# Projektdokumentation

## Aufgabenstellung

## 

Es ist eine Ampelsteuerung zu entwickeln, welche folgende Kriterien erfüllen soll. Hierbei handelt es sich um eine Ampelsteuerung, welche von Fahrzeugen und Fußgängern benutz werden kann.

Die Ampelphasen werden direkt vom Verkehrsgeschehen abhängig gesteuert.Wird ein Fahrzeug in der Nebenstraße erkannt, wird die Nebenstraße für 30 Sekunden grün geschaltet.

Fußgänger, welche die Hauptstraße überqueren wollen, können mithilfe eines Tasters die Hauptstraße auf Rot schalten. Folglich wird die Nebenstraße auf Grün geschalten.

**Kriterien:**

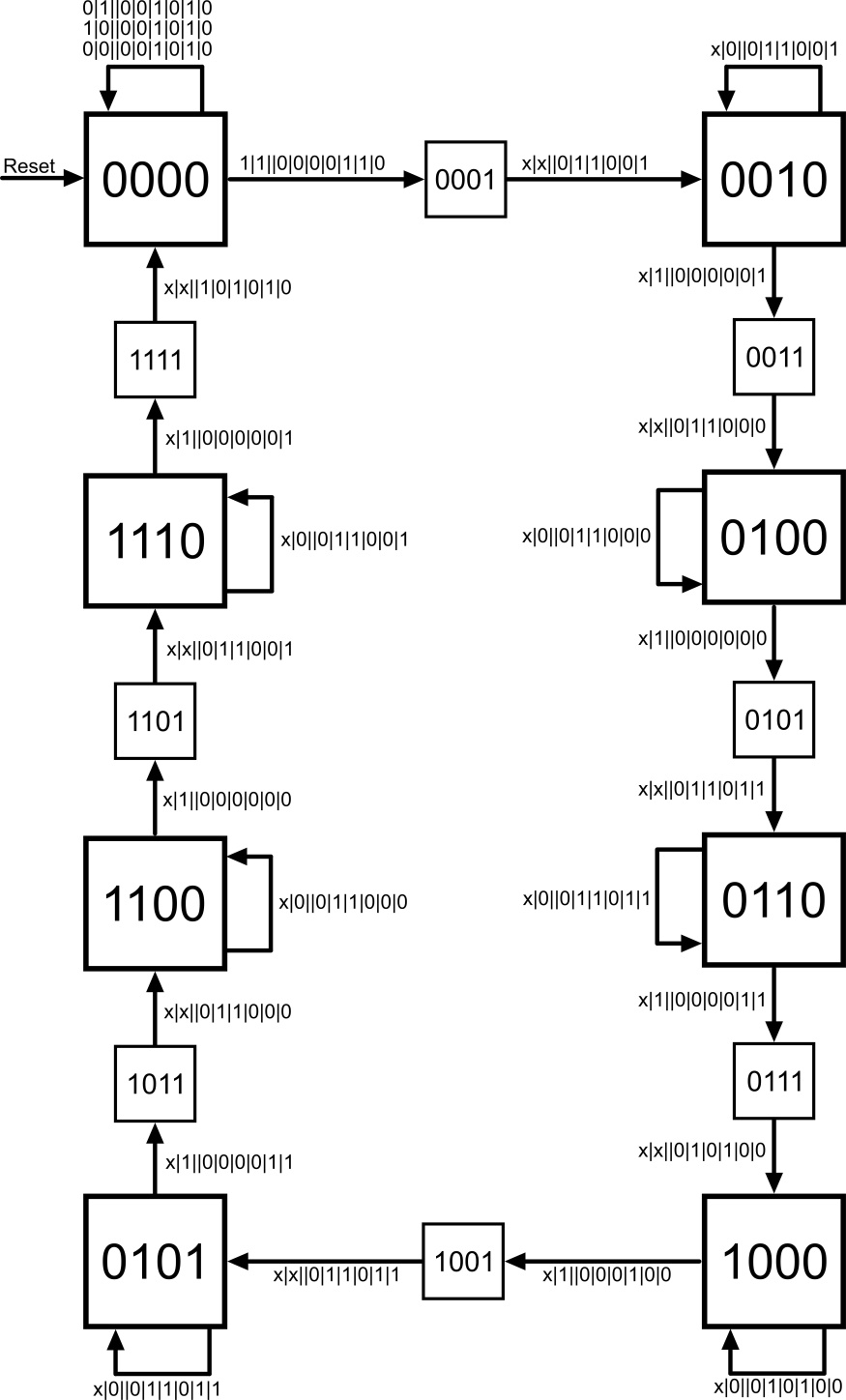
* Im „Default-Modus“ bzw. Normalbetrieb soll die Hauptstraße immer grün sein, die Nebenstraße rot. Wird ein Sensor bzw. ein Taster betätigt, wird die Hauptstraße mit entsprechenden Sicherheitsmaßnahmen, auf Rot geschalten und die Nebenstraße auf Grün.
* Die Hauptstraße muss eine Mindestgrünphase von 60 Sekunden haben.
* Die Nebenstraße hat nach Anforderung eine Grünphase von 30 Sekunden. Danach wird wieder in den Normalbetrieb umgeschaltet.
* Fahrzeuge dürfen nicht nach links oder rechts abbiegen, sondern müssen gerade über die Kreuzung fahren

