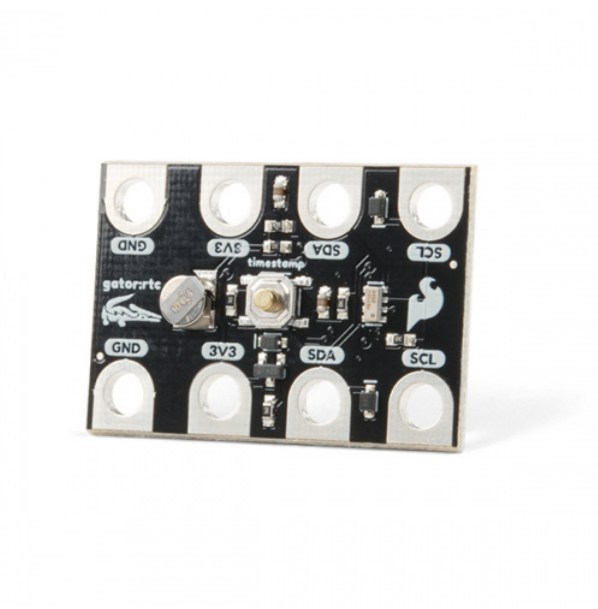
SparkFun gator:RTC Hookup Guide

Introduction

Most of you probably have done a science experiment that required data of some sort of to be recorded. In which case, the data often had a temporal component that also needed to be recorded...



SparkFun gator:RTC - micro:bit Accessory Board
COM-15486

The gator:RTC is a real-time clock. This means that the gator:RTC can be used as a supplemental data logging tool, in conjunction with the gator:log to add precise timing to your data collection. Therefore, student(s) can put down that stopwatch and start focusing more attention on what is happening in their experiments. Other applications for the gator:RTC include time lapse photography, timers and alarms, clocks, and etc.

This tutorial will show you how to get started using this gator:RTC with the gator:bit (v2) in the micro:bit development environment.

Product Showcase: SparkFun gator:log, gator:RTC, & gator:UV

Required Materials

To get started, you'll need a micro:bit to control everything. Each of the products below includes a micro:bit, but the kit and bundle also include some additional accessories that you may want as well.

SparkFun Inventor's Kit for micro:bit
KIT-15228

micro:bit Go Bundle
DEV-14336

micro:bit Board
DEV-14208

To easily use the gator board ecosystem, a gator:bit (v2) will help breakout the necessary pins and you will also need alligator and/or banana cables to connect the gator:bit to the gator:RTC.

SparkFun gator:bit v2.0 - micro:bit Carrier Board
DEV-15162

Alligator Test Leads - Multicolored (10 Pack)
PRT-12978

Banana to Banana Cable - Right Angle
CAB-15368

(**These banana cables have a special diameter on the attachment points designed specifically for use with the micro:bit ecosystem. They may or may not be compatible with the banana cables used on your test equipment.*)

You may already have some of these materials, so feel free to modify your cart as necessary.

Suggested Reading

The gator:RTC sensor is pretty straight forward to use in application. However, you may find the following concepts useful along the way.

Logic Levels
Learn the difference between 3.3V and 5V devices and logic levels.

I2C
An introduction to I2C, one of the main embedded communications protocols in use today.

New!

Qwiic Real Time Clock Module (RV-1805) Hookup Guide

Find out what time it is, even after the power's been out on your project for a while with the Qwiic Real Time Clock (RTC) module.

SparkFun gator:log Hookup Guide

The gator:log is a serial communication based data logger. This tutorial will get you started using the gator:log with the micro:bit platform.

If you aren't familiar with the micro:bit, we recommend reading here for an overview.

micro:bit Ecosystem

We would also recommend taking a look at the following tutorials if you aren't familiar with them.

Getting Started with the micro:bit

The BBC micro:bit is a compact, powerful programming tool that requires no software installation. Read on to learn how to use it YOUR way!

Getting Started with MicroPython and the SparkFun Inventor's Kit for micro:bit

Learn MicroPython with the micro:bit.

How to Load MicroPython on a Microcontroller Board

This tutorial will show you how to load the MicroPython interpreter onto a variety of development boards.

SparkFun gator:bit v2 Hookup Guide

The gator:bit v2 is a breakout board for the BBC micro:bit. The gator:bit exposes almost every pin on the micro:bit to clippable pad with circuit protection. It also has as built-in addressable LEDs and a built-in buzzer.

