Software

# Projektverwaltung / Verwendete Programme

Für die Verwaltung des Projektes und der Projektumsetzung wurden folgende Programme verwendet:



GitHub / GitHub Desktop

GitHub wurde verwendet, um die Programmcodes, die Gehäusedaten, die PCB-Daten, die Stücklisten und die Datenblätter für alle Teammitglieder einfach zugänglich zu machen, ohne dass man sich jedes Mal die aktualisierten Dateien gegenseitig zuschicken muss.



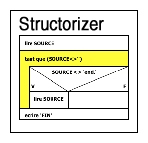
OneDrive

OneDrive wurde hauptsächlich für die Verwaltung der Dokumentation verwendet. OneDrive ermöglicht den gleichzeitigen Zugriff auf die Dokumentation.



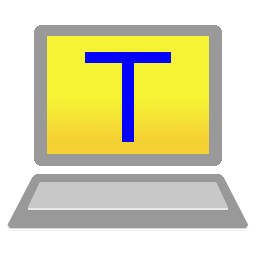
Microchip Studio

Microchip Studio ist die Programmierumgebung, die für dieses Projekt verwendet wurde. Mithilfe von Microchip Studio, ist es möglich gewesen, den ATMEL ICE zu verwenden, um auf den Attiny1606 die Programme zu hochzuladen.



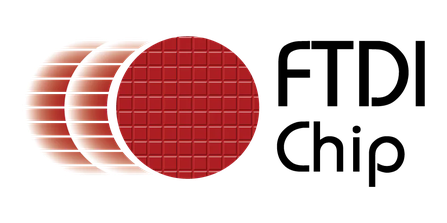
Structorizer

Der Structorizer wurde verwendet, um Struktogramme erstellen zu können. Struktogramme sind nichts anderes als eine vereinfachte Darstellung bzw. eine übersichtlichere Darstellung des Codes.



Teraterm

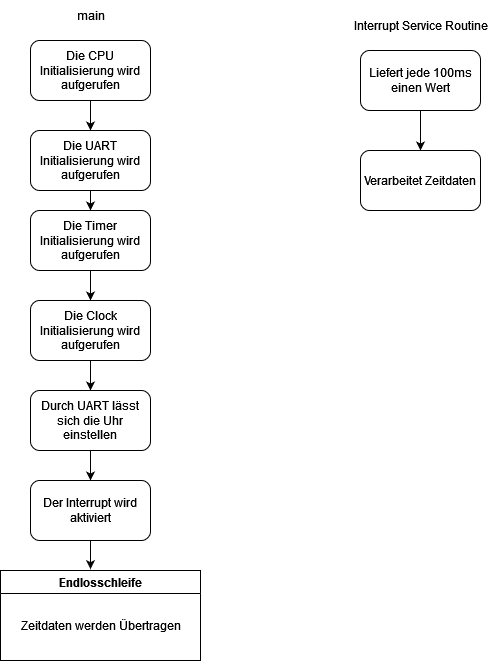
Teraterm wurde verwendet, um die Kommunikation zwischen der Lixieuhr und dem Rechner zu ermöglichen. Mithilfe von Teraterm, lässt sich die Uhrzeit einstellen.



FT\_Prog

Der FTDI Programmer wurde für den Versuch, einen Fehler zu beheben, verwendet.

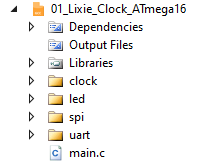
# Projektkonzept / Programmkonzept

Ein Flussdiagram, dass den Ablauf des Programmes übersichtlich veranschaulichen soll:

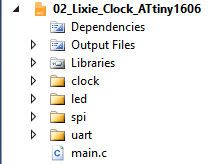
# Entwicklungsablauf / Codebeschreibung

Das Programm wurde erstmals für den Atmega16 geschrieben. Anschließend wurde Das Programm für den tinyAVR® 0-series umgeschrieben. Zuerst wurde der Attiny406 in Betracht gezogen, doch wegen der geringen Flash-Speicherkapazität und weilein UART-Interface programmiert werden sollte, ersetzte man den Attiny406 mit den Attiny1606 der statt 4 Kilobyte, 16 Kilobyte an Speicher bietet.

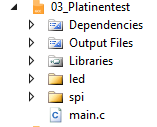
Insgesamt wurde mit 4 Teilprojekten gearbeitet:



Dieses Projekt wurde für den ATmega16 erstellt. Als die Lixie Segmente fertig wurden, wurde der Code mithilfe der MEGACARD V5.5 ausgetestet. Da dieses Projekt nach ein paar Fehlerbehebungen funktioniert hatte, wurde es die Grundlage für den Attiny1606.



Dieses Projekt ist das Hauptprojekt der Abschlussarbeit und wurde für den Attiny1606 verwendet. Von der Funktionsweise unterscheide sich dieses Projekt nicht zum ersten Projekt, jedoch musste der Code für den Atmega16 im ersten Projekt an den Attiny1606 angepasst werden, damit die Uhr vernünftig laufen kann.



