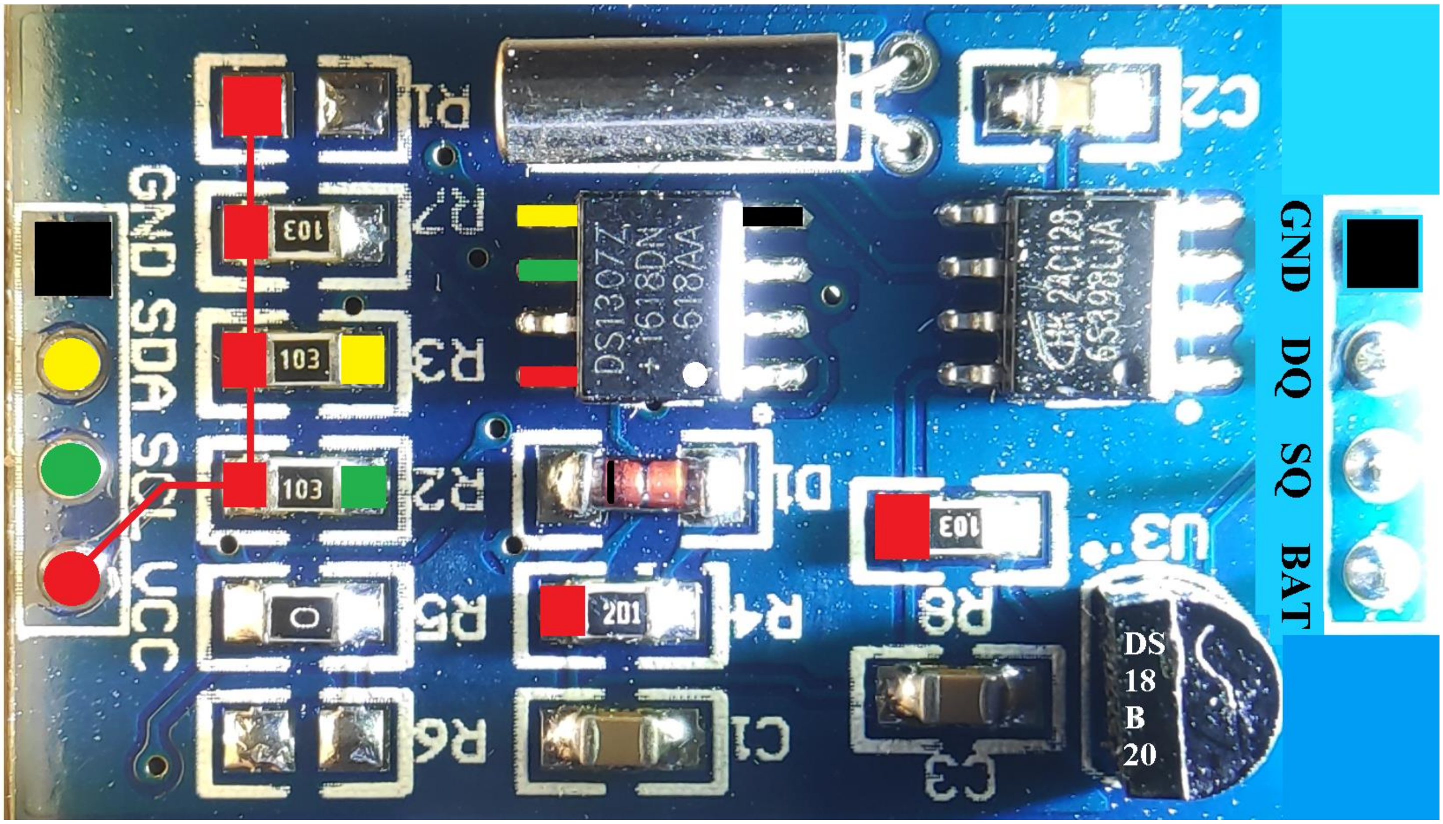
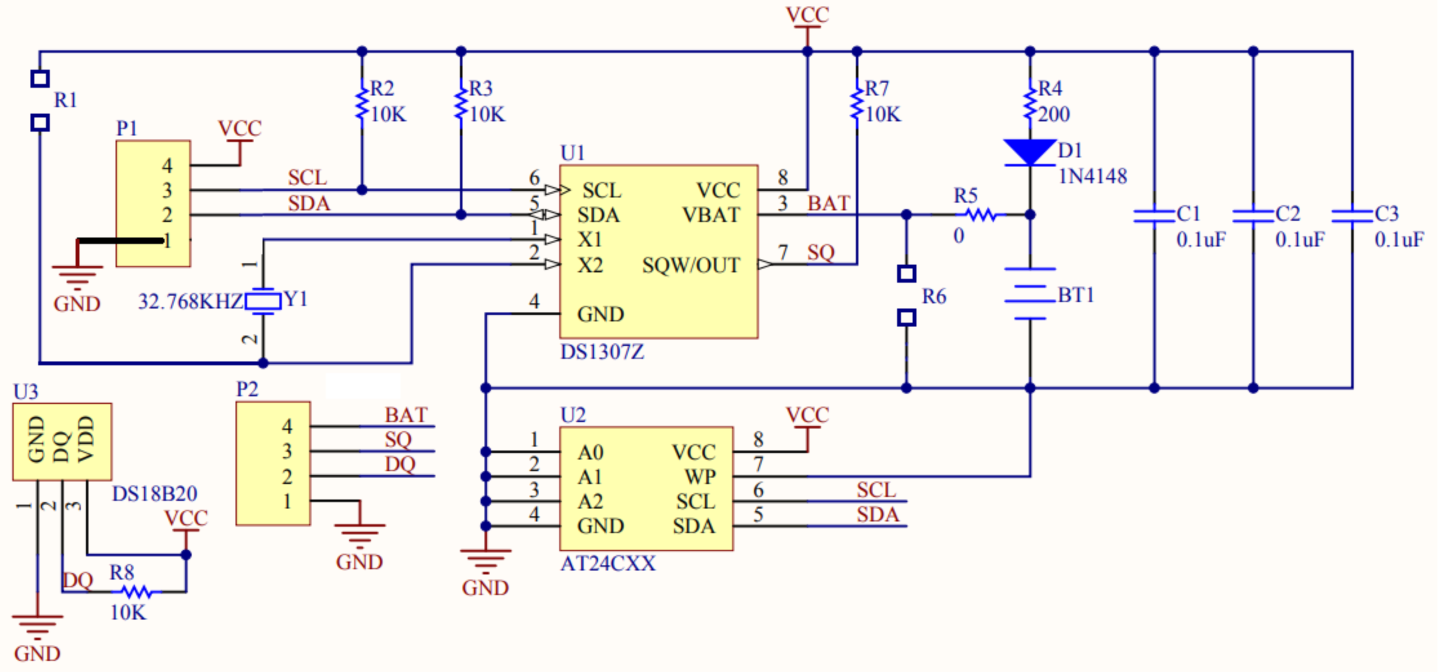