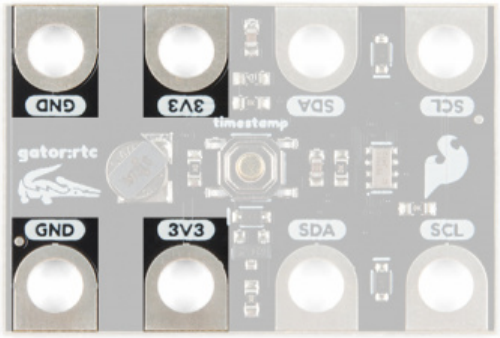
Hardware Overview

The gator:RTC consists of 4 pads for power and data.

Contacts	Direction	Description
GND	N/A	Ground: Reference voltage and ground loop.
3.3V	In	Power: Provides **3.3V** to board.
SDA	Bi-directional	Data: Data and commands are transmitted between the RV-3028 and microcontroller.
SCL	In	Clock: The microcontroller provides the timing needed to communicate on the data line.

Power

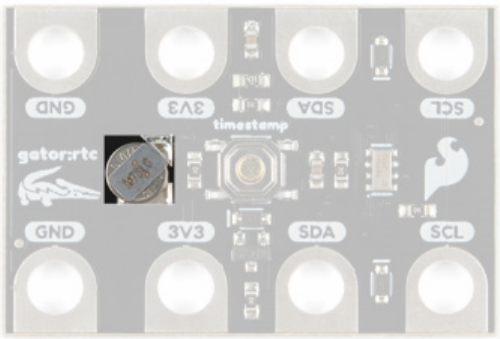
The specified operating voltage for the RV-3028 is between **1.2 - 5V**. For use with the gator:bit (v2) and micro:bit, you should provide **3.3V** through the **3V3** and **GND** pads to keep the logic levels consistent.



Power connection pads.

Backup Battery

The board also includes a **1mAh** lithium-metal rechargeable, coin type backup battery. This allows the RTC module to keep track of time even when power to the board is cut (for low power applications). Although, current consumption varies by how the RTC is operating and the temperature, users should expect a fully charged backup battery to last for several months with the RTC operating conservatively.



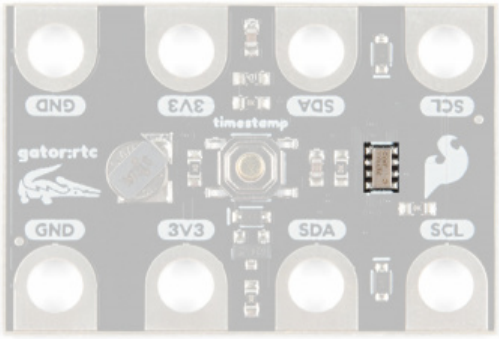
Backup battery.

Here are some of the characteristics of the battery from the datasheet:

Characteristic	**Range**
Nominal Voltage	**3.0V**
Nominal Capacity	1.0mAh (3.1V - 2.0V)
Discharge Current	5µA

RV-3028

The RV-3028 is an extremely low power real-time clock (RTC), consuming only **40 - 60nA** on average at **3V**, which interfaces over an I^2C bus. A real-time clock is used to keep track of the current time, which is a useful, but often forgotten tool. For the most part, users will only need to know I2C addresses of this board to prevent address conflicts with other devices or sensors.



RV-3028: Real-Time Clock.

Here are some of the characteristics of the RV-3028 RTC, for details, check out the datasheet and application manual:

Characteristic	**Range**
Operating Voltage	**1.2V to 5V**
Supply Current (@3V)	**40nA to 60nA** (avg.) (~330nA peak) 5µA to 40µA (active I^2C-bus) 95nA to 150nA (backup battery operation)
Operating Temperature	-40°C to 85°C
Xtal (Internal Oscillator) Frequency	32.768 kHz (± 5ppm @ 25°)
Timing Accuracy	Factory calibrated: **± 1ppm** @ 25°C (no temp. compensation)
Calender Range	Automatic leap year correction: 2000 to 2099
Counter	Seconds, minutes, hours, date, month, year and weekday
I^2C Address	7-bit Slave Address: **X1010010b**, where **X** is the R/W bit **0x52** - Write Address **0xD2** - Read Address

Note: The I^2C address indicated on the datasheet is incorrect. The R/W bit is the MSB and not the LSB, as shown in the table above.

What is Time?

Time is the continuous progression of events which succeed one another from the past and into the future. By most human standards, time is measured in a base unit of time, called a second. Most recently, the second (SI unit of time) was re-defined to a more specific standard:

> It is defined by taking the fixed numerical value of the cesium frequency Δv_{Cs}, the unperturbed ground-state hyperfine transition frequency of the cesium-133 atom, to be 9,192,631,770 when expressed in the unit Hz, which is equal to s^{-1}.

Below, are additional resources regarding the topic of time:

- History of the Definition of a Second
- History of Timekeeping Devices
- How does one arrive at the exact number of cycles of radiation a cesium-133 atom makes in order to define one second?
- These Physicists Watched a Clock Tick for 14 Years Straight: It was to test Einstein's theory of general relativity.

What is an RTC?

RTC stands for real-time clock, which is used to keep track of time. With the convenience of modern electronics such as computers, smartphones, GPS navigation, and IoT/smart devices, the importance of an RTC can be easily overlooked. An RTC is different from an oscillator or hardware clock that only generates a clock signal; the primary difference being that an RTC can keep track of the time. In the case of the gator:log, the RV-3028 is is able to keep track of time by the **month**, **day** (*including **weekday***), **year** (*leap-year accurate from 2000 to 2099*), **hours** (*12-hour format*), **minutes**, and **seconds**. This is useful for a variety of applications including, but not limited to data logging, timers, and other long duration projects like time-lapse photography.

The key benefits of using an RTC in your project is that they are:

- Low Power: The RV-3028 only consumes about 40 - 60nA when only tracking time.
- No Power: With the backup battery, the RV-3028 is able to keep track of time. The clock stored on a micro:bit would resume from the last point of reference or reset depending on the methods used.
- Accuracy: The RV-3028 is factory calibrated to ± 1ppm. This equates to a deviation of less than 32 seconds over the course of the year. (**Your average wrist watch is accurate to about ± 50ppm, which is a variance of 27 minutes per year.*)

The drawbacks of an RTC that should be considered before using it in your project are:

- Resolution: The resolution of the RV-3028 is limited to 1 second intervals.
- Latency: Although the RTC operates on a "*higher*" speed I^2C bus, cycles are still spent accessing timing data from the module. In addition to the limited resolution, this limits the functionality of an RTC for high speed applications, like a data acquisition systems.

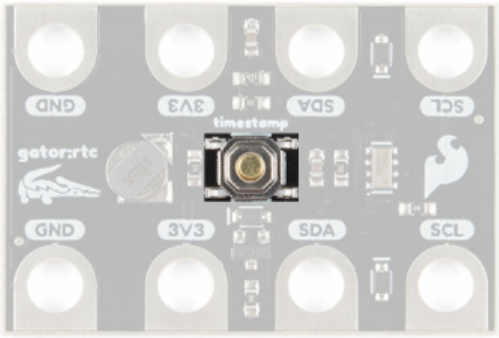
The RTC is most useful for long duration projects. Here is a list of some examples:

- Recording climate data to track changes by season or year. Linked below, are some additional resources on the topic of climate change tracking:
  - Global Climate Change: Vital Signs of the Planet
  - Climate Monitoring
  - How NOAA Keeps Track of Carbon Dioxide and Other Greenhouse Gases
- Time-laspe photography may not need an RTC, but for those interested in capturing exact moments of the day, and RTC would be a great tool to use. Especially, as a redundancy if your timer uses a web-based clock. This is a great article, where the photographer actually uses an Arduino to trigger his photos.
- In a more extreme case, for preppers, who are expecting to live in an underground shelter for extended periods of time and may not have access to GPS signals for accurate time keeping. After 10 years, your

wristwatch may be off by up to 5 hours. That may not be much, but it would be the difference between popping your head out at dawn (e.g. 6AM) or in the middle of the night/day (e.g. 1AM/11AM).

Time Stamp Button

In addition to tracking time, the gator:RTC also includes a time stamp button to mark events. When pressed, the RV-3028 stores a time stamp of the button press. Therefore, it queried consistently, this button can be used to mark significant events in a data log. For example, marking the end point of a pH titration in a data set.

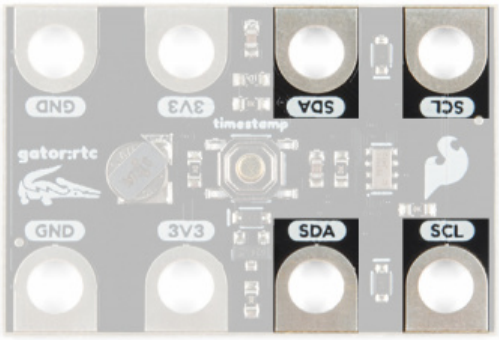


Time stamp button.

Note: Only a single time stamp is stored by the RV-3028. Therefore, it is advisable to query the RTC often enough so that previous time stamps aren't overwritten before they can be collected.

I^{2}C Connection

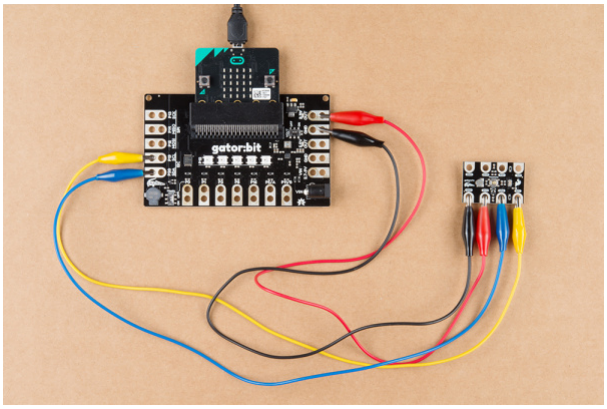
I^{2}C is a communication protocol used to transfer data between *master* and *slave* devices. It is a 2-wire connection consisting of SDA (data) and SCL (clock). The protocol is pretty well defined so it can be shared with multiple devices. This is beneficial for daisy chaining multiple devices together on a single bus. The primary thing users will need to watch out for is address conflicts (you can't have devices with the same address).



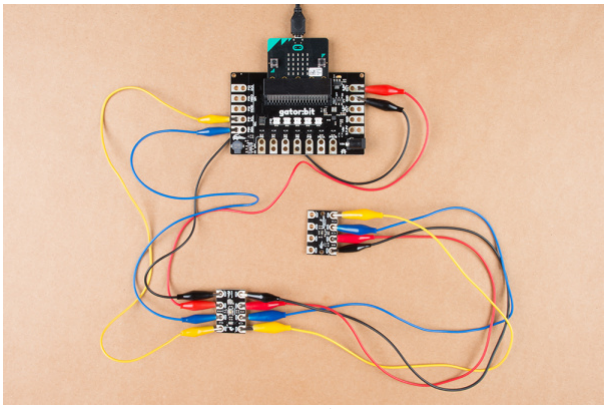
I^{2}C connection pads.

Hardware Assembly

Connecting your gator:RTC to the gator:bit (v2) is simple. The board can also be daisy-chained with other I^{2}C boards. This can easily be done with alligator cables or these *special* banana cables.



Hardware assembly of connections used in example. Click to enlarge.



Example of daisy chaining multiple I²C gator boards. Click to enlarge.

Gator:RTC	GND	3V3	SDA	SCL
Gator:Bit	GND	3.3V OUT	P20 (SDA)	P19 (SCL)

Adding the MakeCode Extension

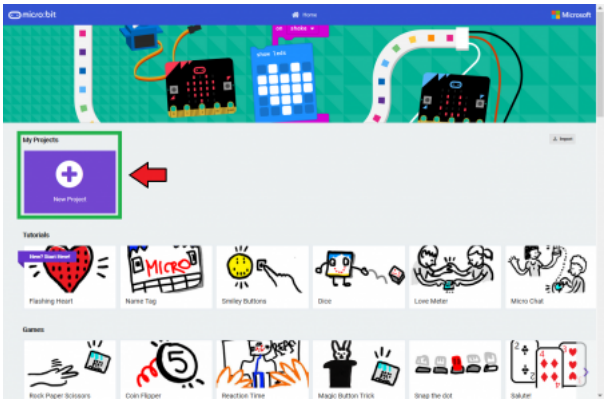
Note: This tutorial assumes you are familiar with MakeCode, the gato:bit (v2), and the micro:bit board.

- If you have not previously used **MakeCode**, please check out the Quick Start Guide fro micro:bit.
- If this is your first time using a **micro:bit**, please review our Getting Started with the micro:bit guide.
- If this is your first time using the **gato:bit (v2)**, please review our SparkFun gator:bit v2 Hookup Guide.

The easiest way to get started using the gator:RTC is to use Microsoft MakeCode, a web-based block editor. This tutorial assumes you are familiar with the with MakeCode, the gato:bit (v2), and the micro:bit development board. If this is your first time check out this guides linked in the suggested reading section (above).

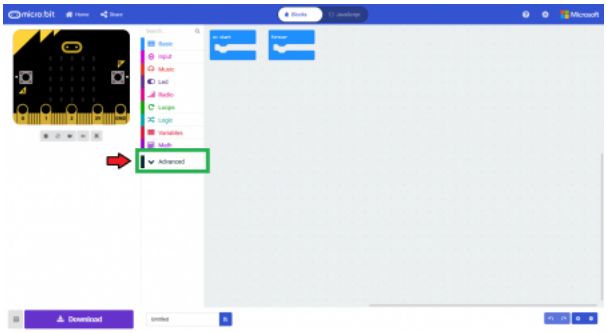
Installing Extensions

To get started with using MakeCode with the miccro:bit, head over to the MakeCode for micro:bit website by Microsoft. Then, click the **New Project** button to start a new project and open the *Editor*. (**Yes, all you only need to get start coding is a computer with internet access, an up-to-date web browser, and an available USB port!*)