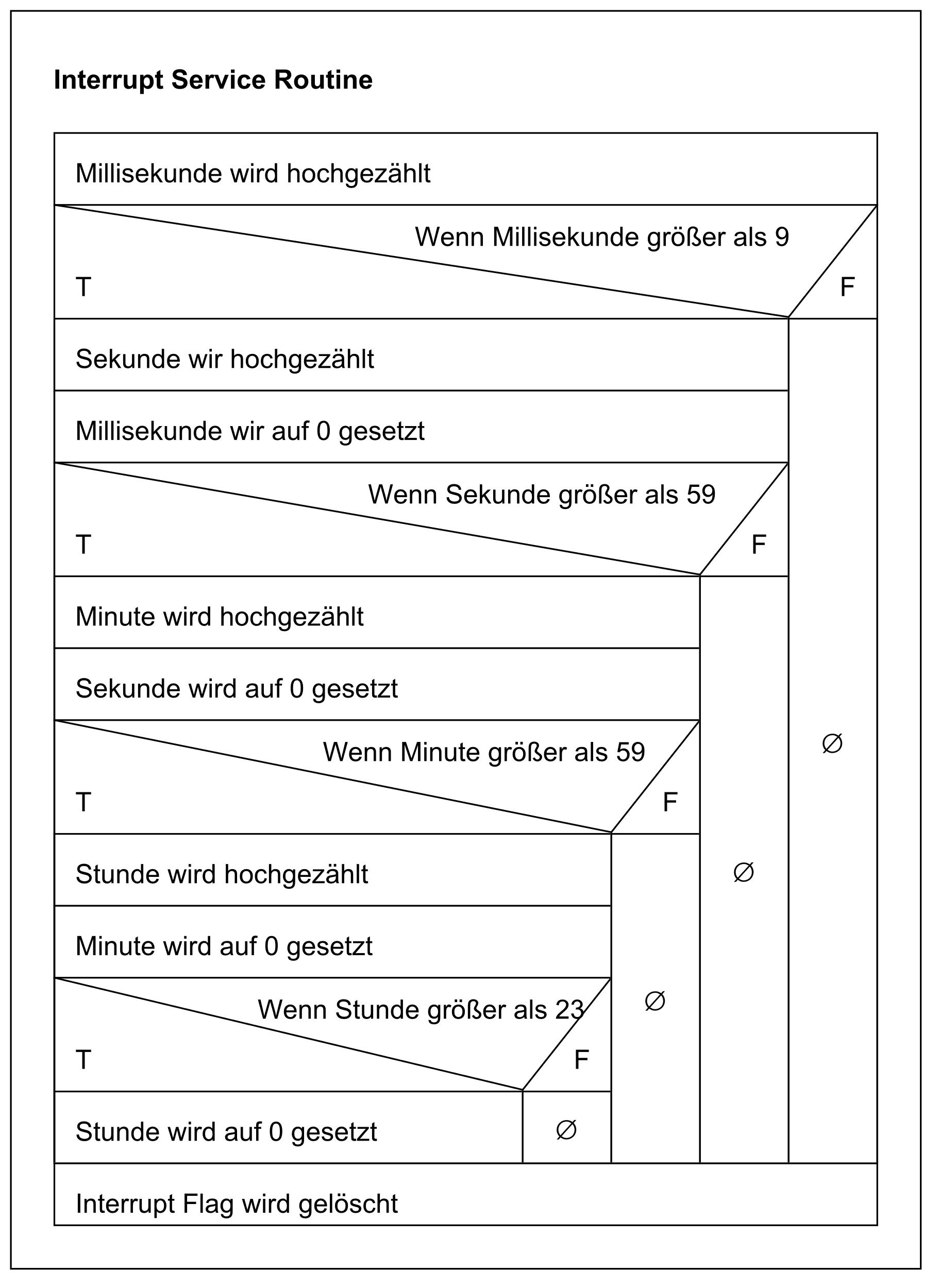
Dieses Projekt wurde für das erstmalige Testen der Segmente verwendet. Es ist hauptsächlich dafür gedacht, die LED’s auf jeder Platine zu Testen und zu Messen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dieses Projekt wurde für das Testen der UART-Funktion des Attiny1606 verwendet. Da es Schwierigkeiten mit der UART-Funktion gab, wurde das Demoprogramm für den Attiny1606 verwendet, um die UART-Funktion zu testen.

