```
1  ##########################################
2  ### This Code is for Raspberry Pi Pico ###
3  ###      copyright 2021 balance19      ###
4  ##########################################
5
6  import machine
7
8  def int_to_bcd(val):
9      """Convert an integer to binary coded decimal (BCD)"""
10     return ((val // 10) << 4) + (val % 10)
11
12 # Class for getting Realtime from the DS3231 in different modes.
13 class Clock:
14     w = ["FRI", "SAT", "SUN", "MON", "TUE", "WED", "THU"]
15     # If you want different names for Weekdays, feel free to add. Couple examples below:
16     # w = ["FR", "SA", "SU", "MO", "TU", "WE", "TH"]
17     # w = ["Friday", "Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday"]
18     # w = ["Freitag", "Samstag", "Sonntag", "Montag", "Dienstag", "Mittwoch", "Donnerstag"]
19     # w = ["viernes", "sabado", "domingo", "lunes", "martes", "miercoles", "jueves"]
20
21     @classmethod
22     def get_default_clock(cls):
23         sda_pin = 18
24         scl_pin = 19
25         port = 1
26         speed = 100000
27         address = 0x68
28         register = 0x00
29         return Clock(sda_pin, scl_pin, port, speed, address, register)
30
31     # Initialisation of RTC object. Several settings are possible but everything is optional.
32     # If you meet these standards no parameters are required.
33     def __init__(self, sda_pin, scl_pin, port, speed, address, register):
34         self.rtc_address = address  # for using different i2c address
35         self.rtc_register = register  # for using different register on device. DON'T change for
DS3231
36         sda = machine.Pin(sda_pin)  # configure the sda pin
37         scl = machine.Pin(scl_pin)  # configure the scl pin
38         self.i2c = machine.I2C(port, sda=sda, scl=scl)  # configure the i2c interface with given
parameters
39
40         self.setup()
41
42     def setup(self, reset=False):
43         now = self.get_time()
44         if now is None:
45             return
46         # else:
47         #     if now[6] < 2024 or reset:
48         #         print("Time not set, time must be set to initialize scheduler")
49         #         self.serial_entry()
50
51     # Method for setting the Time
52     def set_time(self, NowTime=b"\x00\x23\x12\x28\x14\x07\x21"):
53         # NowTime has to be in format like b'\x00\x23\x12\x28\x14\x07\x21'
54         # This is the time 10:53:00, Thursday, 10.06.2024 so it's b"\x00\x53\x10\x28\x14\x06\x24"
55         # It is encoded like this           sec min hour week day month year
56         # Then it's written to the DS3231
57         self.i2c.writeto_mem(int(self.rtc_address), int(self.rtc_register), NowTime)
58
59     def serial_entry(self):
60         print("Please enter the time in the following format:")
61         print("sec min hour weekday month day year")
62         print("Example: 00 53 10 4 6 10 24")
```

```
        print("This is the time 10:53:00, Thursday, 10.06.2024")
        print("Weekday: 0=Friday, 1=Saturday, 2=Sunday, 3=Monday, 4=Tuesday, 5=Wednesday,
6=Thursday")
        print("Month: 1=January, 2=February, 3=March, 4=April, 5=May, 6=June, 7=July, 8=August,
9=September, 10=October, 11=November, 12=December")
        print("Year: 00-99")
        print("Please enter the time now:")
        print("SS MM HH WD MM DD YY")

        try:
            sec, minute, hour, weekday, month, day, year = [int(x) for x in input().split()]
            self.set_time_piece_by_piece(sec, minute, hour, weekday, month, day, year)
        except Exception as e:
            print("Error: %s" % e)

    def set_time_piece_by_piece(self, sec, minute, hour, weekday, month, day, year):
        try:
            # Convert each component to BCD
            bcd_sec = int_to_bcd(sec)
            bcd_minute = int_to_bcd(minute)
            bcd_hour = int_to_bcd(hour)
            bcd_weekday = int_to_bcd(weekday)
            bcd_day = int_to_bcd(day)
            bcd_month = int_to_bcd(month)
            bcd_year = int_to_bcd(year)

            # Create the byte array in the required format
            NowTime = bytes([bcd_sec, bcd_minute, bcd_hour, bcd_weekday, bcd_day, bcd_month,
bcd_year])
            print("NowTime : ", NowTime)
            self.set_time(NowTime)
            print("Time set successfully")
        except Exception as e:
            print("Error setting time: %s" % e)