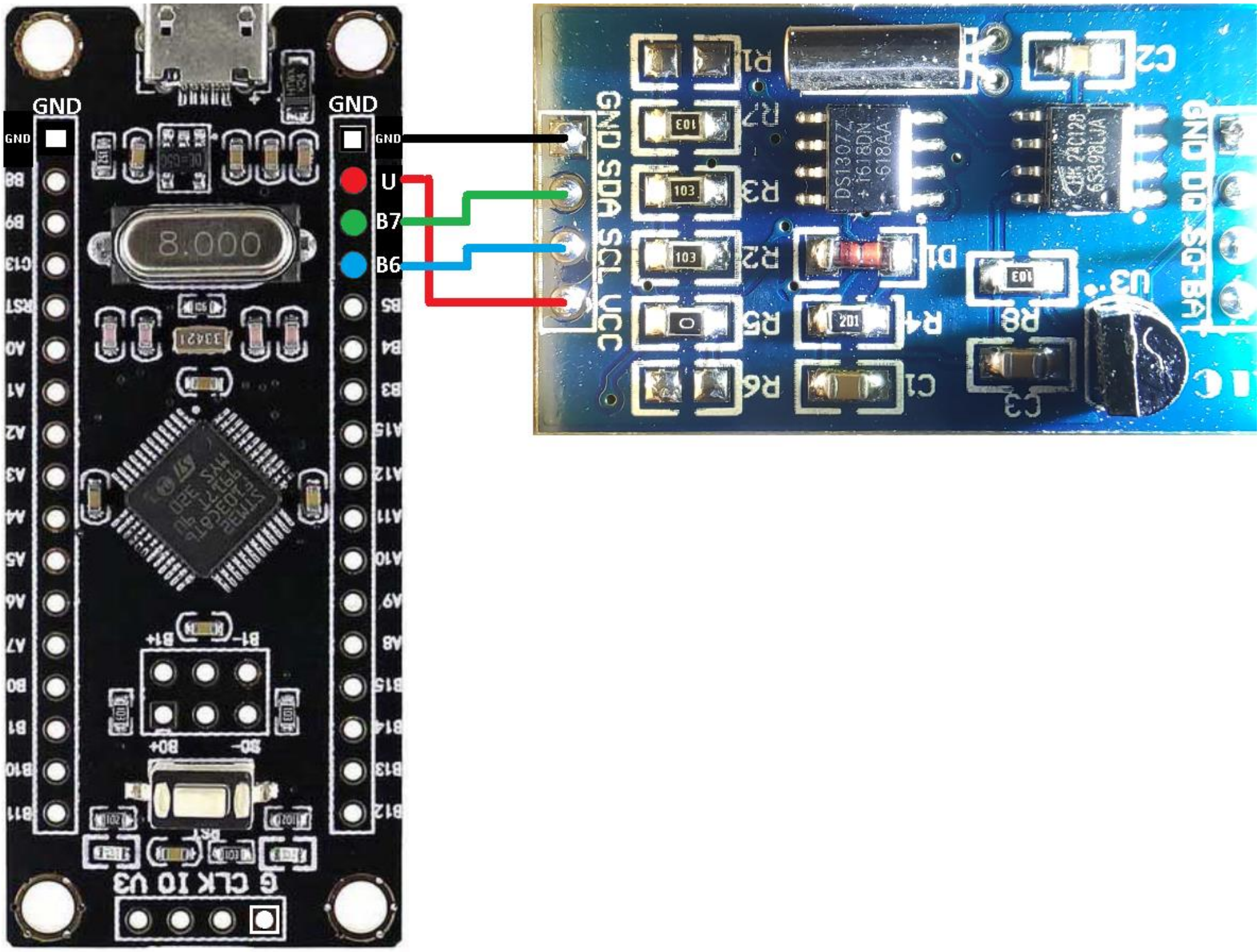
DS1307 – 24C128 – DS18B20 – Module

