

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Национальный исследовательский университет ИТМО»

ОТЧЕТ  
РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ВЗАИМОДЕЙСТВИЯ С ЧАСАМИ  
РЕАЛЬНОГО ВРЕМЕНИ В СТЕНДЕ SDK-1.1М.

Отчёт выполнил студент  
второго курса учебной  
группы Р4231  
Павлов Валентин  
Преподаватель: Ключев  
Аркадий Олегович

Санкт-Петербург  
2023 г.

# 1. ТЗ.

Программа для взаимодействия с часами реального времени в стенде SDK-1.1М.

## 1. Цель проекта:

Разработать программу для стенда SDK-1.1М, предназначенную для чтения и записи текущего времени и даты с использованием периферийных часов реального времени (RTC) MCP79411 и вывода их в консоль отладки среды CLab.

## 2. Требования к ПО:

### 2.1. Функциональные требования:

2.1.1. ПО должно обеспечивать инициализация устройства SDK-1.1М перед началом работы.

2.1.2. ПО должно обеспечивать чтение времени и даты из RTC по интерфейсу I2C.

2.1.3. ПО должно обеспечивать запись времени и даты в RTC по интерфейсу I2C.

2.1.4. ПО должно обеспечивать вывод полученных данных в консоль отладки (в среде CLab).

### 2.2. Нефункциональные требования:

2.2.1. Для написания ПО необходимо использовать IDE STM32CudeIDE.

2.2.2. Программа должна быть реализована на языке C.

2.2.3. Необходимо обеспечить надежное взаимодействие с периферийным устройством RTC MCP79411.

## 3. Состав системы:

### 3.1. Необходимые элементы стенда SDK-1.1М для работы программ:

3.1.1. Микроконтроллер STM32F407VGT6

3.1.2. Часы реального времени MCP7941

## 2. Архитектура системы.

Принципиальная схема элементов стенда, используемых для выполнения задачи стенда SDK-1.1М:

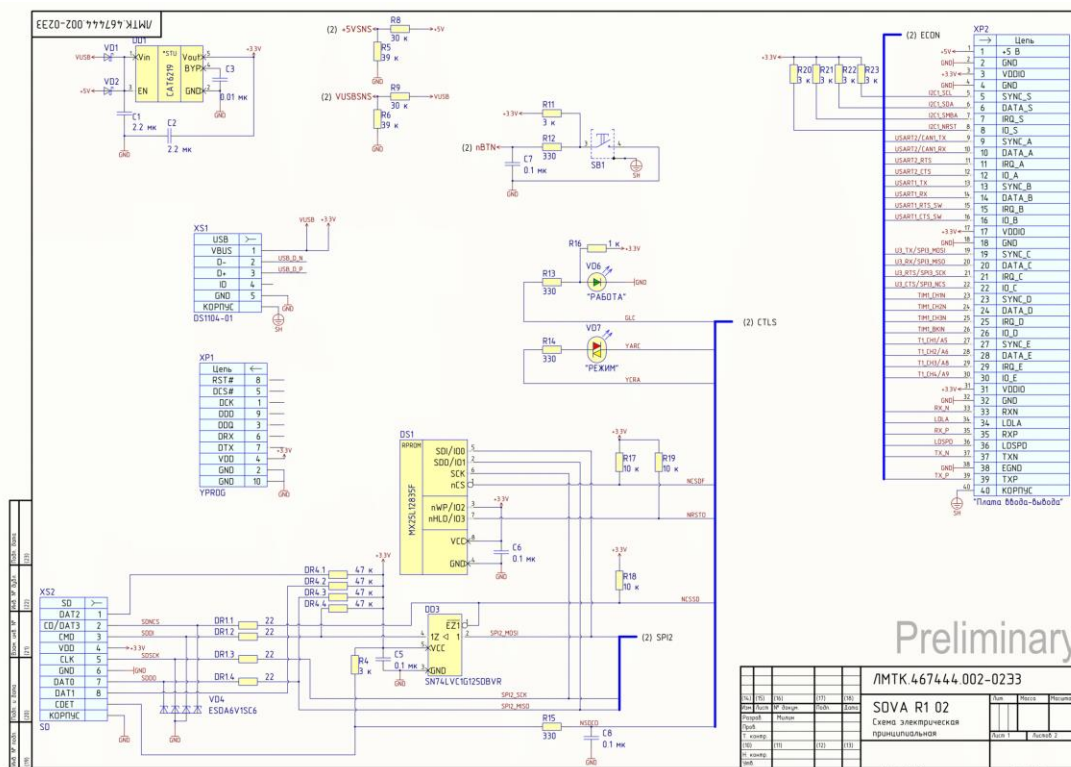


Рисунок 1 – схема подключения питания к стенду.

