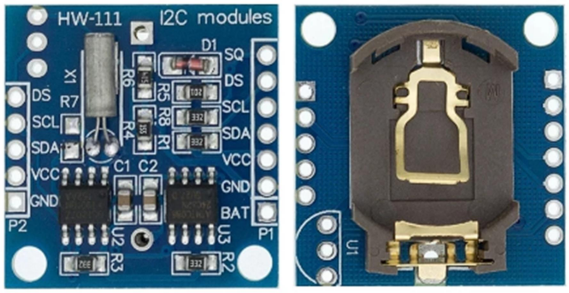
INSTALLING AN RTC MODULE TO DSO2X1X

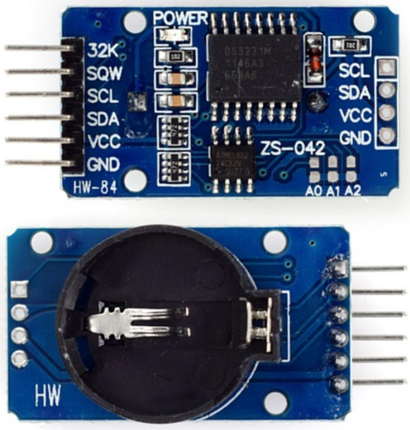
MODULE OPTIONS

There are multiple pre-made modules available al Aliexpress and other online stores. Those based on the DS1307 and DS3231 seem to be very popular. For example:

This one is based on the DS1307 is one of the cheapest. Besides the RTC IC itself, it also includes an AT24C32 I2C serial EEPROM mapped at the same address than the scope's EEPROM, so the module has to be modified (see next section). On its favour, is one of the cheapest, as can be found for about $1,30 at the time of writing.

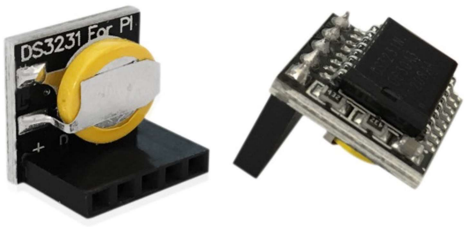


This second one is based on DS3231 and also has an AT24C32 EEPROM, but in this case pins 1-3 (A0-A2) are pulled up, so its default address is 0x57, which does not collide with the DSO's EEPROM. Also provides 3 solder jumpers (labelled A0, A1 and A2) in case you want to change the address by shorting them.



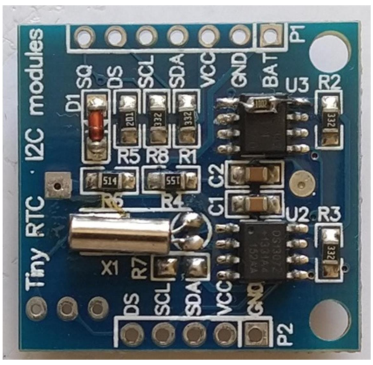
The DS3231 is more precise than the DS1307, as its internal oscillator is temperature compensated. For this compensation, it also includes a temperature sensor that can be read from the I2C bus, which would allow checking the internal temperature of the oscilloscope. This one can be found for approx. $3,80 at the time of writing. Modifying the battery charging circuitry is also convenient in this module, as well as in the first one (see next section).

The third one costs about $2,20 and is also based on DS3231, but this time it doesn't include an EEPROM. Its advantage is that it has the smallest footprint (barely bigger than the IC itself) and a rechargeable battery. It's so small that could be fitted on top the MCU so the internal temperature sensor could be used to check the MCU's temperature if overclocking, for example. This module doesn't need any modifications, as there's no EEPROM or external charging circuitry.



MODULE MODIFICATIONS

If the module has an EEPROM at the same address than the one in the scope (0x50), there would be conflicts when installed in the scope, so modifications are needed. EEPROM's I2C address of this kind is set by grounding or pulling up pins 1-3 (A0-A2, the 3 least significant bits of the address), according to the datasheet. Base address is 0x50, so if these pins are grounded, this would be the EEPROM's address. In this case, it's necessary to unsolder the EEPROM, or pulling up one or more of these pins to change its address. For example, in the photo you can see pin nº 1 lifted from the PCB and pulled up to Vcc with a 10k resistor.