We are only interested in RTC-chip DS1307. We do **not** use memory chip 24C128 and digital thermometer IC DS18B20, and we do **not** use any battery on the back side of the module. We only use the connector on the left side: GND, SDA, SCL, VCC (U = 3,3V).





For I2C and USB communication we use a Black Pill (STM32F103C8T6)



We only write and read the seven yellow timekeeper registers 0x00 up to 0x06.

“When you power up the module the clock halt (CH) bit in the **seconds** register will be set to a 1.”

That means, that we start the oscillator by setting the time in the DS1307-chip, because the value of the seconds sets 0x00-Bit7 = 0.

“The DS1307 serial real-time clock (RTC) is a low power, full binary-coded decimal (BCD) clock/calendar.”

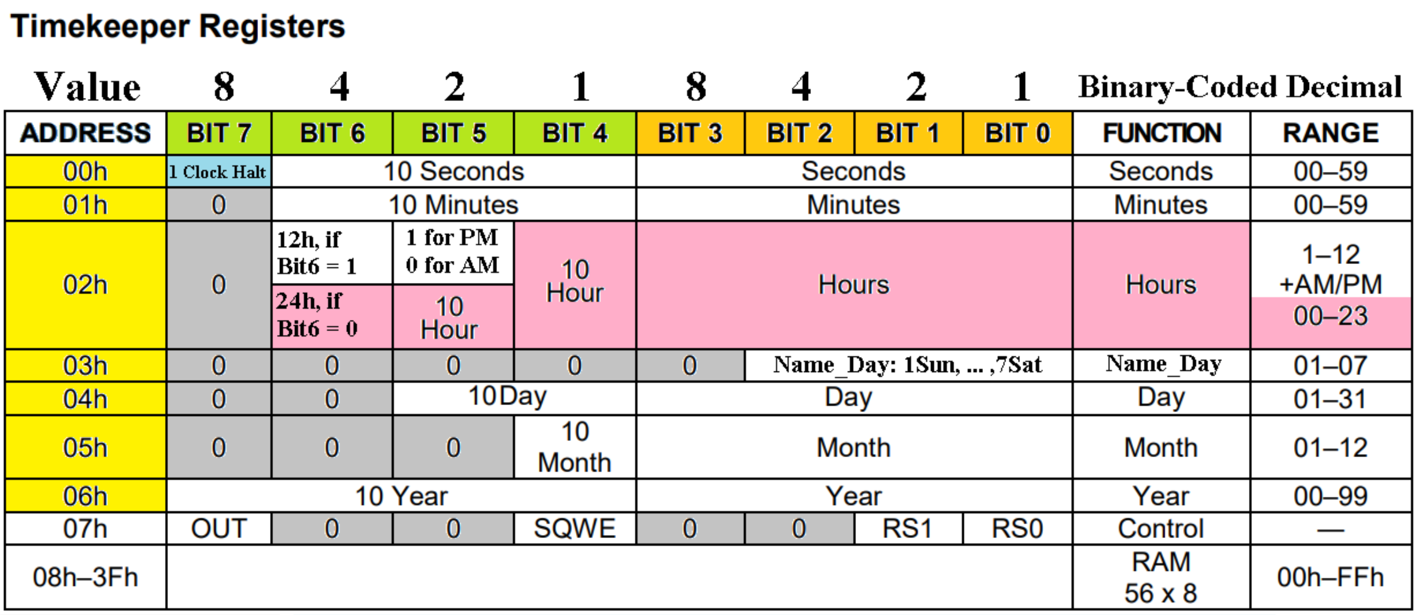
That means, that the four bits Bit7, Bit6, Bit5, Bit4 are used for the **tens (10, 20, …, 90)**, and for the **units**

(0, 1, 2, …, 9) DS1307 is using Bit3, Bit2, Bit1, Bit0.

If we write a decimal value of 59 seconds into the **seconds** register 0x00, we have to transmit via I2C

01011001. Side effect is, that Bit7 is zero, so we start the RTC oscillator with any value for the seconds $0 \leq sec \leq 59$.

For the hours let’s use the 24h modus.



Timekeeper Registers

Value	8	4	2	1	8	4	2	1	Binary-Coded Decimal	
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	1 Clock Halt	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12h, if Bit6 = 1 24h, if Bit6 = 0	1 for PM 0 for AM 10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
03h	0	0	0	0	0	Name_Day: 1Sun, ... ,7Sat			Name_Day	01–07
04h	0	0	10Day		Day				Day	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

0 = Always reads back as 0. **Bit7 of Register 0x00 is the clock halt (CH) bit. When this bit is set to 1, the oscillator is disabled.**

Binary-Coded Decimal (BCD) means: 59_{dec}

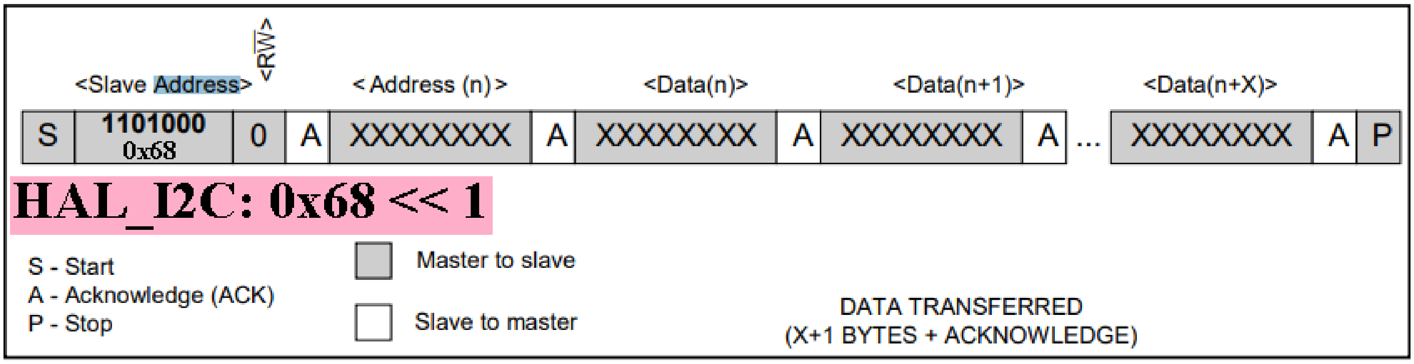
1	0	1	1	0	0	1
BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
4 +	0 +	1	8 +	0 +	0 +	1

In C-code we flip between decimals and binary coded decimals with this two definitions:

```
#define BCDtoDEC(x) ((x >> 4) * 10 + (x & 0x0F))
#define DECtoBCD(x) (((x / 10) << 4) | (x % 10))
```