**Zusatz:**

* Sicherheitsmaßnahme: Zwischen den einzelnen Grünphasen befindet sich für alle Ampeln eine Rotphase mit einer Dauer von 5 Sekunden
* Anforderungen, welche während der *60 Sekunden Mindestgrünphase* gestellt werden, werden gespeichert und erst nach den 60 Sekunden gültig.

## FSM (final state machine)

Name: Button | Timer\_END || Button\_RESET | t0 | t1 | a0 | a1 | a2



**Zustandsfolgetabelle:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Button | Timer\_END | z0 | z1 | z2 | z3 | z0+ | z1+ | z2+ | z3+ | Button\_  Reset+ | t0+ | t1+ | a0+ | a1+ | a2+ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| x | x | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| x | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| x | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| x | x | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| x | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| x | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| x | x | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| x | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| x | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| x | x | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| x | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| x | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| x | x | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| x | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| x | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| x | x | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| x | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| x | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| x | x | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| x | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| x | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| x | x | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

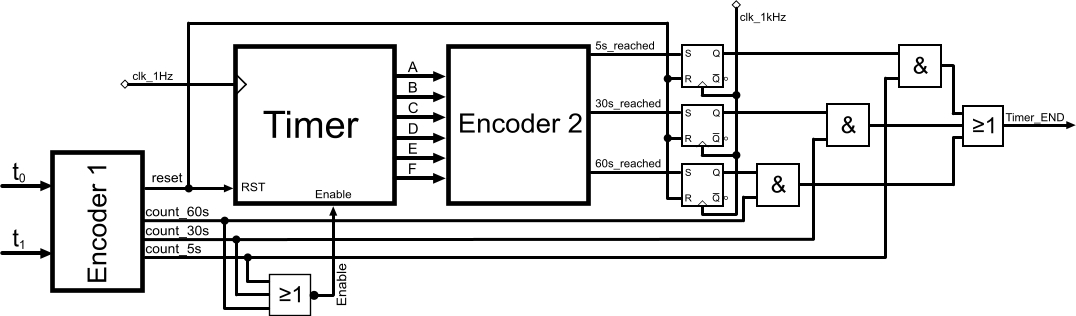
**Zustandsbeschreibung:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Z0 | Z1 | Z2 | Z3 | Beschreibung |
| 0 | 0 | 0 | 0 | "Default-Modus"; HS - grün, NS - rot |
| 0 | 0 | 0 | 1 | Zwischenzustand |
| 0 | 0 | 1 | 0 | HS – gelb, NS - rot |
| 0 | 0 | 1 | 1 | Zwischenzustand |
| 0 | 1 | 0 | 0 | HS - rot, NS - rot |
| 0 | 1 | 0 | 1 | Zwischenzustand |
| 0 | 1 | 1 | 0 | HS - rot, NS - gelb |
| 0 | 1 | 1 | 1 | Zwischenzustand |
| 1 | 0 | 0 | 0 | HS - rot, NS - grün |
| 1 | 0 | 0 | 1 | Zwischenzustand |
| 1 | 0 | 1 | 0 | HS - rot, NS - gelb |
| 1 | 0 | 1 | 1 | Zwischenzustand |
| 1 | 1 | 0 | 0 | HS - rot, NS - rot |
| 1 | 1 | 0 | 1 | Zwischenzustand |
| 1 | 1 | 1 | 0 | HS - gelb, NS rot |
| 1 | 1 | 1 | 1 | Zwischenzustand |