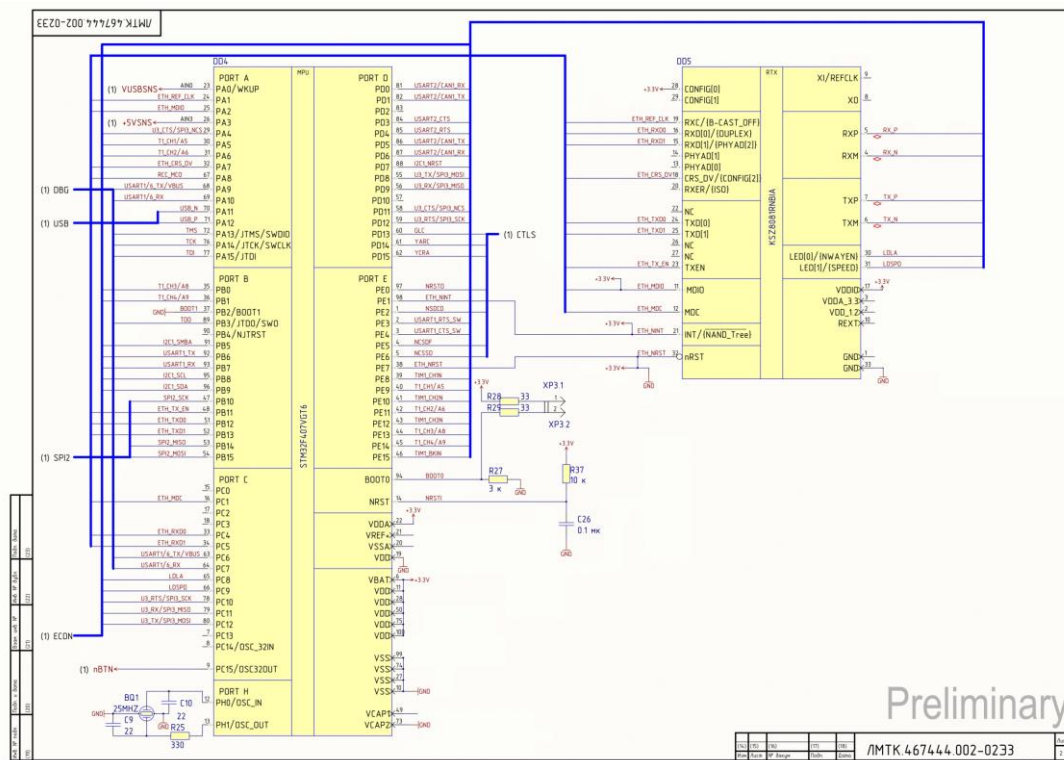
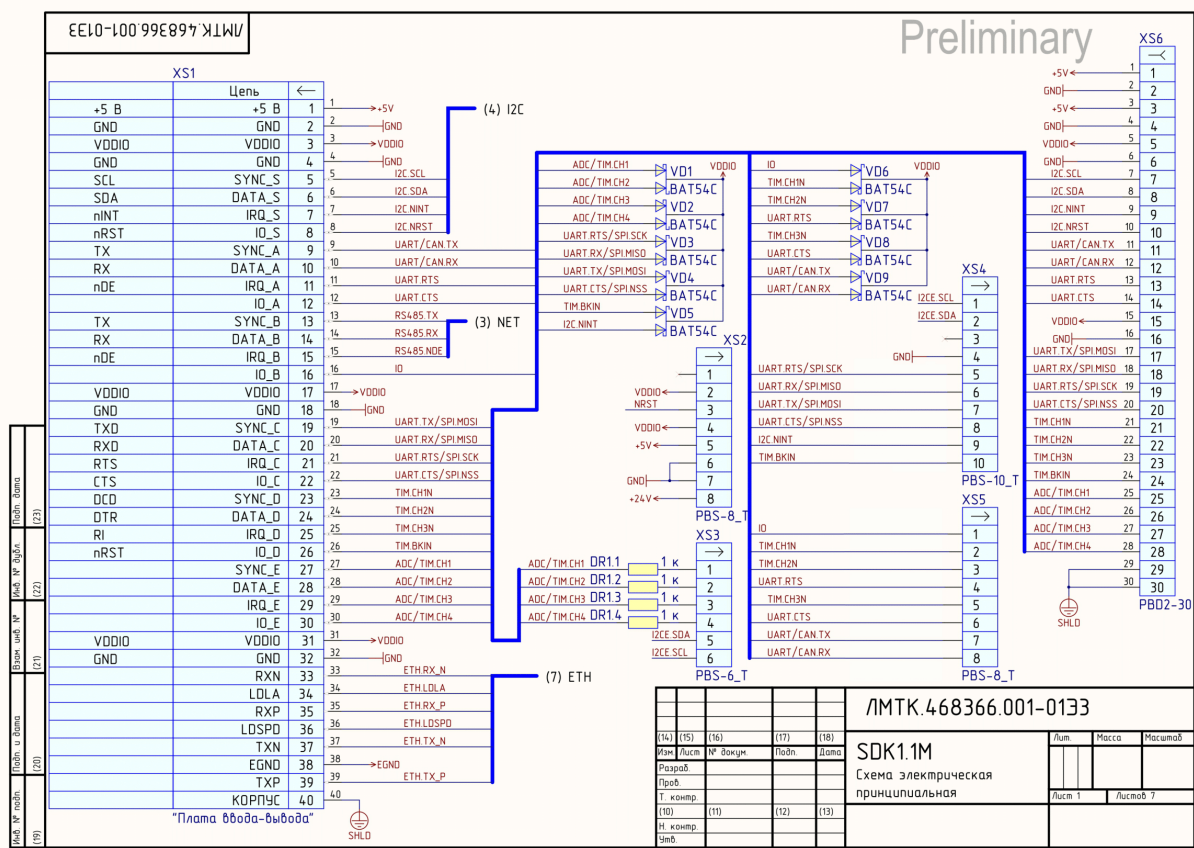