The DS1307-datasheet says, that the DS1307-address is 1101000 = 0x68,

Figure 4. Data Write – Slave Receiver Mode



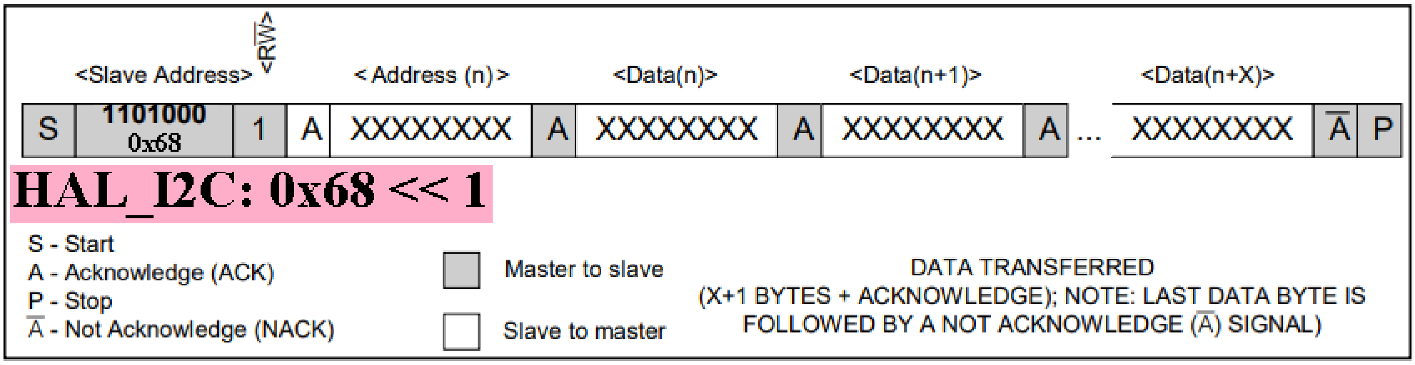
and figure 4 of the datasheet tells us how to write via I2C to the RTC.

```
uint8_t data[7];
HAL_I2C_Mem_Write(&hi2c1, 0x68 << 1, 0x00, 1, data, 7, HAL_MAX_DELAY);
```

Later we put some values into array **data**.

The datasheet as well shows a figure to read from the DS1307-chip.

Figure 5. Data Read – Slave Transmitter Mode



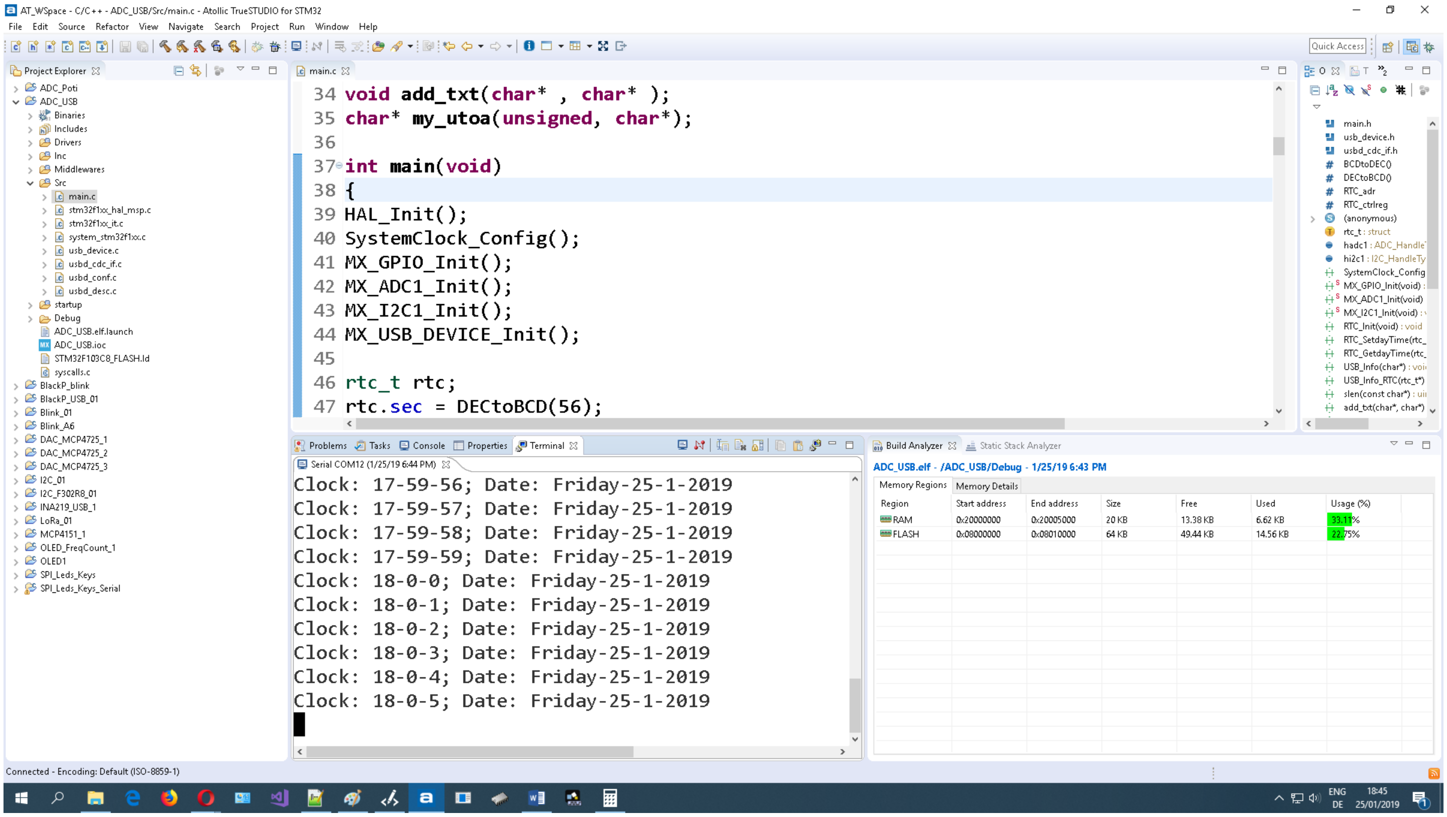
We do this with the C-code

```
uint8_t data[] = {0,0,0,0,0,0,0};
HAL_I2C_Mem_Read(&hi2c1, 0x68 << 1, 0x00, 1, data, 7, HAL_MAX_DELAY);
```