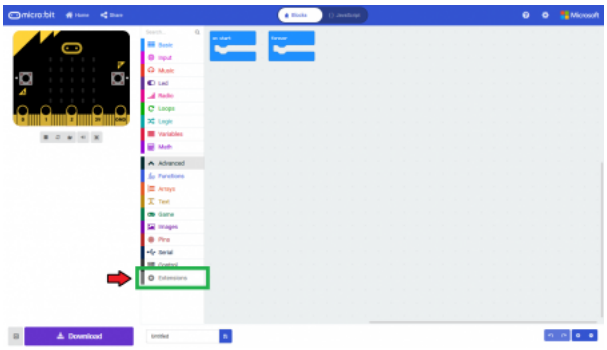
*Click the **New Project** button to start a new project and open the Editor. Click on image to enlarge.*

Once you have the editor up, click on the the `Advanced` block in the block library to reveal the drop down menu.



*Click on the the **Advanced** block in the block library to reveal the drop down menu. Click on image to enlarge.*

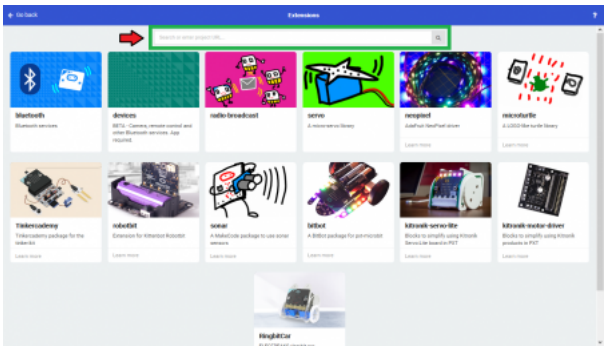
Finally, click on the `Extensions` block. This should bring up the extensions page. (**Alternatively, you could also click on the extensions link from the ⚙ settings menu.*)



*Click on the the **Extensions** block to open the extensions page. Click on image to enlarge.*

There are two methods for finding the gator:RTC extension:

- Search for the **name** used on the extension[3].
- Use the link to the GitHub repository for the **pxt-package** as the search term.



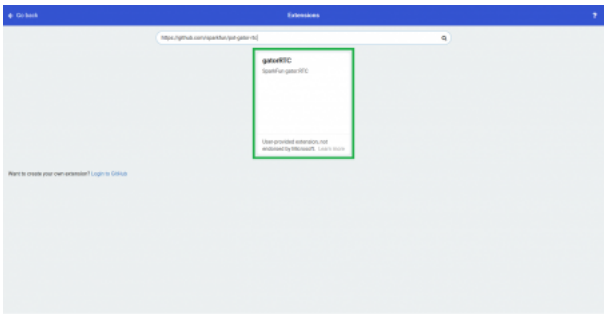
Use the search bar to find the extension you want to install. Click on image to enlarge.

Note: Unfortunately, it does take time to get an extension approved by the micro:bit Educational Foundation before it can be searchable by name. Part of the requirements for the approval process involves a live product page. Therefore, at the time of the launch of this product, the extension has not been approved yet and the only method of adding the extension is to use the link to the GitHub repository of the pxt-package. We will update this tutorial as soon as the extension has been approved.↩

Search for the PXT-Package

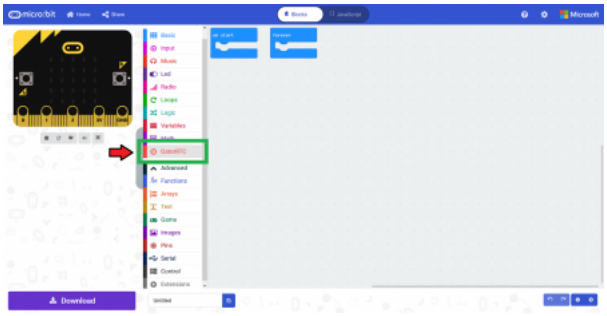
Search for the gator:RTC extension using the GitHub repository link to the pxt-package:

```
https://github.com/sparkfun/pxt-gator-RTC
```



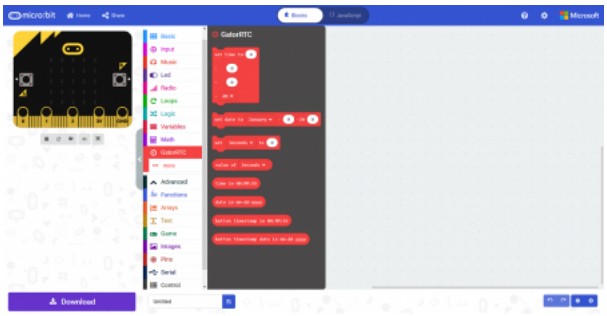
Search for Extension. Then, click on the box to add it to the block library. Click on image to enlarge.

Then, click on the box for the extension to add it to the block library. The gator:RTC extension should now appear in the block library.



Extension in the block library; click on the block to open menu options. Click on image to enlarge.

To verify that you have added the extension, click on the `gator:RTC` block to make sure that your drop down menu options match the image below.

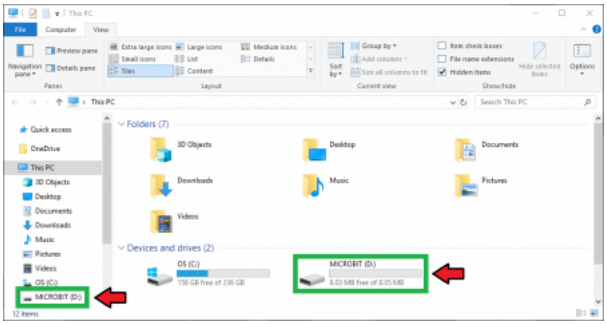


Available blocks for the gator:RTC extension. Click on image to enlarge.

MakeCode Examples

Now that you have added the gator:particle extension to the *Editor*, lets start with some example code. Plug the micro:bit into your computer using an USB micro-B cable after you have assembled your hardware with the instructions from the previous section. The micro:bit should appear on your computer as a removable storage

device.



The micro:bit showing up as a USB drive on a Windows computer. Click to enlarge.

To upload new code, this is where you will be copying the downloaded `.hex` file to later.

Block Functions

set time to ___
: ___
: ___
- _____

This block should allows users to set the time stored in the RTC. The time should take the following format: `HH:MM:SS` in a 12-hour format. This block doesn't prohibit invalid entries (e.g 13:01:53 AM). Double check the time that is being set is valid; otherwise, the RTC may respond improperly when called upon.

- `HH` - Integer value for the our in 12-hour format (i.e. The 3rd hour after 12 should be entered as `3`). There is also a slider that can be easily used.
  - `0` to `12`
- `MM` - Minutes in integer format (i.e. The 20th minute past the hour should be entered as `20`). There is also a slider that can be easily used.
  - `0` to `59`
- `SS` - Seconds in integer format (i.e. The 30th second past the minute should be entered as `20`). There is also a slider that can be easily used.
  - `0` to `59`
- `12-Hour Period` - Drop down menu to select the 12-hour period.
  - `AM` - Ante meridiem (*i.e. before noon*).
  - `PM` - Post meridiem (*i.e. after noon*).

set date to _____
, _____
- ___
- 20___

This block should allows users to set the date stored in the RTC. The date should take the following format: `Weekday, Month - DD - 20YY`. This block doesn't prohibit invalid entries (e.g Feb 29th for leap years, Sept. 31st, or `YY` = 100). Double check the date that is being set is valid; otherwise, the RTC may respond improperly when called upon.

- `Weekday` - Drop down menu to select the weekday.
  - `Monday`
  - `Tuesday`
  - `Wednesday`
  - `Thursday`
  - `Friday`
  - `Saturday`

    - `Sunday`
- `Month` - Drop down menu to select the month.
    - `January`
    - `February`
    - `March`
    - `April`
    - `May`
    - `June`
    - `July`
    - `August`
    - `September`
    - `October`
    - `November`
    - `December`
- `DD` - Day in integer format (i.e. The 3rd should be entered as `3`). There is also a slider that can be easily used.
    - `0` to `31`
- `YY` - Year in integer format (i.e. The 2019 should be entered as `19`). (*The calender is only leap-year accurate for the years **2000 to 2099***.)
    - `0` to `99`

set _____ to ___

This block should allows users to change a specific component of time. This block doesn't prohibit invalid entries. Double check the that the modification is valid; otherwise, the RTC may respond improperly when called upon.

- `Unit of Time` - Drop down menu to select the unit of time you want to change.
    - `Seconds`
    - `Minutes`
    - `Hours`
    - `Date`
    - `Month`
    - `Year`
    - `Weekday`
- `Value` - Value in integer format.
    - `Seconds` - Input an integer:
        - `0` to `59`
    - `Minutes` - Input an integer:
        - `0` to `59`
    - `Hours` - Input an integer:
        - `0` to `12` (*if in 12-hour format*)
        - `0` to `24` (*if in 24-hour format*)
    - `Date` - Input an integer:
        - `0` to `31`
    - `Month` - Input an integer:
        - `0` to `12`
    - `Year` - Input an integer:
        - `0 to 99`
    - `Weekday` - Input an integer:
        - `0 - Monday`
        - `1 - Tuesday`

        - `2 - Wednesday`
        - `3 - Thursday`
        - `4 - Friday`
        - `5 - Saturday`
        - `6 - Sunday`

value of _____

This is a value block that returns the value for the stored component of time from the RTC.

- `Unit of Time` - Drop down menu to select the unit of time you want to retrieve.
    - `Seconds` - Returns an integer:
        - `0` to `59`
    - `Minutes` - Returns an integer:
        - `0` to `59`
    - `Hours` - Returns an integer:
        - `0` to `12` (*if in 12-hour format*)
        - `0` to `24` (*if in 24-hour format*)
    - `Date` - Returns an integer:
        - `0` to `31`
    - `Month` - Returns an integer:
        - `0` to `12`
    - `Year` - Returns an integer:
        - `0 to 99`
    - `Weekday` - Returns an integer:
        - `0 - Monday`
        - `1 - Tuesday`
        - `2 - Wednesday`
        - `3 - Thursday`
        - `4 - Friday`
        - `5 - Saturday`
        - `6 - Sunday`

text of weekday

This is a value block that returns a `string` for the weekday.

time in HH:MM:SS

This is a value block that returns a `string` for the time stored by the RTC in a `HH:MM:SS` format.

date in mm-dd-yyyy

This is a value block that returns a `string` for the date stored by the RTC in a `mm-dd-yyyy` format.

button timestamp in HH:MM:SS

This is a value block that returns a `string` for the previous time, marked by the RTC timestamp button in a `HH:MM:SS` format.

button timestamp in mm-dd-yyyy

This is a value block that returns a `string` for the previous date, marked by the RTC timestamp button in a `mm-dd-yyyy` format.

set time to ___:___:___ in 24 hour mode

This block allows users to set the time stored in the RTC. The time should take the following format: `HH:MM:SS` in a 24-hour format. This block doesn't prohibit invalid entries (e.g 25:01:53). Double check the time that is being set is

valid; otherwise, the RTC may respond improperly when called upon.

- `HH` - Integer value for the our in 24-hour format (i.e. The 3rd hour after midnight should be entered as `3`). There is also a slider that can be easily used.
  - `0` to `24`
- `MM` - Minutes in integer format (i.e. The 20th minute past the hour should be entered as `20`). There is also a slider that can be easily used.
  - `0` to `59`
- `SS` - Seconds in integer format (i.e. The 30th second past the minute should be entered as `20`). There is also a slider that can be easily used.
  - `0` to `59`

time in yyyy-mm-ddThh:mm:ss

This is a value block returns a `string` for the time stored by the RTC in ISO8601 format.

date in dd-mm-yyyy

This is a value block that returns a `string` for the date stored in the RTC in `dd-mm-yyyy` format.

button timestamp in yyyy-mm-ddThh:mm:ss

This is a value block returns a `string` for the previous time, marked by the RTC timestamp button in ISO8601 format.

button timestamp in dd-mm-yyyy

This is a value block returns a `string` for the previous time, marked by the RTC timestamp button in `dd-mm-yyyy` format.

value of button timestamp in _____

This is a value block that returns a specific component of timestamp stored by the RTC.

- `Unit of Time` - Drop down menu to select the unit of time you want to retrieve from timestamp.
  - `Seconds` - Returns an integer:
    - `0` to `59`
  - `Minutes` - Returns an integer:
    - `0` to `59`
  - `Hours` - Returns an integer:
    - `0` to `24` (*in 24-hour format*)
  - `Date` - Returns an integer:
    - `0` to `31`
  - `Month` - Returns an integer:
    - `0` to `12`
  - `Year` - Returns an integer:
    - `0 to 99`

set to _____ time

This block allows users to change the time format between a 12-hour and 24-hour.

- `Standard` - Changes the time to a 12-hour format.
- `Military` - Changes the time to a 24-hour format.