Рисунок 2 – схема подключения микроконтроллера STM32.



### 3. Разработка ПО.

Для разработки ПО в соответствии с ТЗ используется STM32CudeIDE, ПО разрабатывалось на языке C с использованием библиотек HAL.

#### 1. Настройка периферии:

Для работы с RTC используется интерфейс I2C. В данном проекте используется реализация интерфейса из библиотеки HAL. Для работы интерфейса необходимо произвести инициализацию (настройку) интерфейса:

```
void MX_I2C1_Init(void){
    hi2c1.Instance = I2C1; //Инициализация I2C
    hi2c1.Init.ClockSpeed = 400000; //Скорость работы I2C
    hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2; //Режим задержки
    hi2c1.Init.OwnAddress1 = 0; //Режим адресации
    hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT; //Режим адресации
    hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE; //Режим двухадресной
    hi2c1.Init.OwnAddress2 = 0; //Режим адресации
    hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE; //Режим общего вызова
    hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE; //Режим нестройки
    if (HAL_I2C_Init(&hi2c1) != HAL_OK){ //Инициализация I2C
        Error_Handler();
    }
}
```

Для чтения и записи даты и времени с RTC используются функции HAL:

HAL\_I2C\_Mem\_Read – для чтения с устройства

HAL\_I2C\_Mem\_Write – для записи на устройство

#### 2. Взаимодействие с RTC.

MCP79411 имеет 2 встроенных устройства RTCC и EEPROM, работа с разными устройствами осуществляется по разным адресам I2C:

‘1101111’(0x6F) – для с RTCC

‘1010111’(0x57) – для с EEPROM

И ещё бит R/W в конце адреса для обозначения чтения или записи на устройство.

Для записи и чтения с RTC реализованы функции:

```
// This function reads data from the MCP device using the I2C interface.
HAL_StatusTypeDef readMCP(uint8_t *rxData, uint16_t Size, uint16_t
MemAddress){
    return HAL_I2C_Mem_Read(&hi2c1, rtcAddress | 1, MemAddress,
I2C_MEMADD_SIZE_8BIT, rxData, Size, HAL_MAX_DELAY);}

// This function writes data to the MCP device using the I2C interface.
HAL_StatusTypeDef writeMCP(uint8_t *rxData, uint16_t Size, uint16_t
MemAddress){
    return HAL_I2C_Mem_Write(&hi2c1,rtcAddress & 0xFFFE, MemAddress,
I2C_MEMADD_SIZE_8BIT, rxData, Size, HAL_MAX_DELAY);}
```

Где rtcAddress – адрес RTC (0xDF), rxData – ссылка на буфер, Size – размер буфера, MemAddress – адрес в памяти устройства.

Данные о времени и дате хранятся в памяти RTC по адресам 00h – 07h в соответствии с картой памяти рисунок 1.