To make the code more readable, we introduce a composite data type **rtc_t** for the seven timekeeper registers:

```
typedef struct {//59dec is 0101 for 5 and 1001 for 9, or 01011001BCD
  uint8_t sec;
  uint8_t min;
  uint8_t hour;
  uint8_t name_day;//e.g.: 1 for Sunday, up to 7 for Saturday
  uint8_t day;
  uint8_t month;
  uint8_t year;
} rtc_t;
```

We get the following output:



```
void add_txt(char* , char* );
char* my_utoa(unsigned, char*);

int main(void)
{
HAL_Init();
SystemClock_Config();
MX_GPIO_Init();
MX_ADC1_Init();
MX_I2C1_Init();
MX_USB_DEVICE_Init();

rtc_t rtc;
rtc.sec = DECtoBCD(56);
```

```
Clock: 17-59-56; Date: Friday-25-1-2019
Clock: 17-59-57; Date: Friday-25-1-2019
Clock: 17-59-58; Date: Friday-25-1-2019
Clock: 17-59-59; Date: Friday-25-1-2019
Clock: 18-0-0; Date: Friday-25-1-2019
Clock: 18-0-1; Date: Friday-25-1-2019
Clock: 18-0-2; Date: Friday-25-1-2019
Clock: 18-0-3; Date: Friday-25-1-2019
Clock: 18-0-4; Date: Friday-25-1-2019
Clock: 18-0-5; Date: Friday-25-1-2019
```

ADC_USB.elf - /ADC_USB/Debug - 1/25/19 6:43 PM

Memory Regions | Memory Details

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20005000	20 KB	13.38 KB	6.62 KB	33.11%
FLASH	0x08000000	0x08010000	64 KB	49.44 KB	14.56 KB	22.75%

ST-CubeMX generates most of the code as shown in https://www.mikrocontroller.net/attachment/388280/Bulb_Ohm.pdf , and with Atollic TrueSTUDIO we add these lines of gray highlighted code:

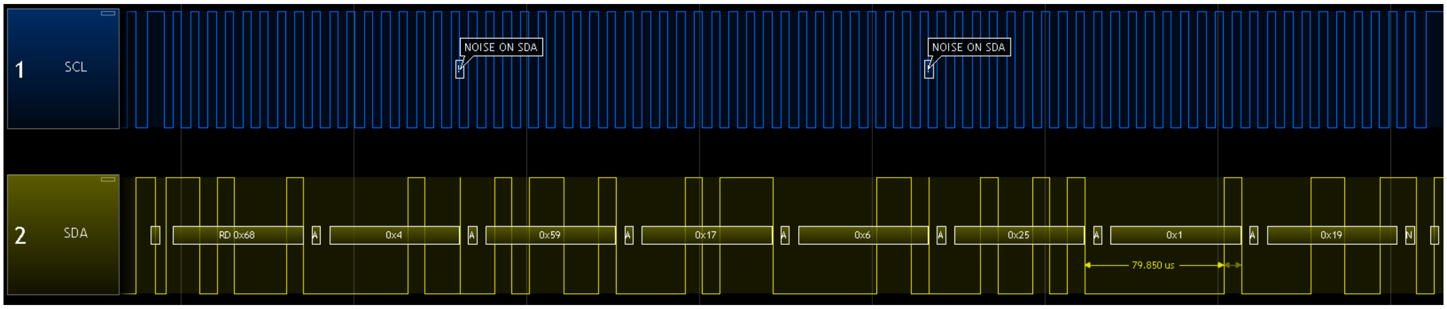
```
#include "main.h"
#include "usb_device.h"
#include "usbd_cdc_if.h"//By reason of: CDC_Transmit_FS(uint8_t*, uint16_t);

#define BCDtoDEC(x) ((x >> 4) * 10 + (x & 0x0F))
#define DECtoBCD(x) (((x / 10) << 4) | (x % 10))

typedef struct {//59dec is 0101 for 5 and 1001 for 9, or 01011001BCD
  uint8_t sec;
  uint8_t min;
  uint8_t hour;
  uint8_t name_day;//e.g.: 1 for Sunday, up to 7 for Saturday
  uint8_t day;
  uint8_t month;
  uint8_t year;
} rtc_t;

I2C_HandleTypeDef hi2c1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);

void RTC_Set_Time_Date(rtc_t*);
void RTC_Get_Time_Date(rtc_t*);
void USB_Info(char*);
void USB_Info_RTC(rtc_t*);
uint16_t slen(const char*);
void add_txt(char* , char* );
char* my_utoa(unsigned, char*);

int main(void)
{
HAL_Init();
SystemClock_Config();
MX_GPIO_Init();
MX_I2C1_Init();
MX_USB_DEVICE_Init();

rtc_t rtc;
rtc.sec = DECtoBCD(56);
rtc.min = DECtoBCD(58);
rtc.hour = DECtoBCD(17);
rtc.name_day = 6;//e.g.: 1 for Sunday, up to 7 for Saturday
rtc.day = DECtoBCD(25);
rtc.month = 1;
rtc.year = DECtoBCD(19);

//Begin: Exercise 2
uint8_t x = 0x93;//Set register 0x07 to 0x93 = 10010011 for 32.768kHz sq-wave;
HAL_I2C_Mem_Write(&hi2c1, 0x68 << 1, 0x07, 1, &x, 1, HAL_MAX_DELAY);
//End: Exercise 2

RTC_Set_Time_Date(&rtc);
while (1){
  RTC_Get_Time_Date(&rtc);
  USB_Info_RTC(&rtc);
  HAL_Delay(1000);
}
}

void RTC_Set_Time_Date(rtc_t *rtc)
{//DS1307_Address is 0x68; MAXIM_DS1307.pdf
    uint8_t data[7];
    data[0]=rtc->sec;
    data[1]=rtc->min;
    data[2]=rtc->hour;
    data[3]=rtc->name_day;//e.g.: 1 for Sunday, up to 7 for Saturday
    data[4]=rtc->day;
    data[5]=rtc->month;
    data[6]=rtc->year;
    if(!(HAL_I2C_Mem_Write(&hi2c1, 0x68 << 1, 0x00, 1, data, 7, HAL_MAX_DELAY) == HAL_OK))
      USB_Info("Error in RTC_Set_Time_Date");
}

void RTC_Get_Time_Date(rtc_t *rtc)
```