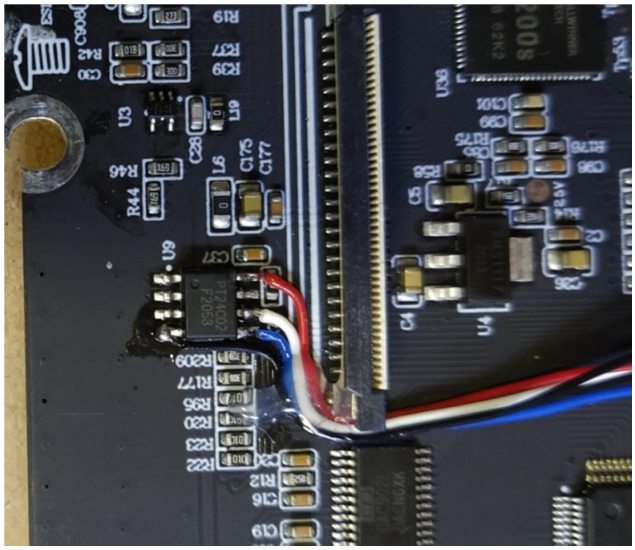


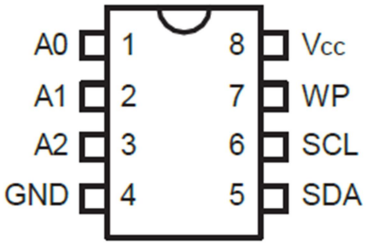
For the first two modules, if using a non-rechargeable coin cell, it's also convenient to modify the battery charging circuit, by removing the diode and/or resistor in series (D1-R5 in the DS1307 module, unlabelled in the second DS3231 module). For the DS1307 module in the photo, shorting R6 and removing R4 is also convenient to improve battery life (these don't exist in the second DS3231 module). See here and here for more information on the modification and schematics of these modules.

MODULE INSTALLATION

In order to connect the RTC module to the oscilloscope's I2C bus, I chose soldering cables to the pins of the serial EEPROM (U9), as the other possibilities (pins of the MCU or the FPGA) would have been much harder.

I used red and black for Vcc (3.3V) and GND and white and blue for SCL and SDA, respectively. These four cables go to the correspondent connections of the modules. I also used some hot melt glue to secure the cables to the PCB to avoid stressing the soldered connections:





The module should then be connected to these cables and fixed in an appropriate place; as this depends on the chosen module, I leave this to your imagination… hot glue or double sided adhesive tape are good options.

SOFTWARE

In order to get the oscilloscope working with the RTC, it's necessary to install the appropriate software for reading the RTC and setting it whenever it's needed (first installation, when changing from summer to winter Daylight Savings Time, etc). After several attempts, the final scripts written by DavidAlfa and published here are the best option. The easiest way to install them by using David's installation packages available at his Google Drive (instructions there in the readme file).

If you prefer manual installation from the console, you need to extract the files from those packages (or create them from the post) and copy them to the scope as follows:

- The script called *S11_date_daemon.sh* to launch the daemon that should be copied to */etc/init.d*
- The daemon itself, called *date_daemon*, should be copied to */usr/bin/*. If using DS3231, 2nd line must be commented out, and the reverse with the 3rd line. This is very important as, leaving the script as is with DS3231 would result in overwriting control registers with random information with unpredictable results
- As the daemon calls them, David's statically compiled I2C tools must also be copied to /usr/bin (at least, *i2cget*; *i2cset* and i2cdetect), and to /usr/lib (*libi2c.so.0*).

The daemon reads the RTC and sets system date and time on boot time, and monitors the USB drive for a file called "date". If it exists and its timestamp is different from previous occasions, then sets the RTC and system time according to the file's timestamp. So, to set the time, you just need to create or update a file called "date" in the root directory of the USB pendrive (can be empty, file contents doesn't matter) and plug it in the scope.

BACKGROUND

The daemon was written for the DS3231 and the DS1307, The first one has the following registers:

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

The daemon reads and writes the registers associated with time and date, 00h to 06h. From the datasheet:

> *"The time and calendar information is obtained by reading the appropriate register bytes. Figure 1 illustrates the RTC registers. The time and calendar data are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the binary-coded decimal (BCD) format. The DS3231 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic-high being PM. In the 24-hour mode, bit 5 is the 20-hour bit (20–23 hours). The century bit (bit 7 of the month register) is toggled when the years register overflows from 99 to 00."*

The equivalent table for the DS1307 is practically the same as far as time registers are concerned, so the script probably will work also OK with this chip:

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12 24	10 Hour PM/ AM	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

> *"The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12-hour or 24-hour mode-select bit. When high, the 12-hour mode is selected.*

In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 to 23 hours). The hours value must be re-entered whenever the 12/24-hour mode bit is changed."

Finally, as can be seen in the previous table, the DS1307 has 56 bytes of RAM from address 08 onwards. The daemon takes advantage of this to store the information needed for date setting. But this zone in the DS3231 contains control registers, so the daemon uses scope's internal flash when working with this chip.