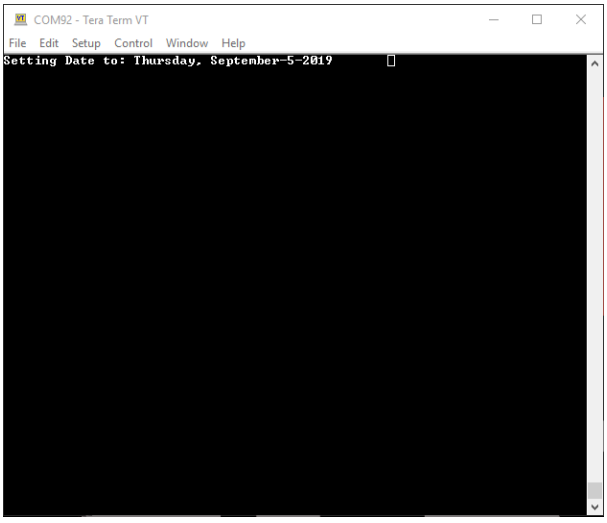
Basic Setup and Time Retrieval

Below, is a simple example of how to take simple readings from the RTC. To use this example, there are multiple options for accessing the `.hex` file:

- Replicate the block functions from the display below to copy the example project. Then download the `.hex` file.
- Expand the display widow to pull up the example project. Then download the `.hex` file.
- Use this link to pull up the example project. Then download the `.hex` file.
- Download the `.hex` file from the button below or the link on the bottom of the display.

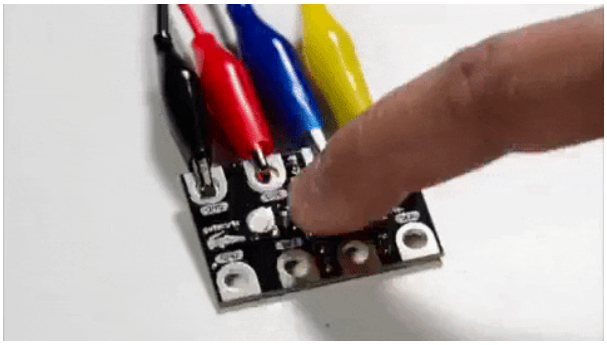
DOWNLOAD .HEX FILE FOR EXAMPLE

The output is redirected over the serial port to avoid conflicts on pins `P0` and `P1`, which are also used for serial communication. To read the sensor values, pull up your favorite serial terminal emulator. Then, connect to the serial port that the micro:bit is on; the default baud rate is 115200 bps. Below, is an example of an output of the time stored by the gator:RTC using the example code.



Example of readings from the RTC.

In the read out from the RTC, you can see with the 1.5 and 0.8 second delays that the resolution of the RTC is limited to one second intervals. (*Between the change from a 1.5 to a 0.8 delay, the seconds value was reset back to 0 seconds.*) Additionally, you should note the various subtleties in timing formats for the functions used.



Pressing timestamp button to change RTC output.

If you are confused to how the timestamp button works, the read out in the `forever` loop prints out both the current time and the timestamp of the last button press. In the example readings, you can see the timestamp change from a few button presses.

Troubleshooting

Power and Connections

Check that your power and data connections are correct and that the power `VOUT` switch is in the `ON` position on the gator:bit (v2).

Resolution

The RTC only reports the time down to 1 second intervals and cannot return fractions of a second. It will report the lower value between seconds until the next second is hit (*e.g. at 15.6 seconds past the minute is will report 15 seconds until the the 16th second is reached*).

If you still have questions or issues with this product, please create a post on our forum under the Gator Products topic.

Additionally, if you are an **educator** and you have class or application based questions, please create a post on our forum under the Educational Products topic.

Resources and Going Further

For more product information, check out the resources below:

- Schematic (PDF)
- Eagle Files (ZIP)
- RV-3028 Datasheet (PDF)
- RV-3028 Application Note (PDF)
- gator:RTC PXT Package
- GitHub Product Repo
- Dimensions (PDF)
- micro:bit Landing Page
- SFE Product Showcase

Interested in the micro:bit? Check out some of these other micro:bit products:

SparkFun micro:climate kit for micro:bit
KIT-15301

SparkFun gator:bit v2.0 - micro:bit Carrier Board
DEV-15162

SparkFun Inventor's Kit for micro:bit Lab Pack
LAB-15229

SparkFun Inventor's Kit for micro:bit
KIT-15228

Need some inspiration for your next project? Check out some of these other micro:bit product tutorials:

micro:bit Educator Lab Pack Experiment Guide
A quickstart guide for the micro:bit educator lab pack.

micro:arcade Kit Experiment Guide
We love games! We love writing games, building games and yes, even building game consoles. So we want to introduce to you the micro:arcade kit for the micro:bit!

Gator:control ProtoSnap Hookup Guide

Buttons and switches and inputs oh my! Start adding more control to your gator:bit with the SparkFun gator:control.

SparkFun gator:bit v2 Hookup Guide

The gator:bit v2 is a breakout board for the BBC micro:bit. The gator:bit exposes almost every pin on the micro:bit to clippable pad with circuit protection. It also has as built-in addressable LEDs and a built-in buzzer.