    # DS3231 gives data in bcd format. This has to be converted to a binary format.
    def bcd2bin(self, value):
        return (value or 0) - 6 * ((value or 0) >> 4)

    # Add a 0 in front of numbers smaller than 10
    def pre_zero(self, value):
        pre_zero = True  # Change to False if you don't want a "0" in front of numbers smaller than
10
        if pre_zero:
            if value < 10:
                value = f"0{value}"  # From now on the value is a string!
        return value

    # Read the Realtime from the DS3231 with errorhandling. Currently two output modes can be used.
    def get_time(self, mode=0):
        try:
            # Read RT from DS3231 and write to the buffer variable. It's a list with 7 entries.
            # Every entry needs to be converted from bcd to bin.
            buffer = self.i2c.readfrom_mem(self.rtc_address, self.rtc_register, 7)
            # The year consists of 2 digits. Here 2000 years are added to get format like "2021"
            year = self.bcd2bin(buffer[6]) + 2000
            month = self.bcd2bin(buffer[5])  # Just put the month value in the month variable and
convert it.
            day = self.bcd2bin(buffer[4])  # Same for the day value
            # Weekday will be converted in the weekdays name or shortform like "Sunday" or "SUN"
            weekday = self.w[self.bcd2bin(buffer[3]) % 7]
            # Uncomment the line below if you want a number for the weekday and comment the line
before.
            # weekday = self.bcd2bin(buffer[3])
            hour = self.pre_zero(self.bcd2bin(buffer[2]))  # Convert bcd to bin and add a "0" if
```

```
necessary
            minute = self.pre_zero(self.bcd2bin(buffer[1]))  # Convert bcd to bin and add a "0" if
necessary
            second = self.pre_zero(self.bcd2bin(buffer[0]))  # Convert bcd to bin and add a "0" if
necessary
            if mode == 0:  # Mode 0 returns a list of second, minute, ...
                return second, minute, hour, weekday, month, day, year
            if mode == 1:  # Mode 1 returns a formated string with time, weekday and date
                time_string = f"{hour}:{minute}:{second}      {weekday} {month}.{day}.{year}"
                return time_string
            # If you need different format, feel free to add

        except Exception as e:
            print("Error: in the DS3231 not connected or some other problem: %s" % e)
            return None


    def manual(self):
        print("Clock Manual Control")
        while True:
            print("1. Set Time")
            print("2. Get Time")
            print("3. Exit")
            choice = int(input("Enter choice: "))
            if choice == 1:
                self.serial_entry()
            elif choice == 2:
                print("Time :")
                print("Format : sec min hour weekday month day year")
                print(self.get_time())
            elif choice == 3:
                return
            else:
                print("Invalid Input")

##########################################
### This Code is for Raspberry Pi Pico ###
###      copyright 2021 balance19      ###
##########################################

# from my_lib import RTC_DS3231
if __name__ == "__main__":
    import time

    # Initialisation of RTC object. Several settings are possible but everything is optional.
    # If you meet the standards (see /my_lib/RTC_DS3231.py) no parameters are needed.
    clock = Clock.get_default_clock()
    clock.manual()
    # clock.set_time_piece_by_piece(0, 53, 10, 4, 6, 10, 24)

    # It is encoded like this: sec min hour week day month year.
    # Uncomment the line below to set time. Do this only once otherwise time will be set everytime
the code is executed.


    # This is the time 10:53:00, Thursday, 10.06.2024 so it's b"\x00\x53\x10\x28\x14\x06\x24"
    # It is encoded like this           sec min hour week day month year
    # rtc.DS3231_SetTime(b"\x00\x54\x08\x24\x11\x06\x24")  # Set time to 10:53:00, Thursday,
10.06.2024

    while True:
        t = clock.get_time()  # Read RTC and receive data in Mode 1 (see /my_lib/RTC_DS3231.py)
        print(t)
        time.sleep(1)
```

```
181
```