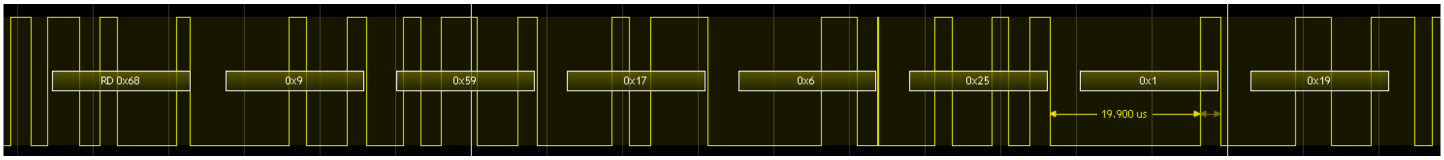
```
{//DS1307_Address is 0x68; MAXIM_DS1307.pdf
    uint8_t data[] = {0,0,0,0,0,0,0};
    if(HAL_I2C_Mem_Read(&hi2c1, 0x68 << 1, 0x00, 1, data, 7, HAL_MAX_DELAY) == HAL_OK){
    rtc->sec=BCDtoDEC(data[0]);
    rtc->min=BCDtoDEC(data[1]);
    rtc->hour=BCDtoDEC(data[2]);
    rtc->name_day=BCDtoDEC(data[3]);//e.g.: 1 for Sunday, up to 7 for Saturday
    rtc->day=BCDtoDEC(data[4]);
    rtc->month=BCDtoDEC(data[5]);
    rtc->year=BCDtoDEC(data[6]);
    } else {rtc->sec=1; rtc->min=2; rtc->hour=3; rtc->name_day=4; rtc->day=5; rtc->month=6; rtc->year=7;}
}

uint16_t slen(const char* s) {
    uint16_t i;
    for (i = 0; s[i] != 0; i++);
    return i;//s[0] not 0 then i=1;
}

void add_txt(char* out, char* in) {
   while (*out != 0) out++;
   while (*in != 0) {
      *out++ = *in++;
   }
   *out = 0;
}

char* my_utoa(unsigned val, char *str)
{
  //static char   buffer[10];
  char*   cp = str;
  unsigned v;
  char c;
  v = val;
  do {
    v /= 10;
    cp++;
  } while(v != 0);
  *cp-- = 0;
  do {
    c = val % 10;
    val /= 10;
    c += '0';
    *cp-- = c;
  } while(val != 0);
  return  cp;
}

void USB_Info(char *str)
{
  char txt[64] = {};
  add_txt( txt, str);
  add_txt( txt, "\n\r");
  CDC_Transmit_FS((uint8_t *)txt, slen(txt));
}

void USB_Info_RTC(rtc_t *rtc)
{
  char txt[128] = {}, h[32] = {};
  add_txt(txt, "Clock: ");
  my_utoa(rtc->hour, h);
  add_txt(txt, h); add_txt(txt, "-");
  my_utoa(rtc->min, h);
  add_txt(txt, h); add_txt(txt, "-");
  my_utoa(rtc->sec, h);
  add_txt(txt, h); add_txt(txt, "; Date: ");
  switch(rtc->name_day) {
  case 1:
      add_txt(txt, "Sunday"); add_txt(txt, "-"); break;
  case 2:
      add_txt(txt, "Monday"); add_txt(txt, "-"); break;
  case 3:
      add_txt(txt, "Tuesday"); add_txt(txt, "-"); break;
  case 4:
      add_txt(txt, "Wednesday"); add_txt(txt, "-"); break;
  case 5:
      add_txt(txt, "Thursday"); add_txt(txt, "-"); break;
  case 6:
      add_txt(txt, "Friday"); add_txt(txt, "-"); break;
  case 7:
      add_txt(txt, "Saturday"); add_txt(txt, "-"); break;
  }
  my_utoa(rtc->day, h);
  add_txt(txt, h); add_txt(txt, "-");
  my_utoa(rtc->month, h);
  add_txt(txt, h); add_txt(txt, "-20");
  my_utoa(rtc->year, h);
  add_txt(txt, h);
  add_txt(txt, "\n\r");
  CDC_Transmit_FS((uint8_t *)txt, slen(txt));
}

void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
```

```
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) Error_Handler();
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK) Error_Handler();
PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC|RCC_PERIPHCLK_USB;
PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL_DIV1_5;
if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) Error_Handler();
}
static void MX_I2C1_Init(void)
{
hi2c1.Instance = I2C1;
hi2c1.Init.ClockSpeed = 400000;
hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
hi2c1.Init.OwnAddress1 = 0;
hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
hi2c1.Init.OwnAddress2 = 0;
hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
if (HAL_I2C_Init(&hi2c1) != HAL_OK) Error_Handler();
}
static void MX_GPIO_Init(void)
{
__HAL_RCC_GPIOD_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();
}
void Error_Handler(void){}
#ifdef  USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line){}
#endif
```

Exercise 1: Change I2C speed from 100000 to 400000 and check the changes with a Logic Analyzer.