Interrupt Service Routine - Struktogramm



Interrupt Service Routine – Code

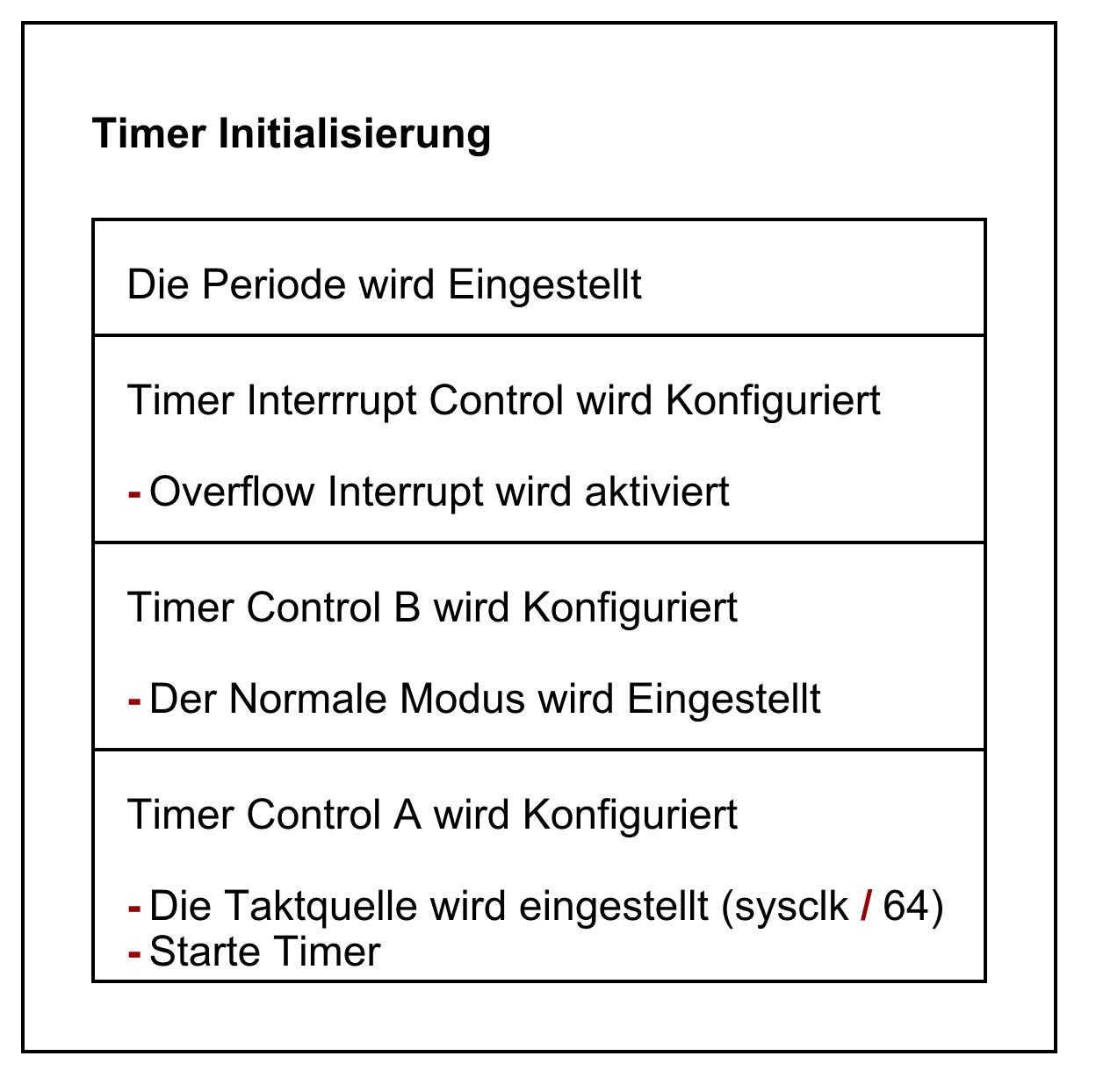


Interrupt Service Routine - Beschreibung  
Dieser Interrupt wurde so konfiguriert, dass die ISR jede 100ms anspringt. Um die Zeit zählen zu können, muss bei jedem Aufruf die Variable „miliseconds“ hochgezählt werden. Wenn die Variable den wert 9 übersteigt, wird er zurückgesetzt und die Variable „seconds.time“ wird um 1 hochgezählt. Sollte „seconds.time“ den Wert 59 übersteigen, wird die Variable auf 0 gesetzt und die Variable „hours.time“ wir um 1 hochgezählt. Wenn zu guter Letzt „hours.time“ den Wert 23 übersteigt, wird „hours.time“ auf 0 gesetzt.

Mit diesem Prinzip kann jede Uhr verarbeitet werden, egal ob sie mit LED’s angesteuert wird oder über eine 7 Segment Anzeige läuft.

Am Schluss muss das Interrupt Flag manuell gelöscht werden.

Timer Setup / Initialisierung – Struktogramm



Timer Setup / Initialisierung – Code

Timer Setup / Initialisierung - Beschreibung

Um die Interrupt Service Routine für die Uhr nutzen zu können, muss der Timer richtig eingestellt werden.

Um das Aufrufen der ISR richtig einstellen zu können, muss der OCR-Wert berechnet werden.  
Dies ist mit folgender Formel möglich:  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Zuerst wurde versucht, die ISR jede Sekunde aufzurufen.  
Weil das Ergebnis unter 2^8 befinden muss und keine Kommastellen haben sollte, war dies leider nicht möglich. Das sind die Ergebnisse:

Letzen Endes haben wir uns für jede 100ms entschieden, da dieser Wert sich unter 2^8 befindet und keine Kommastellen lieferte.  
Das Ergebnis lautet wie folgt:

Damit die ISR anspringt, wurde der Overflow Interrupt aktiviert.  
Der Timer wird im Normalen Modus betrieben.  
Am Schluss wird der Vorteiler auf 64 eingestellt und der Timer startet.

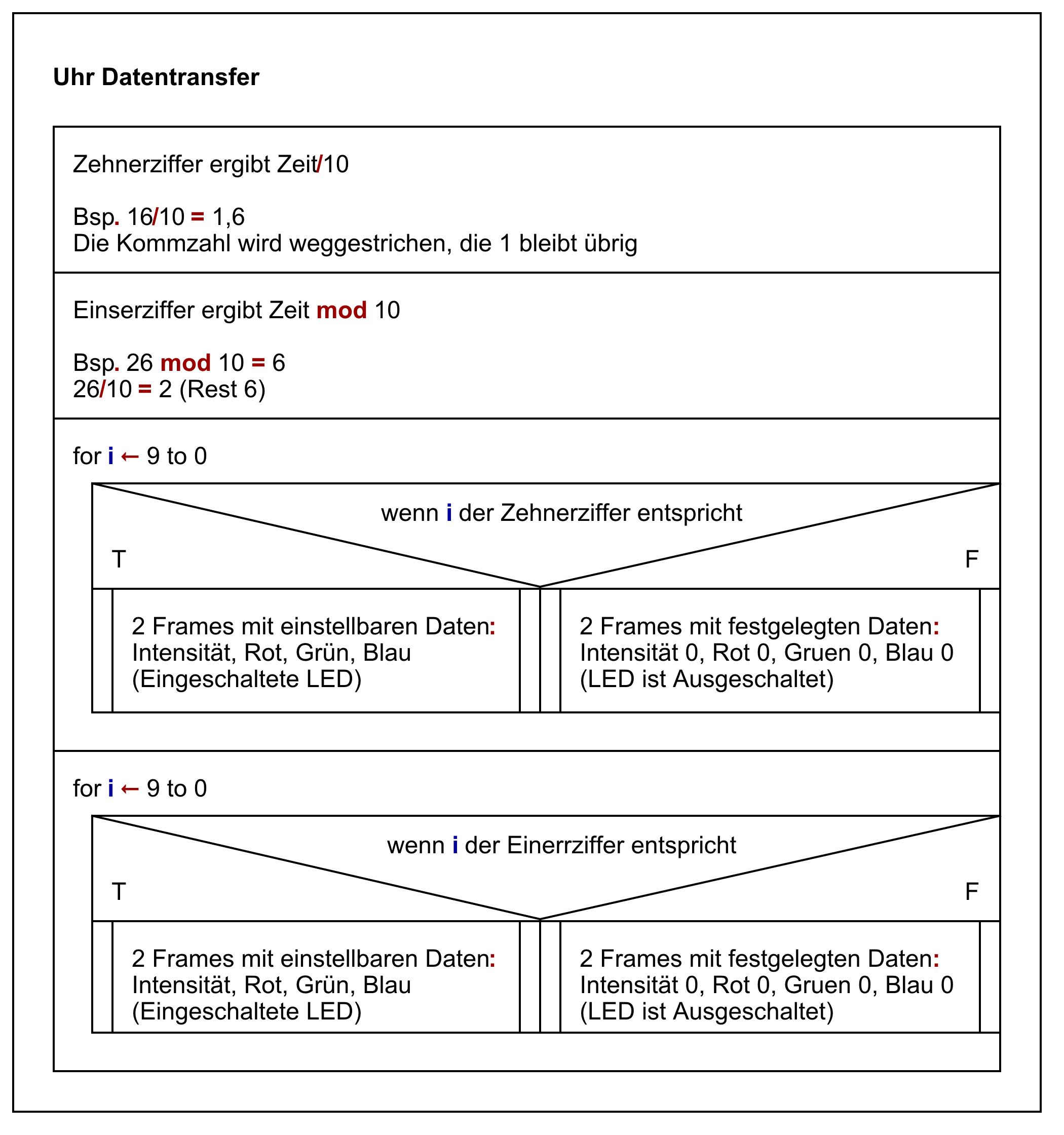
Zeitverarbeitung – Struktogramm

Zeitverarbeitung – Code



Zeitverarbeitung – Beschreibung

In diesem Unterprogramm findet Datentransfer der Uhr statt. Die Funktion übernimm die Sekunden, Minuten und Stunden und übergibt die der Funktion „clock\_transmit“, damit diese die Daten der Zeit verarbeiten kann. Vor und nach den Sekunden-, Minuten- und Stundentransfer befindet sich das Start- und Endframe, damit die LED’s wissen, dass sie angesprochen werden.

Datentransfer - Struktogramm

Datentransfer - Code



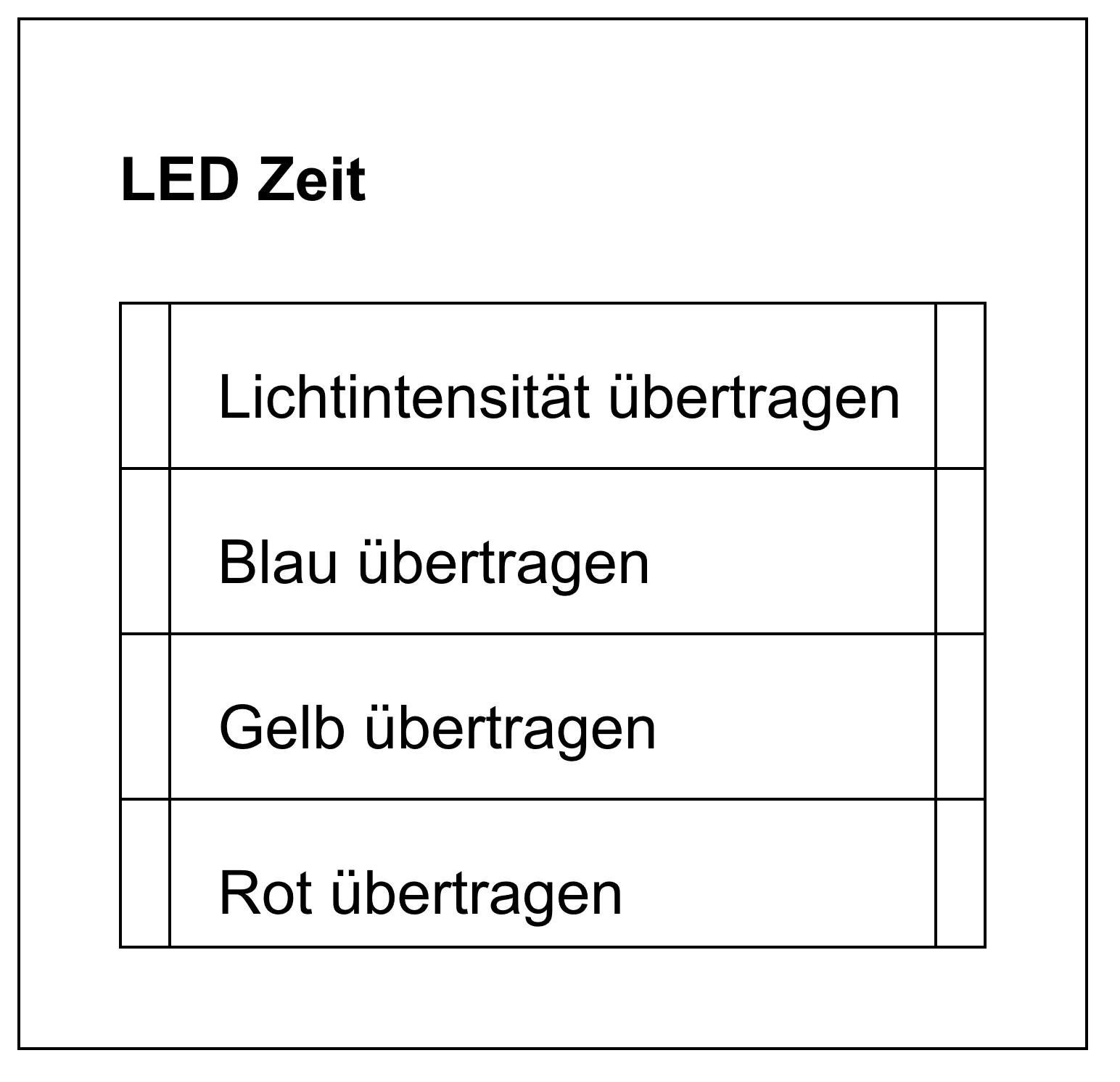
Datentransfer – Beschreibung

Dieser Teil des Codes ist für die Beleuchtung, bzw. das richtige Schalten der LED’s zuständig.

Um die Zehnerziffer beleuchten zu können, wird die Zeit durch 10 geteilt.   
Weil wir mit Intergers arbeiten, wird die Kommazahl weggestrichen und die Zehnerziffer bleibt übrig.

Für die Beleuchtung der Einserziffer wird die Zeit mit 10 moduliert. Das bedeutet, dass die Zahl durch 10 dividiert wird und die daraus resultierende Kommazahl das Ergebnis darstellt.

LED-Zeit - Struktogramm



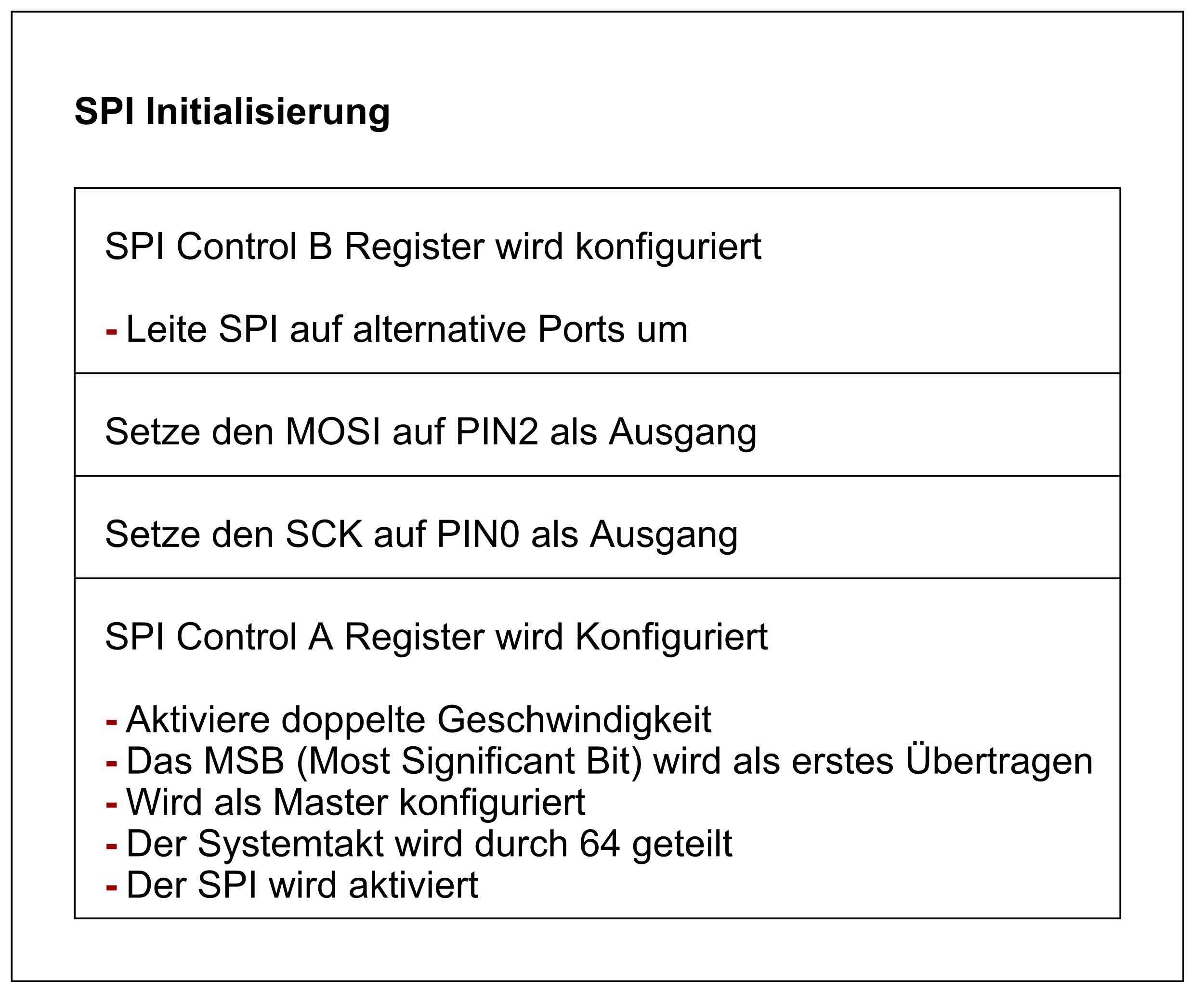
LED-Zeit – Code



LED-Zeit – Beschreibung

In dieser Funktion werden die Datenframes übertragen, die die LED’s brauchen, um zu wissen mit Welcher Intensität und mit welcher Farbe sie leuchten müssen.

SPI-Initialisierung – Struktogramm



SPI-Initialisierung – Code

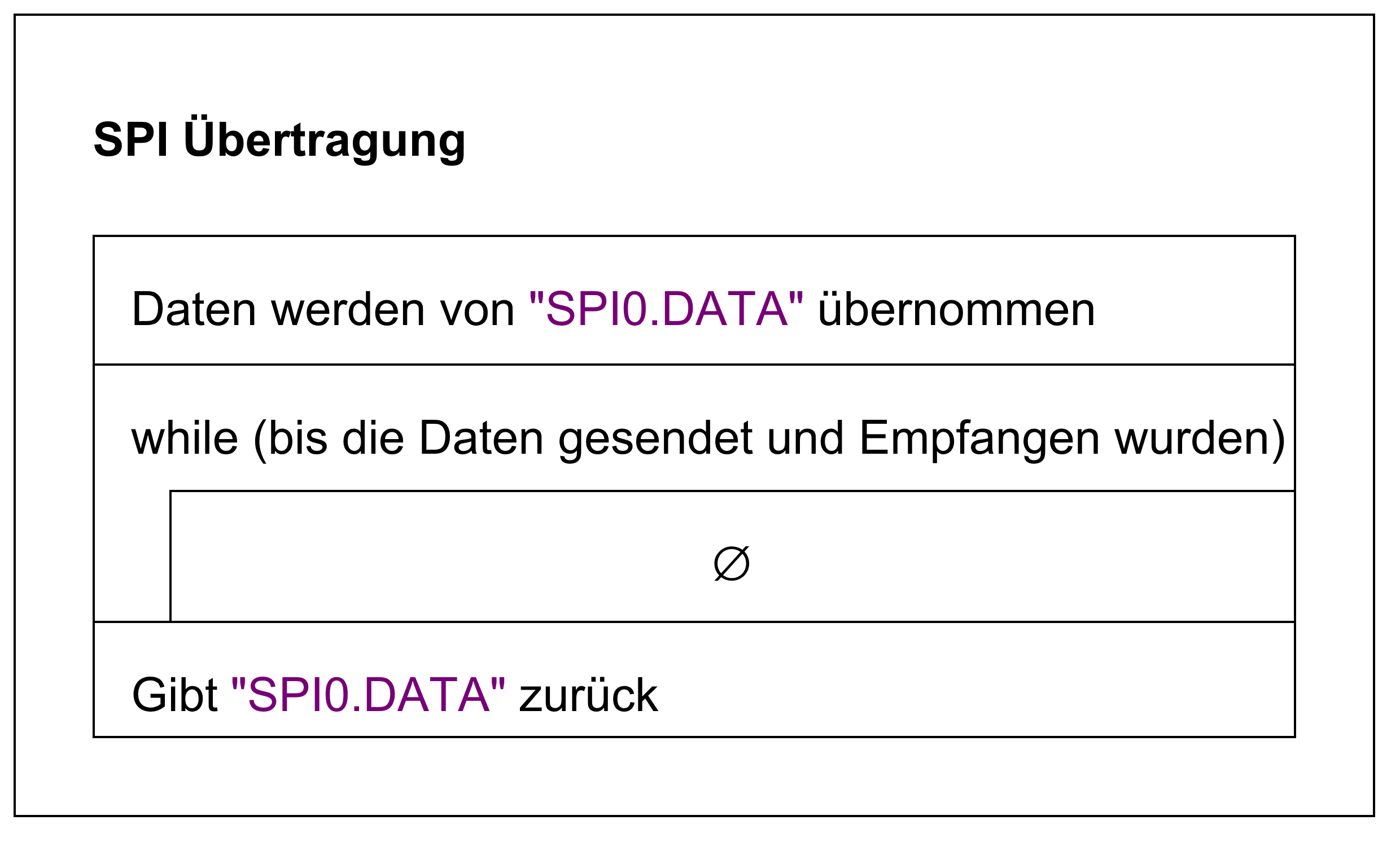


SPI-Initialisierung – Beschreibung

Da PC2 (PORTC / Pin 2) und PC0 (PORTC / Pin 0) alternative SPI-Pins sind, müssen diese Manuell eingestellt werden.

Für die Uhr werden nur MOSI (Master Output / Slave Input) und SCK (Taktleitung) verwendet.  
Der Geschwindigkeit des Taktes vom SPI wird verdoppelt, der SPI wird als Master und der Prescaler wird auf 64 eingestellt. Am Schluss wird das SPI-Modul aktiviert.

SPI-Übertragung – Struktogramm

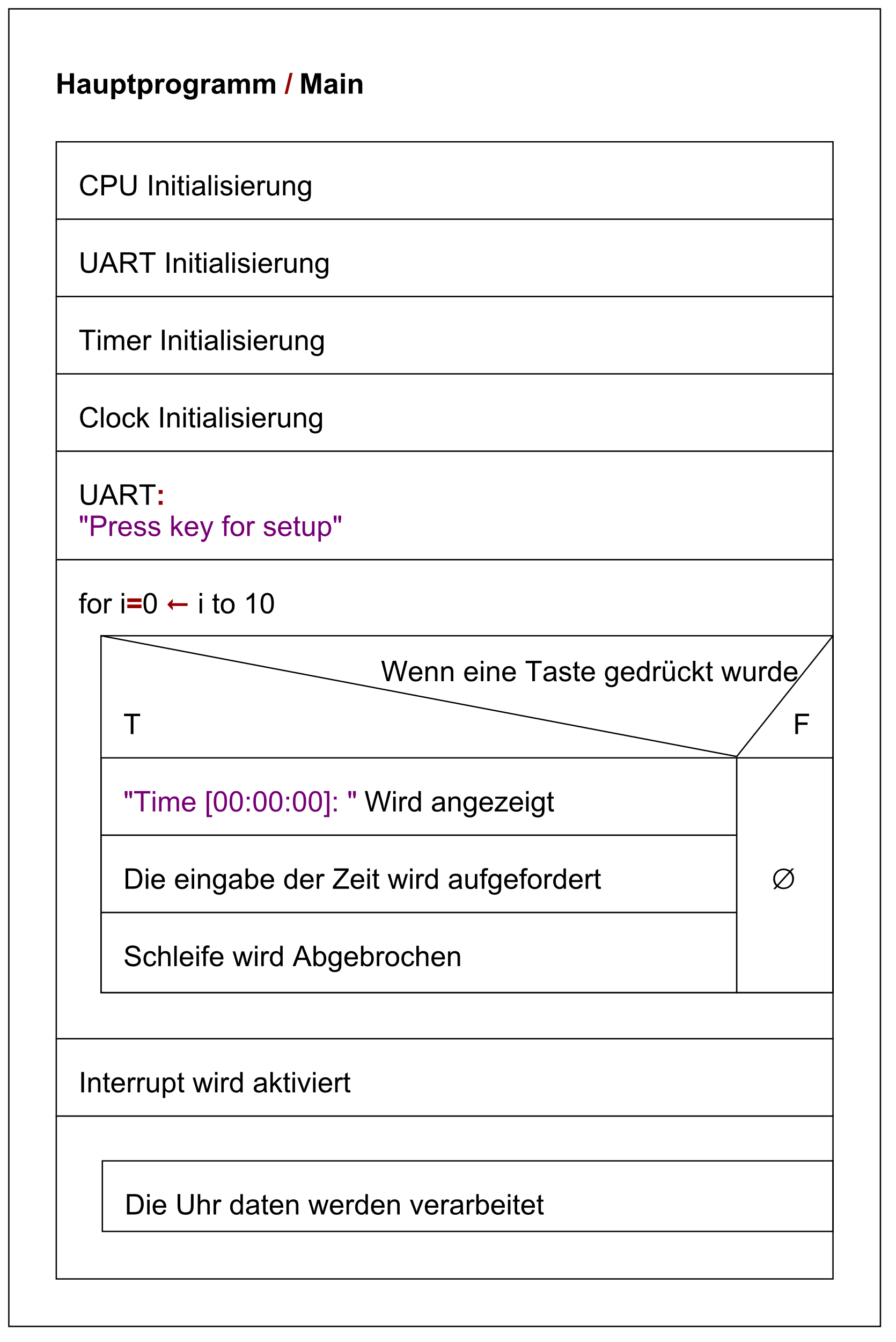


SPI-Übertragung – Code



SPI-Übertragung – Beschreibung

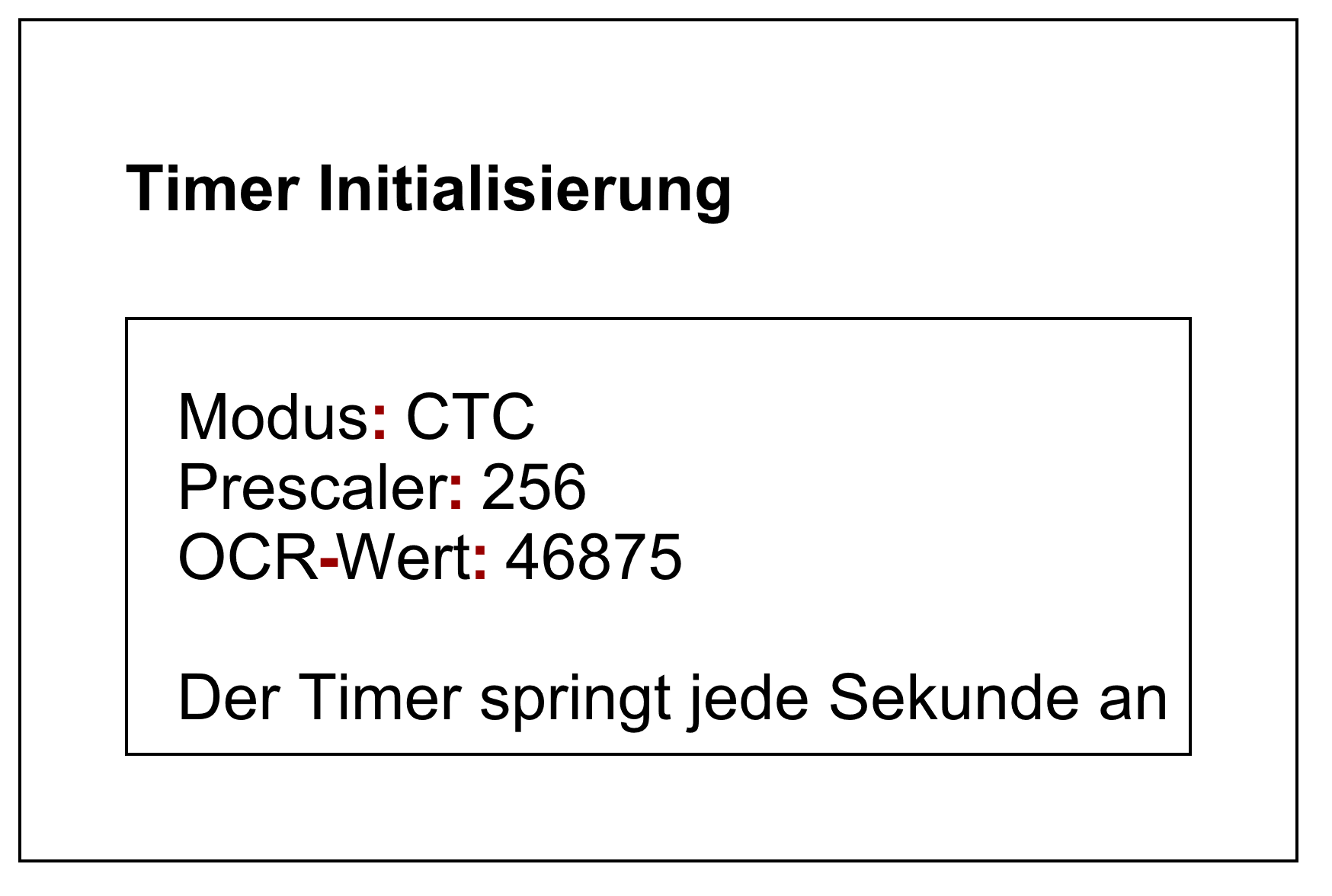
Der Code für die SPI-Übertragung wurde vom Datenblatt „TB3215-Getting-Started-with-SPI-90003215A.pdf“ übernommen und befindet sich auf Seite 17.

Das Hauptprogramm (main) – Struktogramm

Das Hauptprogramm (main) – Code

Das Hauptprogramm (main) – Beschreibung

Das Hauptprogramm ruft die Initialisierungen auf und verzögert das Starten der Uhr um 10 Sekunden, damit die Uhr über UART eingestellt werden kann, in dem man innerhalb dieser 10 Sekunden auf eine Taste drückt. Sollten diese 10 Sekunden vergangen sein oder die Uhrzeit eingestellt worden sein, werden die restlichen Initialisierungen aufgerufen und der Interrupt wird aktiviert. Anschließend läuft die Verarbeitung der Zeitdaten in der Endlosschleife.

Timer Setup / Initialisierung – Struktogramm (Atmega16)

Timer Setup / Initialisierung – Code (Atmega16)

Timer Setup / Initialisierung – Beschreibung (Atmega16)

Ein Bild, das Diagramm enthält.

Automatisch generierte BeschreibungUm den OCR-Wert des Timer 1 zu berechnen, wurde folgende Formel verwendet und umgeformt:

Der Timer ist im CTC-Modus und lässt die Interrrupt Service Routine durch das Ergebnis, dass sich unter 2^16 befindet und keine kommastelle beinhaltet, jede Sekunde anspringen.