**TABLE 5-1: DETAILED RTCC REGISTER MAP**

Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Section 5.3 "Timekeeping"									
00h	RTCSEC	ST	SECTEN2	SECTEN1	SECTEN0	SECON3	SECON2	SECON1	SECON0
01h	RTCMIN	—	MINTEN2	MINTEN1	MINTEN0	MINONE3	MINONE2	MINONE1	MINONE0
02h	RTCHOUR	—	12/24	AM/PM HRTEN1	HRTEN0	HRONE3	HRONE2	HRONE1	HRONE0
03h	RTCWKDAY	—	—	OSCRUN	PWRFAIL	VBATEN	WKDAY2	WKDAY1	WKDAY0
04h	RTCDATE	—	—	DATETEN1	DATETEN0	DATEONE3	DATEONE2	DATEONE1	DATEONE0
05h	RTCMTH	—	—	LPYR	MHTTEN0	MTHONE3	MTHONE2	MTHONE1	MTHONE0
06h	RTCYEAR	YRTEN3	YRTEN2	YRTEN1	YRTEN0	YRONE3	YRONE2	YRONE1	YRONE0
07h	CONTROL	OUT	SQWEN	ALM1EN	ALM0EN	EXTOSC	CRSTRIM	SQWFS1	SQWFS0
08h	OSCTRIM	SIGN	TRIMVAL6	TRIMVAL5	TRIMVAL4	TRIMVAL3	TRIMVAL2	TRIMVAL1	TRIMVAL0
09h	EEUNLOCK	Protected EEPROM Unlock Register (not a physical register)							

Рисунок 1. – Карта памяти регистров времени RTC.

Для получения времени и даты с устройства реализована функция:

```
// This function retrieves the current date and time from the MCP device.
HAL_StatusTypeDef getDateTIme(RTC_TimeTypeDef *sTime, RTC_DateTypeDef *sDate);
```

Для получения данных из регистров секунд, минут, дня недели, даты, года используется маска в соответствии с картой памяти. Для получения данных о часах и месяцах используются вспомогательные функции:

```
// This function extracts the hour value from the register value of the MCP
device
uint8_t extractHours(uint8_t registerValue);

// This function extracts the month value from the register value of the MCP
device.
uint8_t extractMonth(uint8_t registerValue);
```

extractHours извлекает значение часов в зависимости от режима исчисления установленного в 12/24 бите и значения AM/PM бита в случае 12-часового режима. extractMonth извлекает значение месяца с учётом поправки на високосный год.

Для начала отсчёта в RTC необходимо установить стартовый бит (ST) для этого реализована функция:

```
// This function sets the ST (Start) bit of the MCP device.
HAL_StatusTypeDef setSTBit();
```

Для остановки отсчёта RTC необходимо сбросить ST бит, для этого реализована функция:

```
// This function clears the ST (Start) bit of the MCP device.
```



```
HAL_StatusTypeDef clearSTBit();
```

Работа счётчика в RTC отображается в бите OSCRUN, для считывания бита реализована функция:

```
// This function reads the OSCRUN bit from the MCP device.  
uint8_t readOSCRUNBit();
```

Также EXTOS бит указывает на источник тактирования устройства, где 0 – кварцевый резонатор подключённый к RTC, 1 – внешний источник тактирования. Поскольку в стенде SDK-1.1M используется внешний кварцевый резонатор реализована функция для сброса бита EXTOS:

```
// This function clears the EXTOSC bit of the MCP device.  
HAL_StatusTypeDef clearEXTOSCBit();
```

Установка времени и даты в регистры устройства реализована в функции:

```
// This function sets the date and time on the MCP device.  
HAL_StatusTypeDef setDateTime(RTC_TimeTypeDef *sTime, RTC_DateTypeDef *sDate);
```

Где RTC\_TimeTypeDef и RTC\_DateTypeDef структуры для хранения даты и времени определённые в драйвере RCC.

Функция для установки даты-времени работает по алгоритму, рекомендованному в документации на RTC.

1. Сбросить ST бит для остановки отсчёта, чтобы избежать “переворачивания” времени и даты.
2. Сбросить EXTOSC чтобы убедиться, что тактирование идёт от кварцевого резонатора.
3. Убедиться, что отсчёт остановлен и бит OSCRUN равен 0.
4. Записать в регистры даты и времени новые данные.
5. Установить ST бит для продолжения счёта.