**Zwischenzustand:**

Die Zwischenzustände werden benötigt, um den Timer (Timing) zurückzusetzen und um eine neue Zeit zu fordern.

## Timing:



**Eingänge:**

* t0, t1  - Steuerleitungen für die Zeitdauer
* clk\_1Hz - Clock für *Timer*
* clk\_1kHz - Clock für *R-S FlipFlops*

**Ausgänge:**

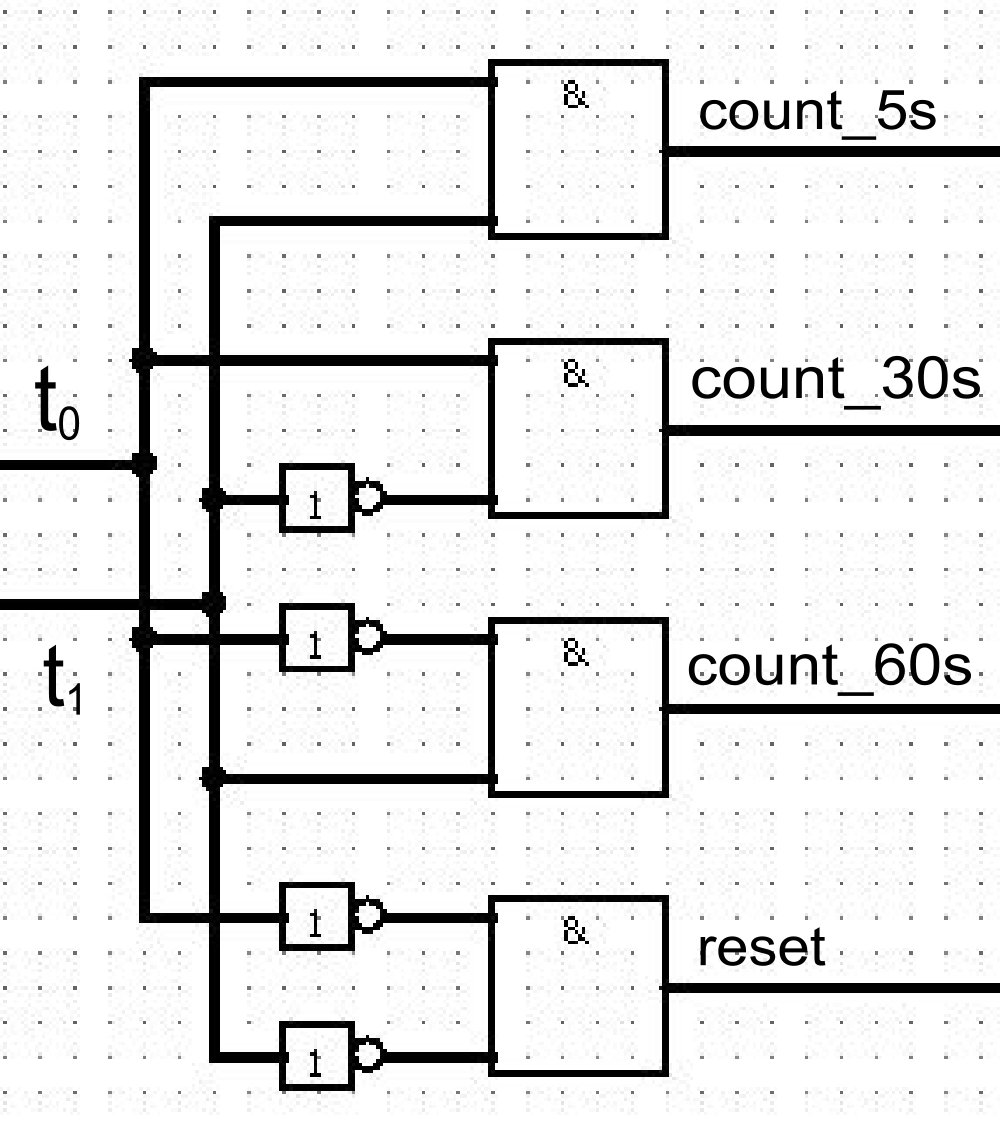
* Timer\_END - Signalisiert, dass geforderte Zeitdauer erreicht wurde

