

---

# Laborprotokoll

## SPI Schnittstelle

---

Systemtechnik Labor  
5BHIT 2017/18

Martin Wölfer

Note:  
Betreuer: WEIJ

Version 0.1  
Begonnen am 23. Januar 2018  
Beendet am 24. Januar 2018

# Inhaltsverzeichnis

<b>1</b>	<b>Aufgabenstellung</b>	<b>1</b>
<b>2</b>	<b>Aufbau</b>	<b>2</b>
2.1	Pinbelegung . . . . .	2
2.1.1	$Q_A - Q_H$ . . . . .	2
2.1.2	GND . . . . .	2
2.1.3	$V_{CC}$ . . . . .	2
2.1.4	SER . . . . .	3
2.1.5	OE . . . . .	3
2.1.6	RCLK . . . . .	3
2.1.7	SRCLK . . . . .	3
2.1.8	SRCLR . . . . .	3
2.2	Konkreter Aufbau . . . . .	3
<b>3</b>	<b>Umsetzung</b>	<b>6</b>
3.1	GPIOC konfigurieren . . . . .	6
3.2	GPIOA konfigurieren . . . . .	6
3.3	SPI konfigurieren . . . . .	6
3.4	main() . . . . .	7
3.5	Probleme . . . . .	7

# 1 Aufgabenstellung

Der Chip SN54HC595N von Texas Instruments realisiert ein Schieberegister. Eingang ist eine serielle SPI-Schnittstelle, der Ausgang besteht aus 8 unabhängigen Pins. Den Chip erhältst du bei der Übung (10 Stück stehen zur Verfügung).

Realisiere damit eine Ampel. Verwende für die Zeitintervalle Timer Interrupts.

Erstelle ein Protokoll. Darin soll zumindest ein Foto, der Code und entsprechende Erklärungen enthalten sein.

## 2 Aufbau

### 2.1 Pinbelegung

**SN54HC595 . . . J OR W PACKAGE**  
**SN74HC595 . . . D, DB, DW, N, OR NS PACKAGE**  
**(TOP VIEW)**

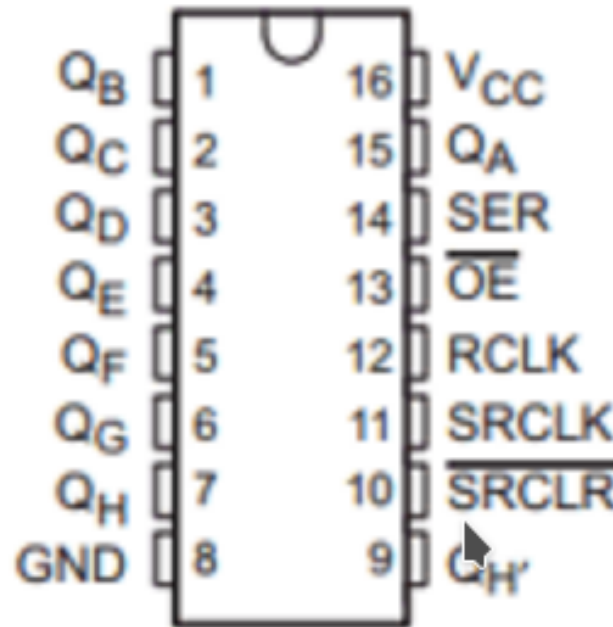


Abbildung 1: Pin-Belegung des SN54HC595N Chips

#### 2.1.1 $Q_A - Q_H$

Sind die jeweiligen Ausgänge, welche in dem Beispiel die LEDs mit Spannung versorgen. Wichtig dabei ist auch immer mit GND (sprich GROUND) zu verbinden:

#### 2.1.2 GND

Der GROUND Anschluss des Chips. Muss immer mit dem GROUND Ports des STM-Boards verbunden sein.

#### 2.1.3 $V_{CC}$

Muss mit dem 3Volt Anschluss vom STM-Board verbunden werden, damit der Chip genug, aber nicht zuviel Spannung bekommt.

#### 2.1.4 SER

Dieser Port wird benötigt für die sogenannte "MOSI-Verbindung" (**M**aster **O**ut - **S**lave **I**n) . Er wird dafür verwendet Board Daten an den Chip zu senden. Er muss mit dem Port PA11 Pin am Board verbunden werden.