Для формирования данных для записи в регистры RTC используется вспомогательная функция.

```
// This function forms the date and time data to be written to the MCP device.  
void formDateTimeData(uint8_t* data, RTC_TimeTypeDef *sTime, RTC_DateTypeDef *sDate);
```

Данные для записи формируются в соответствие с картой памяти устройства.

Основной цикл программы выглядит следующим образом:

```
for(int i=0; i<6; i++){  
    HAL_Delay(1000);  
    i2cStatus = getDateTime(&sTime, &sDate);  
    if (i2cStatus == HAL_OK){  
        SDK_TRACE_Print("Время и дата: %02d:%02d:%02d %02d/%02d/%02d\n",  
sTime.Hours, sTime.Minutes, sTime.Seconds, sDate.Date, sDate.Month,  
sDate.Year);  
    }  
    else {
```


```
        Error_Handler();  
    }  
    SDK_TRACE_Timestamp(PRINT, 0);  
}
```

В данном цикле 6 раз считывается дата и время с RTC и выводится в консоль среды CLab используя функцию SDK\_TRACE\_Print, между циклами выдерживается пауза в 1 секунду.



## 4. Тестирование.

Перед началом тестирования была запущена программа и считаны дата и время установленные на RTC рисунок 5.



```
/***** SDK-1.1M Trace start (30.10.2023 21:23:37) *****/  
1001.091: Время и дата: 00:00:00 01/01/01  
2002.091: Время и дата: 00:00:00 01/01/01  
3003.091: Время и дата: 00:00:00 01/01/01  
4004.091: Время и дата: 00:00:00 01/01/01  
5005.091: Время и дата: 00:00:00 01/01/01  
6006.091: Время и дата: 00:00:00 01/01/01  
/***** SDK-1.1M Trace stop *****/
```

Рисунок 5 – Результат выполнения программы до установки времени в RTC.

Как можно заметить время, и дата установлены в значения по умолчанию и отчёт не идёт. Затем для тестирования в часы реального времени была записана информация о дате и времени. Для этого реализована функция:

```
HAL_StatusTypeDef setCurrentDateTime(void){  
    RTC_TimeTypeDef sTime = {0};  
    RTC_DateTypeDef sDate = {0};  
    //Получение текущей даты и времени (замените это на реальное получение  
    времени и даты)  
    sTime.Hours = 23;           // Часы (от 0 до 23)  
    sTime.Minutes = 39;        // Минуты (от 0 до 59)  
    sTime.Seconds = 30;        // Секунды (от 0 до 59)  
    sTime.TimeFormat = RTC_HOURFORMAT_24; // Формат времени (12-часовой AM)  
  
    sDate.WeekDay = RTC_WEEKDAY_SUNDAY; // День недели (понедельник)  
    sDate.Date = 22;           // День месяца (от 1 до 31)  
    sDate.Month = 10;          // Месяц (от 1 до 12)  
    sDate.Year = 23;           // Год (последние две цифры)  
  
    return setDateTime(&sTime, &sDate);  
}
```

После добавления вызова данной функции в основную программу и её выполнения в CLab, были полученные следующие результаты при считывании с устройства рисунок 6.

```
/***** SDK-1.1M Trace start (30.10.2023 21:25:26) *****/  
1012.096: Время и дата: 23:39:30 22/10/23  
2013.096: Время и дата: 23:39:31 22/10/23  
3014.096: Время и дата: 23:39:32 22/10/23  
4015.096: Время и дата: 23:39:33 22/10/23  
5016.096: Время и дата: 23:39:34 22/10/23  
6017.096: Время и дата: 23:39:35 22/10/23  
/***** SDK-1.1M Trace stop *****/
```

Рисунок 6 – Результат выполнения программы после установки даты и времени в RTC.

Как можно заметить в RTC было установлено требуемое время и часы ведут верный отсчёт. Затем спустя некоторое время программа была запущена повторно и был получен следующий результат рисунок 7.

```
/***** SDK-1.1M Trace start (30.10.2023 22:18:53) *****/  
1001.081: Время и дата: 00:29:17 23/10/23  
2002.081: Время и дата: 00:29:18 23/10/23  
3003.081: Время и дата: 00:29:19 23/10/23  
4004.081: Время и дата: 00:29:20 23/10/23  
5005.081: Время и дата: 00:29:21 23/10/23  
6006.081: Время и дата: 00:29:22 23/10/23  
/***** SDK-1.1M Trace stop *****/
```

Рисунок 7 – Результат выполнение программы спустя некоторое время.

Как можно заметить, даже при завершении выполнения основной программы и перепрошивки контроллера часы вели корректный отсчёт.