**Funktionsbeschreibung:**

Die Steuerleitungen t0 und t1 werden im *Encoder 1* encodiert. Bei Anforderung einer Zeit (*count\_60s, count\_30s* oder *count\_5s*) wird der *Timer* aktiviert (Enabled).  
  
Die Ausgänge des *Timer* werden im *Encoder 2* encodiert und mithilfe von *R-S FlipFlops* gespeichert.

Mithilfe der *UND-Gatter*, werden die gespeicherten Werte mit dem geforderten Wert verglichen und bei Übereinstimmung wird der Ausgang *Timer\_END* auf HIGH gesetzt.

## Encoder 1:



**Eingänge:**

* t0
* t1

**Ausgänge:**

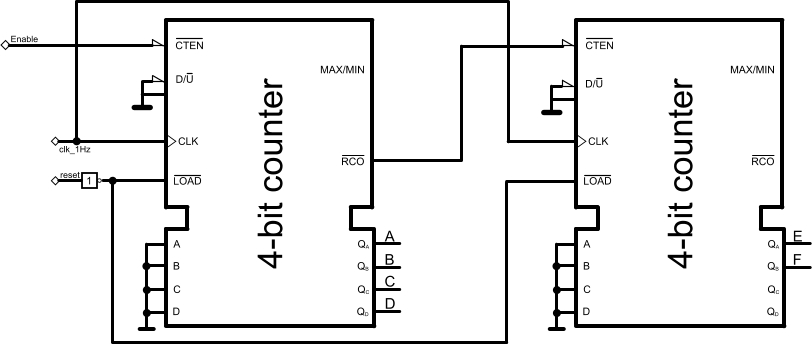
* count\_5s - 5 Sekunden werden gefordert
* count\_30s - 30 Sekunden werden gefordert
* count\_60s - 60 Sekunden werden gefordert
* reset - Timer wird zurückgesetzt

**Funktion:**

Die Steuereingänge t0 und t1 werden im *Encoder 1* encodiert.

|  |  |  |
| --- | --- | --- |
| t0 | t1 | Beschreibung |
| 0 | 0 | *Reset* wird auf HIGH geschalten |
| 0 | 1 | *count\_60s* wird HIGH geschalten |
| 1 | 0 | *count\_30s* wird HIGH geschalten |
| 1 | 1 | *count\_5s* wird HIGH geschalten |

## Timer:



**Eingänge:**

* Reset - Reset des Zählers
* Clk\_1Hz - Clock
* Enable - Aktivieren/ Deaktivieren des Zählers