#### 2.1.5 OE

Dieser Port muss ebenfalls mit dem GROUND Port des Boards angeschlossen werden, damit der Eingang an den Ausgang geleitet wird

#### 2.1.6 RCLK

Der serielle Takt, normalerweise mit GROUND verbunden, kann aber offen gelassen werden, wird durch den Strich gekennzeichnet.

#### 2.1.7 SRCLK

Die serielle Clock, normalerweise mit PC11 verbunden

#### 2.1.8 SRCLR

Wie schon erwähnt, wird durch den Strich gekennzeichnet dass dieser Port offen gelassen werden kann, wird aber in dem Beispiel mit  $V_{CC}$  verbunden.

### 2.2 Konkreter Aufbau

Zum Aufbau wurde eine externe Schaltplatte verwendet, um die Portzuweisung einfacher zu gestalten.

Bilder sind von Filip Scopulovic:

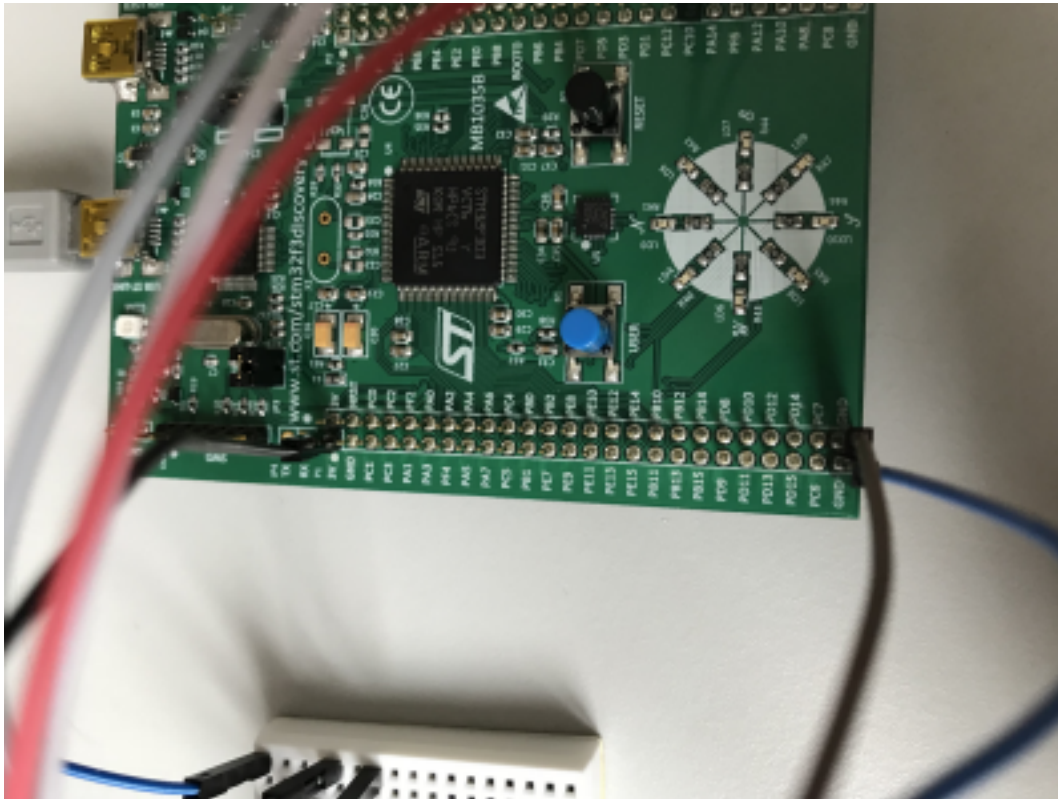


Abbildung 2: Verkabelung am Board 1

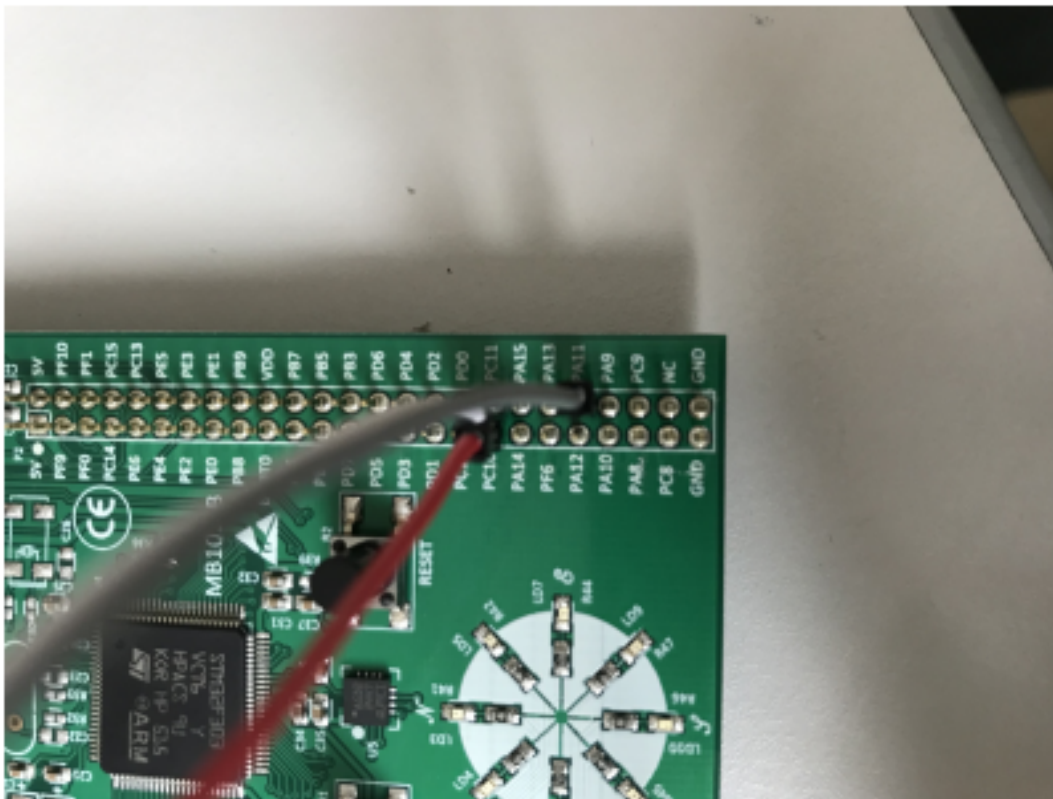


Abbildung 3: Verkabelung am Board 2

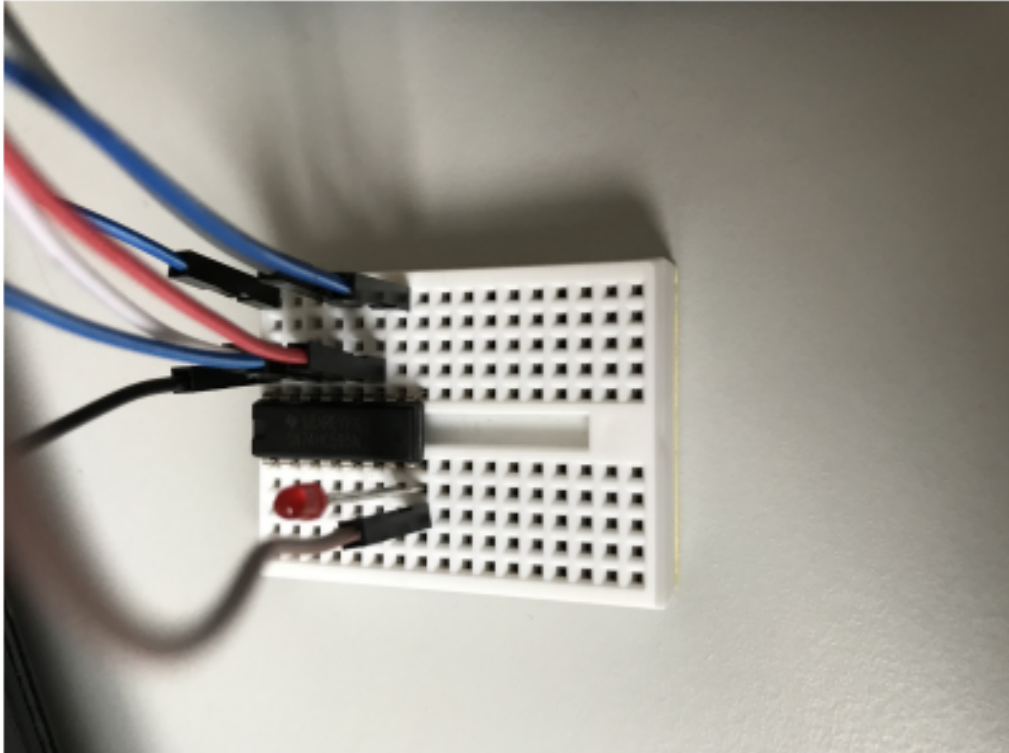


Abbildung 4: Verkabelung auf der Schaltplatte inklusive der Portanschliessung des Chips

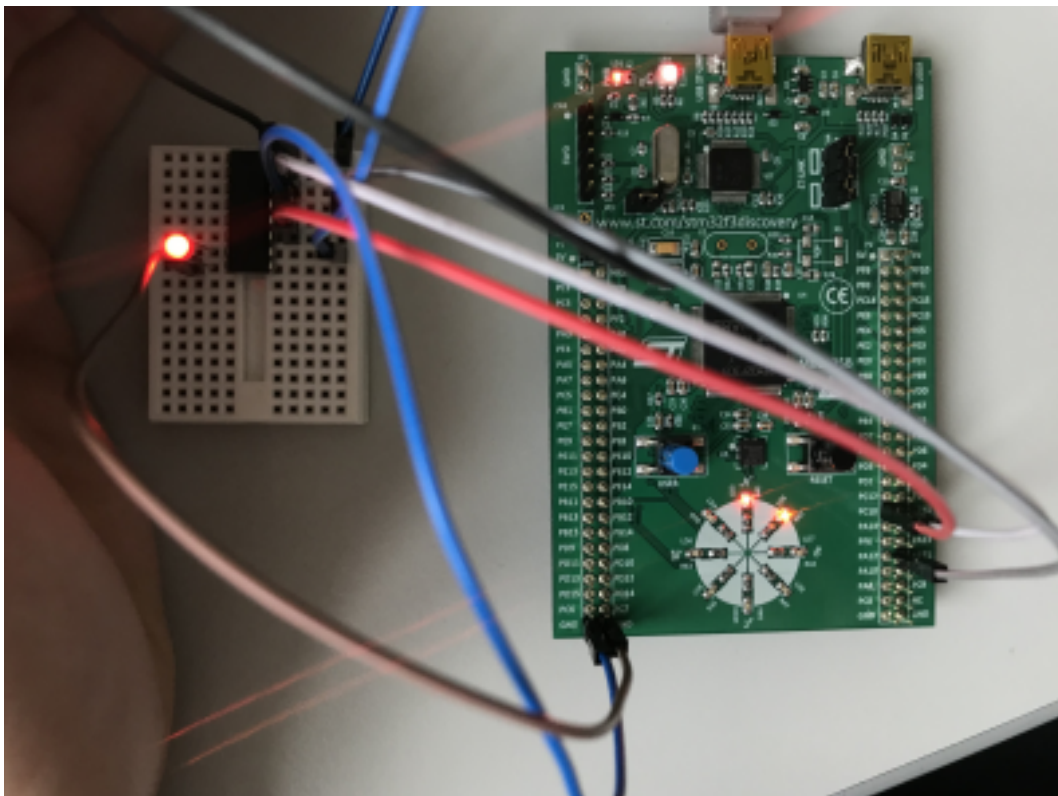


Abbildung 5: Gesamter Aufbau

## 3 Umsetzung

### 3.1 GPIOC konfigurieren

Zuerst müssen der Pins des Ports C konfiguriert werden:

```
1 void configureGPIOC () {  
    GPIOC_CLK_ENABLE();  
3    // Pins 10, 11 und 12 konfigurieren  
    GPIO_InitStruct.Pin = GPIO_PIN_10| GPIO_PIN_11| GPIO_PIN_12;  
5    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;  
    GPIO_InitStruct.Pull = GPIO_PULLDOWN;  
7    GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;  
    GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;  
9    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);  
}
```

### 3.2 GPIOA konfigurieren

Natürlich müssen auch die Pins des Ports A initialisiert werden:

```
void configureGPIOA () {  
2    GPIOA_CLK_ENABLE();  
    GPIO_InitStruct.Pin = GPIO_PIN_11;  
4    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;  
    GPIO_InitStruct.Pull = GPIO_PULLUP;  
6    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;  
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);  
8 }
```

### 3.3 SPI konfigurieren

Natürlich muss auch SPI initialisiert und konfiguriert werden:

```
void configureSPI () {  
2    SPI3_CLK_ENABLE() ;  
    SpiHandle.Instance = SPI3;  
4    SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_256;  
    SpiHandle.Init.Direction = SPI_DIRECTION_2LINES;  
6    SpiHandle.Init.CLKPhase = SPI_PHASE_1EDGE;  
    SpiHandle.Init.CLKPolarity = SPI_POLARITY_LOW;  
8    SpiHandle.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLED;  
    SpiHandle.Init.CRCPolynomial = 7;  
10    SpiHandle.Init.DataSize = SPI_DATASIZE_8BIT;  
    SpiHandle.Init.FirstBit = SPI_FIRSTBIT_MSB;  
12    SpiHandle.Init.NSS = SPI_NSS_SOFT;  
    SpiHandle.Init.TIMode = SPI_TIMODE_DISABLED;  
14    SpiHandle.Init.NSSPMode = SPI_NSS_PULSE_DISABLED;  
    SpiHandle.Init.CRCLength = SPI_CRC_LENGTH_8BIT;  
16    SpiHandle.Init.Mode = SPI_MODE_MASTER;  
    rc = HAL_SPI_Init(&SpiHandle);  
18 }
```



### 3.4 main()

Es wurden nun alle Funktionen geschrieben um alle benötigten Ports und Schnittstellen zu initialisieren. Nun muss eine main Methode geschrieben, welche 1. Alle diese Funktion aufruft und 2. den eigentlichen Code ausführt, welcher die LED zum blinken bringt:

```
1  int main(void) {  
2      //zusätzlich zu den anderen Funktionen wird noch HAL_Init aufgerufen um alle Treiber zu starten fuer  
        den HAL driver  
        HAL_Init();  
4      configureGPIOC ();  
        configureGPIOA ();  
6      configureSPI ();  
        //Es wird eine int Variable erstellt  
8      uint8_t data = 0;  
        for(;;) {  
10         // Der ~ Operator invertiert alle Bits einer Variable, in dem konkreten Beispiel wird bei jedem  
            Durchlauf der while-true loop data auf 1 gesetzt wenn es davor 0 war, und auf 0 gesetzt wenn  
            es davor 0 war. Dadurch blinkt die LED alle 5 sekunden auf, weil bei SPI_Transmit ein Delay  
            von 5000ms angegeben wird, zusätzlich zu den anderen Parameter  
            data = ~data ;  
12         HAL_StatusTypeDef rc1 = HAL_SPI_Transmit(&SpiHandle , &data , 1 , 5000) ;  
        }  
14 }
```

### 3.5 Probleme

Das Problem ist nun, wenn das Projekt ausgeführt wird, blinkt die LED nicht. Daher keine Zeit mehr übrig bleib in der Schule (zu Hause besitze ich keine Kabel zum verbinden, deswegen nur die Fotos des Mitschülers), um die Konfiguration zu überprüfen, muss ich Ihnen das Protokoll leider nur so abgeben. Dadurch dass das Testen des Blinkens nicht funktioniert hat, hat es keinen Sinn gemacht sich an die Implementation der Ampel zu trauen.

## Literatur

- [1] A.S. Tanenbaum and M. Van Steen. *Verteilte Systeme: Prinzipien und Paradigmen*. Pearson Studium. Addison Wesley Verlag, 2007.

## Abbildungsverzeichnis

1	Pin-Belegung des SN54HC595N Chips . . . . .	2
2	Verkabelung am Board 1 . . . . .	4
3	Verkabelung am Board 2 . . . . .	4
4	Verkabelung auf der Schaltplatte inklusive der Portanschliessung des Chips . . . . .	5
5	Gesamter Aufbau . . . . .	5