (see steps 1-3 of SPI programming): use the menu **Tools/Burn Bootloader.**

To upload your code you can simply **press the arrow button** in the main window of the Arduino IDE or the menu **Sketch/Upload**

## Programming examples

### Blink

Usually you might want to check that everything is working properly by programming the sketch Blink.ino that you will find in the Arduino IDE menu **Files/Examples/Basic/Blink**

The led labelled “L” on the board will be blinking at low speed.

If id does not, check that you have defined the board as explained in [Board definition / MigthyCore](#_Board_definition_/)

### 2nd Serial port

There are two serial ports.

1) You can test that they are working using the sketch that you will find in the Arduino IDE menu **File/Examples/Communication/MultiSerial**

This sketch needs to have a serial device attached to the Serial 1 port (pin 2 RX1, pin 3 TX1).

2) If you don’t have such a device you can simply try this sketch

<https://github.com/mrguen/644-1284-Wide-Power/blob/main/examples/Endless_Serial_0_softwareSerial/Endless_Serial_0_softwareSerial.ino>

Connect pins 1 (TXO) and 2 (RX1) together.

Program the sketch through the USB port. In the Serial Monitor, if you type a letter to send to the board through the Serial port, it will display endlessly on the screen because it is sent back by serial “0” Tx pin to the serial monitor and also to serial “1” Rx, that writes again to serial “0” Tx.

### 0.96” OLED display

There is a group of four plated through-holes on the board to solder a 0.96” OLED 128x64 pixels display. Not all I2C OLED modules connections are arranged in the same order so check it before soldering in your display. These holes are connected to

GND – 5V – SCL - SDA

You can get an OLED kit from Pandauino, including all the spacers, screws and headers needed to plug the OLED in.

1. **Using Greiman’s text libray, modified by mrguen:**

This library <https://github.com/mrguen/SSD1306Ascii> offers a large choice of nice fonts and is lighter than the Adafruit Bundle, so it might be very convenient if you only need to display text

1. **Using Adafruit libraries**

Adafruit provides a set of libraries for display rendering, including text and graphics. You need to install these:

[Adafruit\_SSD1306](https://github.com/adafruit/Adafruit_SSD1306)

[Adafruit-GFX-Library](https://github.com/adafruit/Adafruit-GFX-Library)

### 1024 samples FFT

You can use the 16 KB of RAM of the 1284 Wide and Power boards to compute a 1024 samples FFT.

A tutorial is available at <https://www.instructables.com/1024-Samples-FFT-Spectrum-Using-an-Atmega1284/> Thus it was written for the 1284 Narrow board, it is essentially the same. Only the pin numbering is different.

## Schematics

TODO

## Troubleshooting

|  |  |
| --- | --- |
| **Problem** | **Possible solution** |
| The power led (blue) does not light up. | Check the voltage at the 5V pin. It should be between 4.5V and 5.25V  If not, maybe the polyswitch "PS” or the diode D101 or the regulator U2 might be damaged. But it is unlikely since the polyswitch should protect the circuit.  Most common cause is a bad connection to the voltage source (Vin or USB). |
| After you burn the bootloader the orange led does not blink twice | This led is supposed to constantly blink twice when the bootloader is present and no program was downloaded. If not, the led pin definition might be wrong. Check the proper configuration of the environment. [Board definition / MigthyCore](#_Board_definition_/) |
| When trying to download from the Arduino IDE you get a series of errors:  avrdude: stk500\_recv(): programmer is not responding avrdude: stk500\_getsync() attempt 1 of 10: not in sync: resp=0x1e  … | Check that you installed the [CH340 driver](#_CH340_driver)  Download the bootloader. See [Programming](#_Programming) / [USB programming](#_USB_programming)  If everything is properly configured when you connect the board using an USB cable it should be enumerated as a COM port "USB- SERIAL CH340".  If not, check the USB cable and connections. |