Answer: hi2c1.Init.ClockSpeed = 400000;

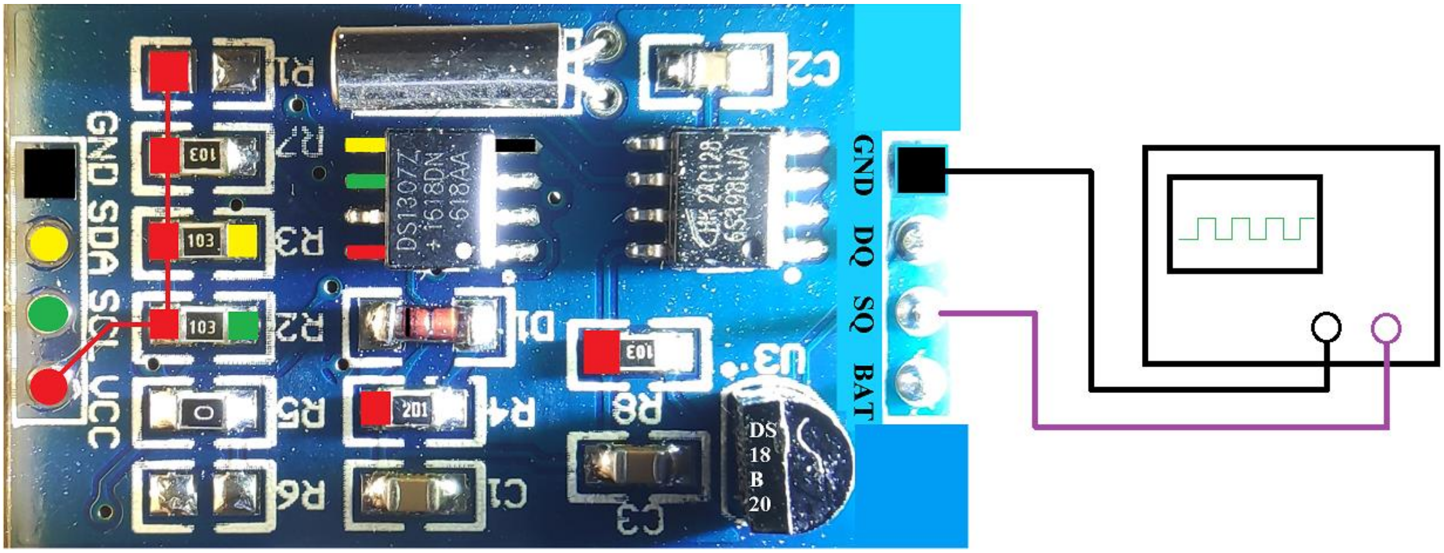




Exercise 2: The DS1307 control register 0x07 is used to control the operation of the SQW/OUT pin. Use the datasheet to generate a 32.768kHz square wave signal.

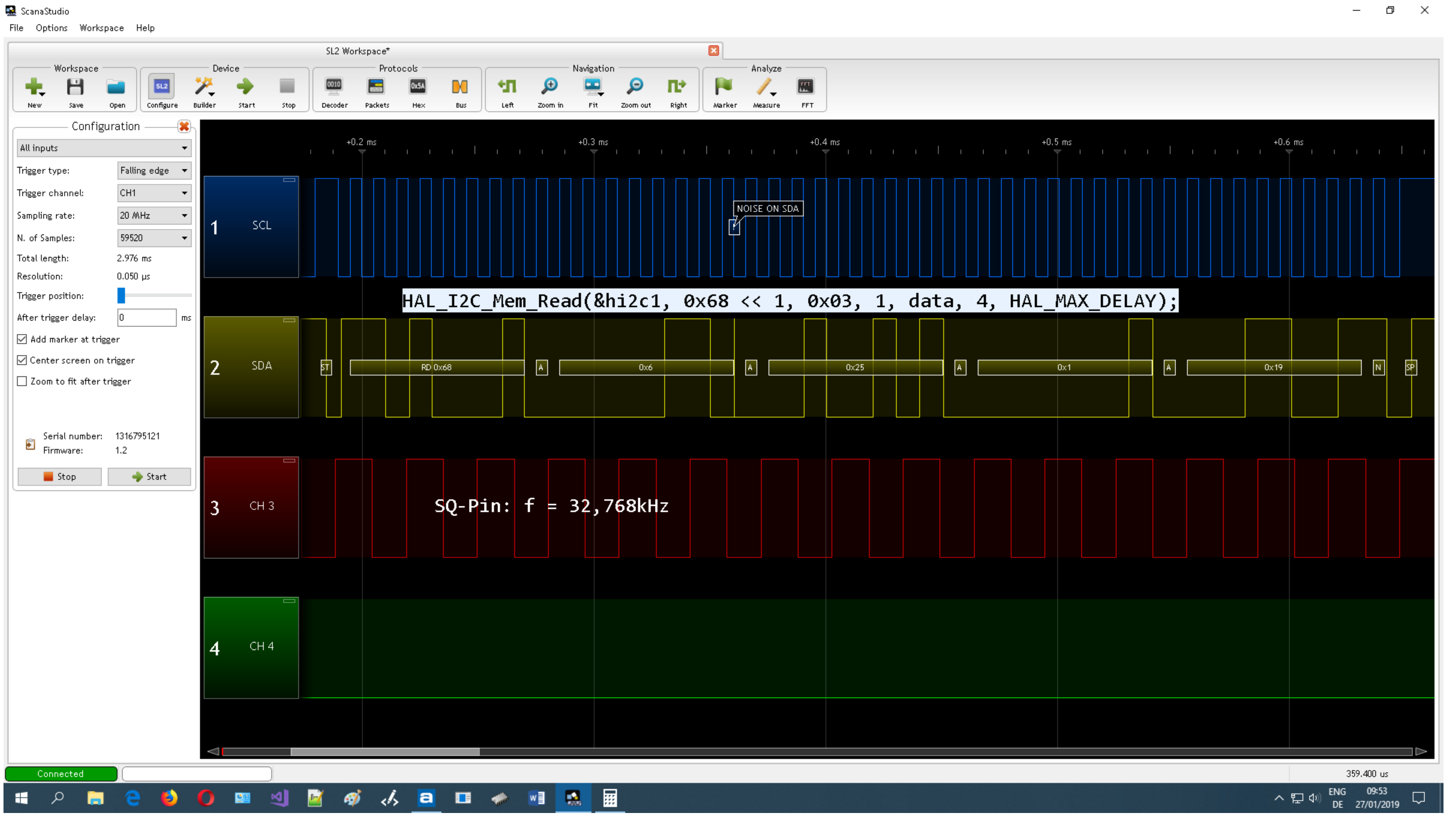
Answer:

```
…
rtc.year = DECtoBCD(19);
//Begin: Exercise 2
uint8_t x = 0x93;    //set register 0x07 to 0x93 = 10010011
HAL_I2C_Mem_Write(&hi2c1, 0x68 << 1, 0x07, 1, &x, 1, HAL_MAX_DELAY);
//End: Exercise 2
RTC_SetdayTime(&rtc);
while (1){
…
```



Exercise 3: C-code for to read only timekeeper registers 0x03, 0x04, 0x05, 0x06.

Answer:



ScanaStudio

SL2 Workspace*

Configuration

All inputs

Trigger type: Falling edge

Trigger channel: CH1

Sampling rate: 20 MHz

N. of Samples: 59520

Total length: 2.976 ms

Resolution: 0.050 µs

Trigger position:

After trigger delay: 0 ms

☑ Add marker at trigger

☑ Center screen on trigger

□ Zoom to fit after trigger

Serial number: 1316795121

Firmware: 1.2

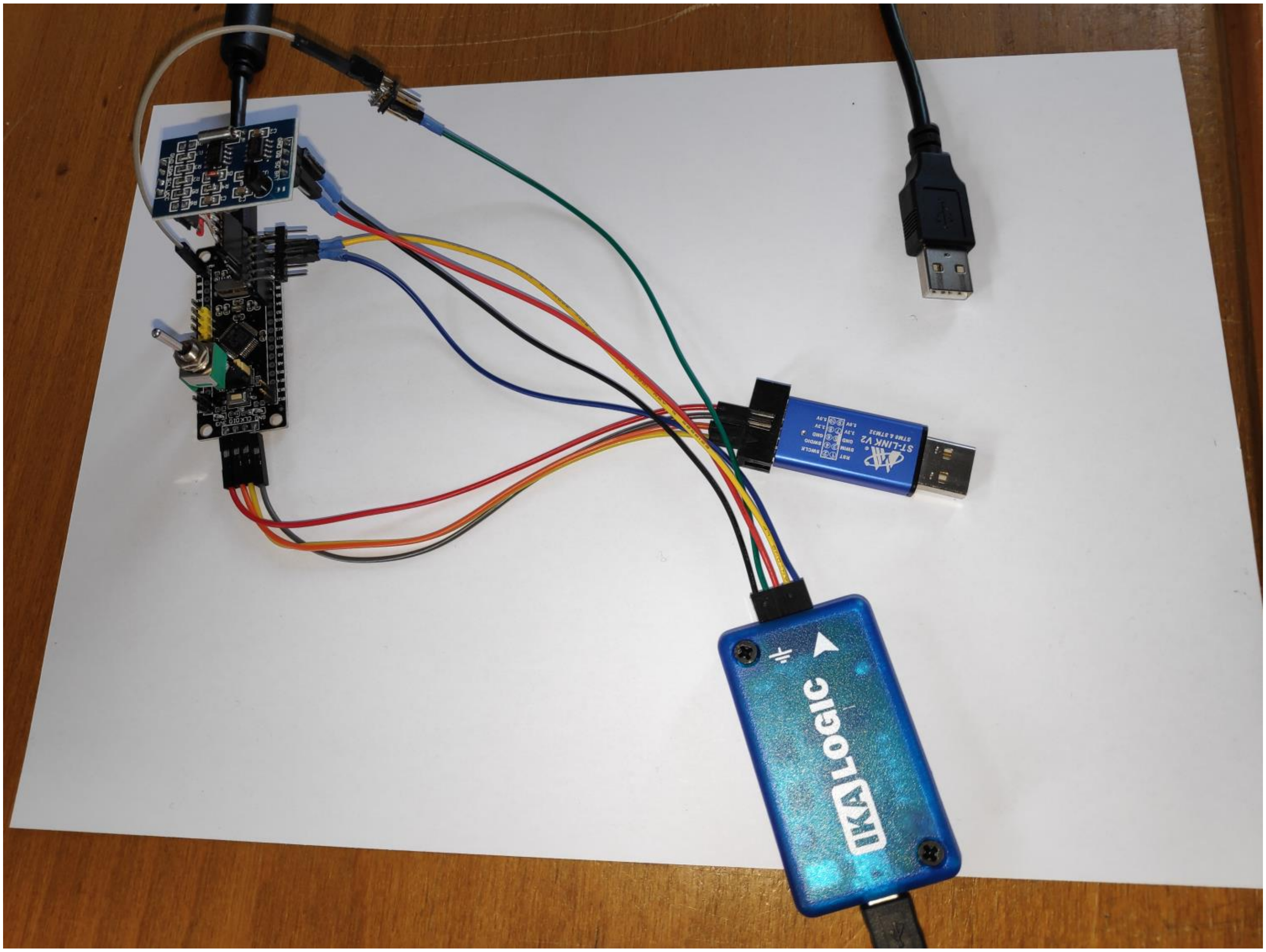
```
HAL_I2C_Mem_Read(&hi2c1, 0x68 << 1, 0x03, 1, data, 4, HAL_MAX_DELAY);
```

SQ-Pin: f = 32,768kHz

NOISE ON SDA

Connected

359.400 us



… have fun with STM32!

edgarmarx@t-online.de