**Assignment: Watch**

Learning goals 2

Introduction 2

Watch design overview 3

Assignments 5

Part A: implementing bit manipulation 5

Part B (optional): implementing the Watch on an Arduino 5

Appendix: Watch device registers 6

Configuration registers 6

0x20: Configuration register (CONF) 6

Time registers 6

0x21: Time value register MSB (Read/Write) 6

0x22: Time value register LSB (Read/Write) 6

Date registers 7

0x23: Date value register MSB (Read/Write) 7

0x24: Date value register LSB (Read/Write) 7

# Learning goals

* Bit manipulation (part A)
* I2C device programming (part B)
* General programming exercise (part C)

# Introduction

This assignment will program an I2C Watch device that stores the current time and date. The I2C device has the user registers as given in table 1.

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Definition** |
| 0x20 | CONFIG | Configuration |
| 0x21 | TIME\_BITS\_HIGH | Time value (MSB) |
| 0x22 | TIME\_BITS\_LOW | Time value (LSB) |
| 0x23 | DATE\_BITS\_HIGH | Date value (MSB) |
| 0x24 | DATE\_BITS\_LOW | Date value (LSB) |

*Table 1: User registers of I2C watch device*

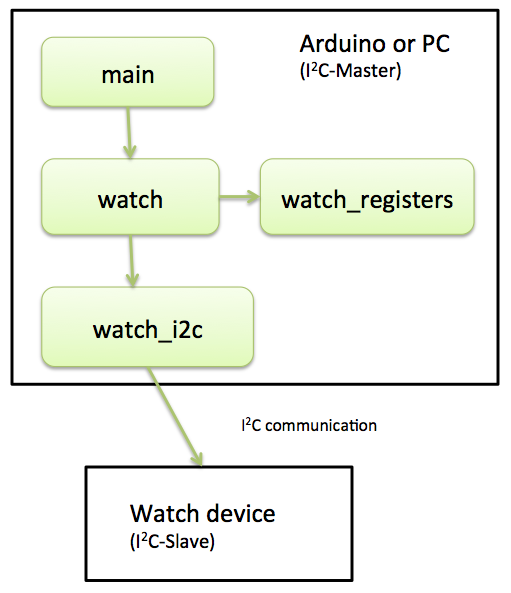
The device has one register for configuring the watch (CONF) and four other registers for storing the date and time. Both date and time are stored in a 16 bits using two byte registers.

The meaning of the registers in table 1 can be found in the appendix.

# Watch design overview

In figure 1 you can find a possible design for the watch. This design assumes that code is running on a device like an PC or Arduino an communicates via I2C with the Watch device.

In order to get the watch running you will have to implement the ‘watch\_register’ module. The other modules are already available.



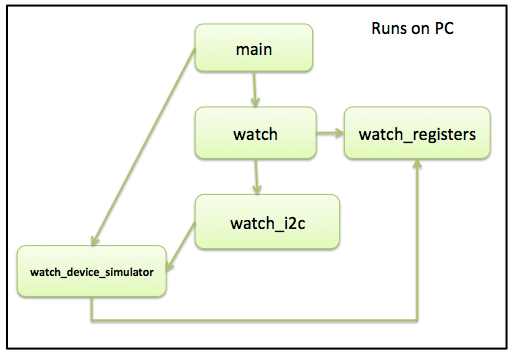
*Figure 1: Code modules of the watch design*

In table 2 you can find an overview of the modules.

|  |  |
| --- | --- |
| **Module** | **Summary** |
| main | Contains the main method. Show example code on how to use the watch |
| watch | Control the watch |
| watch\_registers | Manipulates the register bytes of the watch according the register specification (see appendix) |
| watch\_i2c | Sending and retrieving bytes to the Watch device via I2C. |

*Table 2: Modules of the watch design*

In order to decrease development time a simulator for the Watch device is used. The module design using the simulator is show in figure 3.

**

*Figure 3: Code modules of the watch design using a simulator*

|  |  |
| --- | --- |
| **Module** | **Summary** |
| watch\_device\_simulator | Contains a simulator for the I2C watch device. The simulator makes use of the module ‘watch\_registers’ for manipulating the register values. |

*Table 3: Modules of the watch design using a watch simulator*

For making this assignment you can make use of the ‘Watch’ project code. This project can be compiled and executed on a PC. You can find the build instructions in table 4.

|  |  |
| --- | --- |
| **Build instructions:** | **Description:** |
| make | Builds the executable called ‘watch’. You can run it by executing ./watch in your terminal. |
| make test | Builds and executes the unit tests. |

Table 4: Build instructions

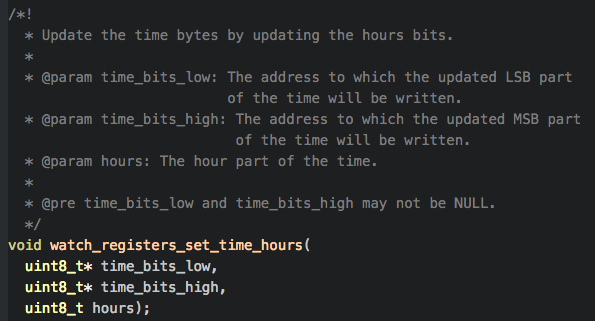
# Assignments

## Part A: implementing register manipulation

Implement the methods as found in ‘watch\_registers.h’ for manipulating the watch device registers. Also complete the unit tests in ‘watch\_registers\_test.c’.

**Tips**:

* Also make sure to read the method documentation in ‘watch\_registers.h’ before you start. When for instance setting the hours in the in the TIME\_BITS\_LOW and TIME\_BIT\_HIGH you should leave the minute en second bits unchanged. Also note the [precondition](https://en.wikipedia.org/wiki/Precondition) (@pre) that states that this method may NOT be called with NULL parameters.



* Make sure to use the register specification as can be found in the Appendix.

**Note: for implementing the ‘set’ methods it is not allowed to use the ‘get’ methods. This would decrease the learning effect of this exercise!**

## Part B (optional): migrating the Watch to Arduino platform (Master + Slave)

Note: this part is likely to more time consuming than part A.

Implement the Watch device on an Arduino. In order to do so you will have to:

* Create code for an Arduino I2C slave that will be the ‘Watch device’. You might reuse parts of the code.
* Create code for an Arduino I2C master. For this you can of course also reuse part of the already available code.

## Part C (optional): extending the simulator

The watch simulator currently only provides a bare minimum of functionality. It can for instance be extended to support the configuration settings and date functionality.

# Appendix: Watch device registers

## Configuration registers

### 0x20: Configuration register (Read/Write)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | **Bit** |
| - | - | - | - | **PS** | **TR[1]** | **TR[0]** | **TF** | **Function** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **Default** |

**TF**

0: Time format hh:mm

1: Time format hh:mm:ss

**TR[1:0]**

00: Time refresh disabled

01: Time refresh every 1 second

10: Time refresh every 30 seconds

11: Time refresh every minute

**PS[1:0]**

0: Time is updated

1: Time updated is paused

## Time registers

The time is stored in registers TIMEH and TIMEL:

### 0x21: Time value register MSB (Read/Write)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | **Bit** |
| **HR[3]** | **HR[2]** | **HR[1]** | **HR[0]** | **MIN[5]** | **MIN[4]** | **MIN[3]** | **MIN[2]** | **Function** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **Default** |

### 0x22: Time value register LSB (Read/Write)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | **Bit** |
| **MIN[1]** | **MIN[0]** | **SEC[5]** | **SEC[4]** | **SEC[3]** | **SEC[2]** | **SEC[1]** | **SEC[0]** | **Function** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **Default** |

The bits in the above registers store the current time:  
  
**SEC[5:0]**

The number of seconds (0..59)

**MIN[5:0]**

The number of minutes (0..59)

**HR[3:0]**

The number of hours (0..11).   
Note: the clock does not support 24-hour time format.

## Date registers

The time is stored in registers DATEH and DATEL:

### 0x23: Date value register MSB (Read/Write)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | **Bit** |
| **DM[4]** | **DM[3]** | **DM[2]** | **DM[1]** | **DM[0]** | **M[3]** | **M[2]** | **M[1]** | **Function** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **Default** |

### 0x24: Date value register LSB (Read/Write)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | **Bit** |
| **M[0]** | **YR [6]** | **YR [5]** | **YR [4]** | **YR [3]** | **YR [2]** | **YR [1]** | **YR[0]** | **Function** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **Default** |

The bits in the above registers store the current date:  
  
**YR[6:0]**

The number of years since the year 2000 (0..127)

**M[3:0]**

The month (1..12)

**DM[4:0]**

The day of month (1..31)