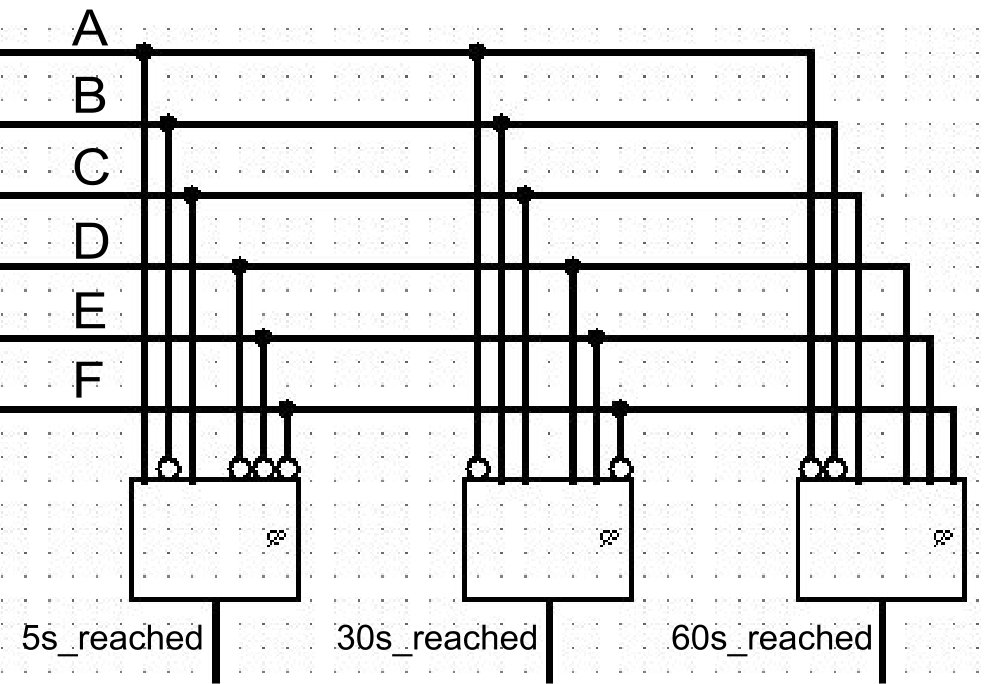
**Ausgänge:**

* A bis F - Binäre Ausgabe des Zählerstandes

**Funktionsbeschreibung:**

Als Zähler werden zwei synchrone *4-bit counter* verwendet, welche miteinander kaskadiert sind.   
Bei einem Reset, wird der Zähler auf ‚0‘ vorgeladen (zurückgesetzt).

## Encoder 2:



**Eingänge:**

* A bis F - aktueller Zählerstand

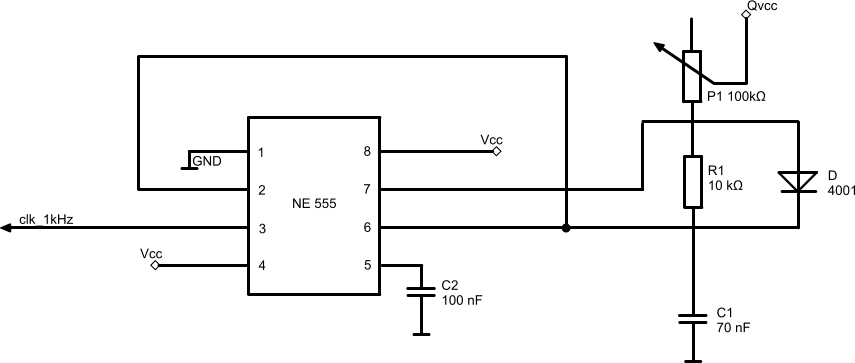
**Ausgänge:**

* 5s\_reached - gibt HIGH aus, wenn Zählerstand 5-Sekunden
* 30s\_reached - gibt HIGH aus, wenn Zählerstand 30-Sekunden
* 60s\_reached - gibt HIGH aus, wenn Zählerstand 60-Sekunden

**Funktionsbeschreibung:**

Der Zählerstatus (*5s\_reached*, *30s\_reached* oder *60s\_reached*) wird über einer Verknüpfung der Leitungen A bis F (aktueller Zählerstand) mithilfe von *UND- Gattern* ausgegeben.

## CLOCK:



**Ausgänge:**

* Clk\_1kHz - gibt eine Spannung Vcc mit einer Frequenz von 1kHz aus

**Funktion:**

Die *CLOCK* wird als Takt für die synchronen Bauteile verwendet.  
Weiteres wirkt die *CLOCK* indirekt auf die Zählgeschwindigkeit (definiert Dauer einer Sekunde)

**Dimensionierung:**

Das Potentiometer P1 wird so eingestellt, dass am Pin 3 eine Frequenz von 1 kHz ausgegeben wird.  
Laut eines NE555 – Rechners liegt der Wert bei

## Verwendete Technologien:

* Magix Grafik Designer
  + Erstellen der Grafiken
* Logism
* PsoC Creator

## Testaufbau: