Elektronische Morsetaste, CMOS-Ausführung

Claude L. Frantz (DJØOT)

20. Mai 2022

Abbildung 1: Die Taste in TTL Technologie, von vorn. Wie an dem einen Schalter zu erkennen ist, sind zwei Geschwindigkeitsbereiche vorgesehen.

1 Zielsetzung

Diese elektronische Morsetaste wird mit einem Paddle gesteuert. Sie besitzt die Squeeze-Funktionalität und erzeugt auch die Pause zwischen zwei Morsezeichen. Abgesehen von dem Oszillator, ist die Schaltung rein synchron aufgebaut, was für eine hohe Betriebssicherheit sorgt und was einen großen Geschwindigkeitsbereich ermöglicht. Im Ruhezustand ist der Stromverbrauch derart gering, dass ein Betriebsschalter überflüssig wird. Dank der Verwendung der CMOS-Reihe 4000, ist ein großer Betriebsspannungsbereich möglich.

2 Aufbau

Die Schaltung kann wie folgt unterteilt werden:

- Die Logikeinheit, die die Signale erzeugt.
- Der Taktgenerator der den Takt für die Logikeinheit erzeugt.
- Die Schnittstellenschaltungen, die das erzeugte Signal für die Ausgabe anpassen.

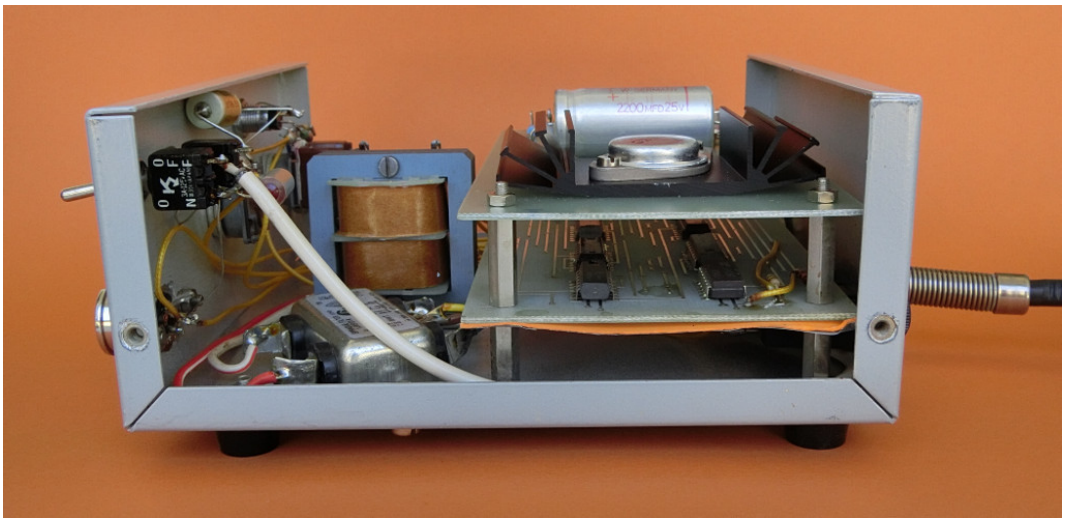

Abbildung 2: Die Innenansicht der Taste in TTL Technologie, von der Seite.

2.1 Der Taktgenerator

Seine Aufgabe ist die Erzeugung des Taktsignales für die Logikeinheit. Nachdem dieser Takt nur erzeugt wird wenn die Logikeinheit Signale erzeugt, besteht die Notwendigkeit diesen Takt so zu erzeugen, dass die Umschaltung vom Ruhebetrieb in den aktiven Betrieb in der Weise erfolgt, wie die Logikeinheit sie benötigt.

2.2 Die Logikeinheit

Sie arbeitet rein synchron, d.h. dass alle Kippstufen den gleichen Takt bekommen und somit ihren Zustand nur nach einem Takt ändern können. Allein die Kontakte vom Paddle können den Zustand, unabhängig vom Takt, verändern.

Diese sorgen dafür, dass die Logikeinheit in den aktiven Zustand übergeht bzw. dass sie in diesem Zustand bleibt. Nach dem Start in den aktiven Zustand, können die Kippstufen ihren Zustand nur noch bei einer Taktflanke verändern. Das bezieht sich damit auch auf das Ausgangssignal. Erst wenn die Logikeinheit in den Ruhezustand gegangen ist, kann sie durch die Kontakte vom Paddle wieder gestartet werden. Die Logikeinheit kann in folgende Teile unterteilt werden:

- Der 3-bit Zähler, in dem das Nutzsignal erzeugt wird. Hinzu gerechnet wird noch ein Flip-Flop, welches das Längenmerkmal des zuletzt erzeugten Signals (Punkt oder Strich) speichert.
- Die Eingangsschaltung an der der Paddle angeschlossen ist.
- Die kombinatorische Schaltung die die eben erwähnte Teile verbindet.

Die Eingangsschaltung speichert die Signale die vom Operator über den Paddle eingegeben wurden, bis sie vom Rest der Schaltung übernommen werden können. Sie sorgt auch dafür, dass der Oszillator, aus dem Ruhezustand, richtig gestartet wird.

Das eigentliche Nutzsignal, das letztendlich den Sender tasten wird, wird im 3-bit Zähler erzeugt. Es entspricht dem umgekehrten Signal der letzten Zählerstufe. Jedem Zählerstand entspricht ein Teil, mit der Länge von einem Punkt, vom dem zu erzeugenden Morsesignal. Der Takt sorgt dafür, dass der Zähler weitergeschaltet wird. Abhängig vom Wunsch des Operators, über den Paddle, sorgt die kombinatorische Schaltung dafür, dass der Zähler, zum geeigneten Zeitpunkt, auf einen bestimmten und geeigneten Wert zurückgesetzt wird, damit die gewünschte Signale erzeugt werden können. Jedem Zählerstand entspricht ein bestimmter Teil des zu erzeugenden Morsesignals. Abgesehen von dem Ruhezustand, verweilt der Zähler, in jedem Zustand, für eine Zeitdauer, die der Länge eines Punktes entspricht. Diese Zählerstände, in dezimaler Darstellung, sind wie folgt zugeordnet:

1: Der Beginn eines Striches.

2: Die Fortsetzung eines Striches.

3: Die weitere Fortsetzung eines Striches, oder auch ein ganzer Punkte.

4: Die Pause zwischen den Teile eines Morsezeichens, oder auch der Beginn einer längeren Pause, so z.B. zwischen zwei Morsezeichen.

5: Die Fortsetzung einer Pause zwischen zwei Morsezeichen.

6: Die weitere Fortsetzung einer Pause zwischen zwei Morsezeichen.

7: Der Ruhezustand, der auch einer noch längeren Pause entspricht.

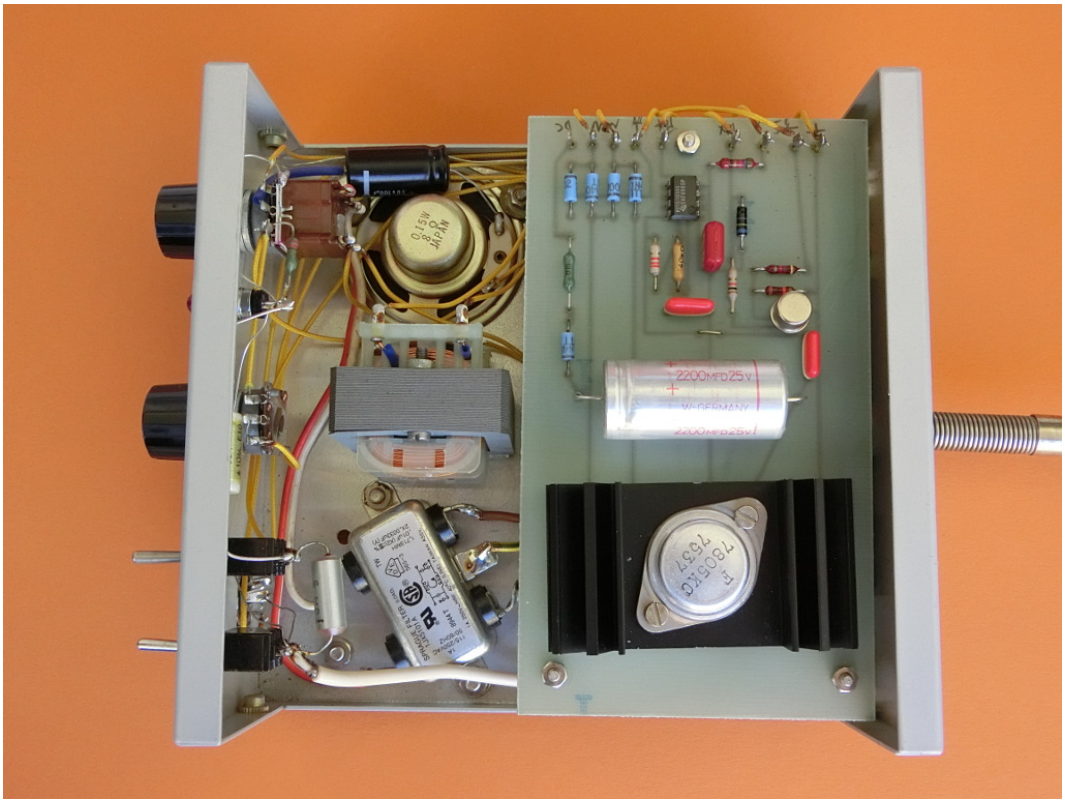

Abbildung 3: Die Innenansicht der Taste in TTL Technologie, von oben. Diese Platine beinhaltet das Netzteil mit Gleichrichter und Regler, sowie die Schnittstellenschaltungen.

2.3 Die Schnittstellenschaltungen

Sie sind folgende:

- Die Taststufe für negative Spannung.
- Die Taststufe für positive Spannung.
- Der Monitortonerzeuger.

3 Arbeitsweise

Die Arbeitsweise basiert auf einem 3-stelligem binären Aufwärtszähler, mit synchroner Lademöglichkeit, bei dem der invertierte Ausgang der dritten Stufe als Ausgang dient. Wenn der Zähler mit dem Wert 1 gestartet wird und wenn er bis

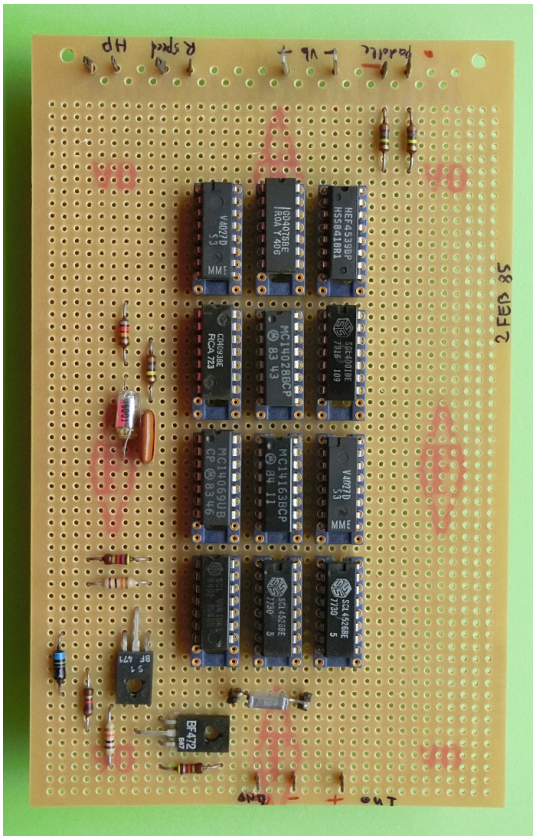

Abbildung 4: Prototyp der CMOS-Ausführung, von oben

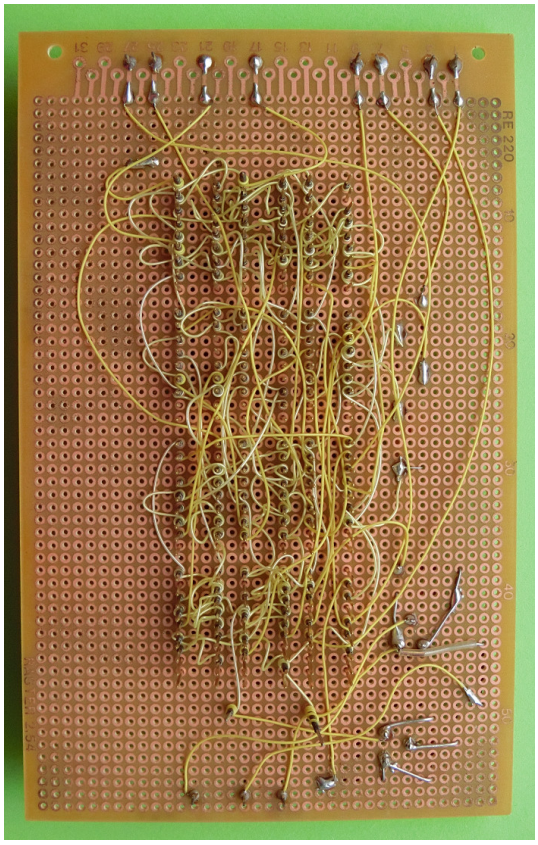

Abbildung 5: Prototyp der CMOS-Ausführung, von unten (Wire-Wrap®)

zum Wert 6 zählt, dann entspricht das Ausgangssignal dem Morsezeichen »O«, gefolgt von einer Pause von drei Punkte, was der Pause zwischen zwei Morsezeichen entspricht. Die Frequenz des Taktes entspricht der Länge von einem Punkt. Nehmen wir an, dass wir die Möglichkeit schaffen, zwischen den Taktsignale den Zähler so anzusteuern, dass er, beim nächsten Takt, entweder

- einfach weiterzählt,
- auf den Wert 1 schaltet,
- oder auf den Wert 3 schaltet.

Bei entsprechender Ansteuerung, haben wir dann die Möglichkeit beliebige Morsezeichen zu erzeugen, ggf. gefolgt von einer Pause von einer Länge von drei Punkte. Das ist die Grundlage der Arbeitsweise dieser elektronischen Taste.

Im Ruhezustand, der auch der langen Pause entspricht, befindet sich der Zähler in dem Zustand 7. Der Taktgenerator ist gesperrt. Sobald ein Kontakt am Paddle betätigt wird, wird das entsprechende Flip-Flop der Eingangsschaltung gesetzt. Dadurch geht das Signal »DoD« hoch, was wiederum die Sperrung des Taktoszillators aufhebt. Letzterer besteht aus einem einfachen NAND-Gatter mit Triggercharakteristik, gefolgt von einem 8-stufigen Abwärtszähler. Nachdem die Sperrung diesen Zähler auf Null gezogen hat, entspricht der nächste Zustand der Dezimalzahl 255. Somit wird sehr bald eine Taktflanke für den zentralen Takt erzeugt. Abhängig von der Länge des Morseelement das zu erzeugen ist (Punkt oder Strich), wird das Signal »LoadB« auf 1 (entspricht einem Punkt) oder auf 0 (entspricht einem Strich) gesetzt. Steht »LoadB« auf 1, wird bei dieser ersten Taktflanke, der Zähler auf 3 gesetzt. Anderenfalls wird der Zähler auf 1 gesetzt. Der nun angelaufene zentralen Takt schaltet den Zähler immer weiter. Für die Zustände 3, 4, 6 und 7 wird ein eigenes Signal erzeugt, was diese Zustande anzeigt (Signale S3, S4, S6 und S7). Sobald der Zustand 4 erreicht wird, erfolgt eine Pause des Ausgangsnutzsignals (umgekehrtes QC) für die Dauer von einem Punkt und es erfolgt eine Rücksetzung der beiden Flip-Flops der Eingangsschaltung. Zeigt das »DoD« Signal an, dass ein Kontakt am Paddle betätigt wurde oder immer noch betätigt wird, wird anschließend das nächste Element des Morsezeichens erzeugt, dadurch dass der Zähler wieder auf 1 oder 3 gesetzt wird, wie zuvor auch geschehen. Ist »DoD« nicht gesetzt, wird noch eine weitere Pause von 2 Punkte angehängt. Es wird zunächst angenommen, dass es eine Pause zwischen zwei Morsezeichen sein könnte. Steht der Zähler im Zustand 6 und »DoD« ist gesetzt, wird mit dem nächsten Morsezeichen begonnen. Der Zähler wird wieder auf 1 oder 3 gesetzt. Wird der Zustand 6 erreicht und »DoD« ist immer noch nicht gesetzt, wird angenommen, dass es sich um eine längere Pause handelt. Der Zähler wird dann in den Zustand 7 übergehen, der Taktgenerator wird gesperrt werden, die Schaltung geht in den Ruhezustand über.

Um die Zustände 1 bis 7 einnehmen zu können, genügt ein Zähler mit 3 binären Stellen. Hier wird jedoch ein Zähler mit 4 Stellen verwendet. Die vierte Stelle wird hier wie ein eigenes Flip-Flop verwendet. Diese Bauweise wird dadurch möglich, weil die Bedingungen zum Laden dieser Stelle, die gleiche sind wie für die Ladung des Zählerstandes. Ferner, werden nur die Zählerstände 1 bis 7 verwendet, so dass das Hochzählen des Zählers niemals einen Übertrag in die letzte Stellen verursachen wird.

Die kombinatorische Logik könnte in unterschiedlicher Weise realisiert werden. Das könnte mit Gattern und Invertern realisiert werden, wobei diverse Varianten möglich wären. Das gleicher Ergebnis könnte auch mit einem programmierbaren Speicher realisiert werden, z.B. als EPROM. Um die Anzahl der

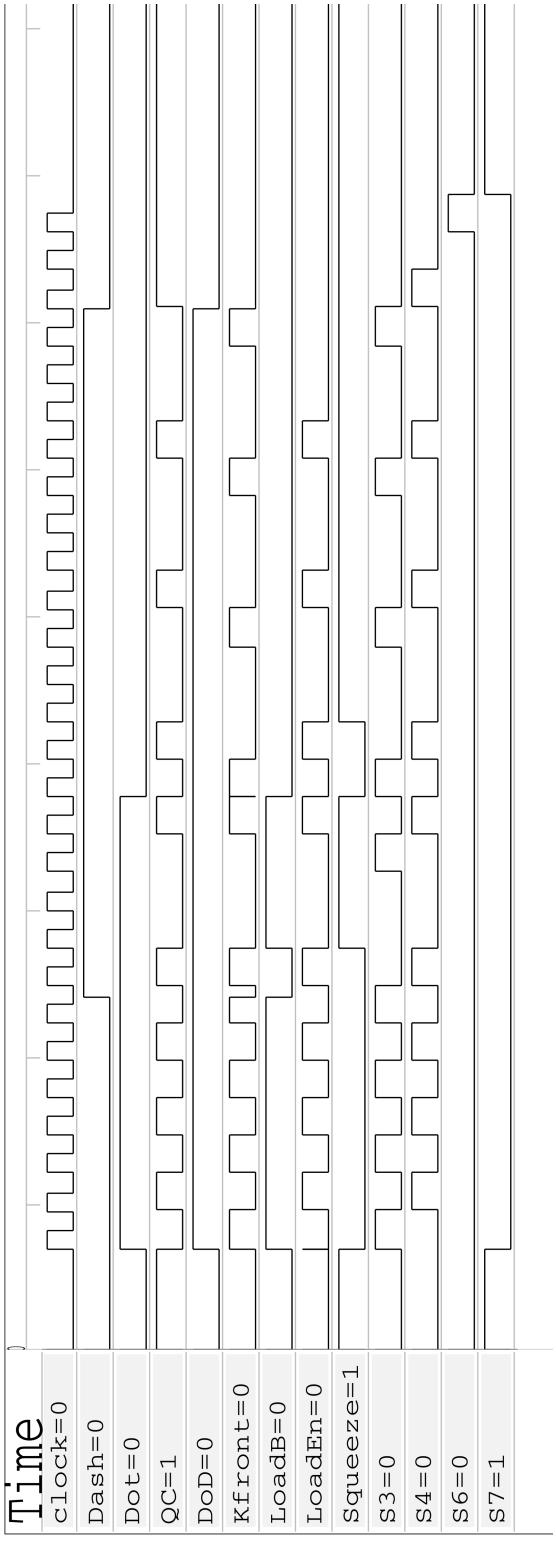

Abbildung 6: Ergebnis einer Simulation, bei der »Squeeze« eingesetzt wird.

IC-Gehäuse zu reduzieren, wurde eine Mischung von Gattern, Invertern und kombinatorische IC mit spezialisierter Funktionen verwendet (Multiplexer und Decoder).

3.1 Beschreibung der Signale

Es wird hier die Bedeutung der wichtigsten Signale der Logikeinheit beschrieben:

Dot: Die Aufforderung, einen Punkt zu erzeugen, ist in der Eingangsschaltung gespeichert.

Dash: Die Aufforderung, einen Strich zu erzeugen, ist in der Eingangsschaltung gespeichert.

DoD: In der Eingangsschaltung ist irgend eine Aufforderung gespeichert.

Kfront: Beim nächsten Takt, sollen die bisherigen Aufforderungen, die in der Eingangsschaltung gespeichert sind, gelöscht werden.

LoadEn: Beim nächsten Takt, soll der Zähler auf einen neuen Wert gesetzt werden statt weitergeschaltet zu werden.

LoadInh: Das Gegenteil von »LoadEn«.

LoadB: Das Bit mit dem Wert 2, das in den Zähler zu laden ist. Das niedrigwertigste Bit ist immer 1, das höchswertigste Bit ist immer 0. Somit können nur die Werte 1 oder 3 geladen werden.

Squeeze: Der Wert von »LoadB«, der beim letzten Laden anstand.

S3: Der Zähler steht im Zustand 3.

S4: Der Zähler steht im Zustand 4.

S6: Der Zähler steht im Zustand 6.

S7: Der Zähler steht im Zustand 7.

4 Hinweise für den Operator

Dem Operator stehen diverse elektronische Tasten mit Squeeze-Funktion zur Verfügung. Verständlicherweise arbeiten sie nicht alle mit der gleichen internen Logik. Wenn ein Operator bereits Erfahrung mit der einen Taste hat, kann es eine Umstellung erfordern, wenn er auf eine Taste mit einer anderen Logik umsteigt, besonders wenn sie über eine Logik für die Pause zwischen den Zeichen besitzt. So eine Umstellung kann mit dieser Taste selbstverständlich auch notwendig werden.

5 Historie

Ursprünglich wurde diese Taste in der TTL-Technologie entwickelt und sie hatte ein stabilisiertes Netzteil. Im Detail, war die Schaltung geringfügig anders. Ausgehend davon, wurde eine Schaltung in der CMOS 4000 Technologie entwickelt, die die Arbeitsweise voll übernommen hatte, jedoch mit Änderungen, die durch die Umstellung der Technologie erforderlich wurden. Diese Ausführung war für eine Speisung durch Trockenbatterien ausgelegt. Später wurde die Schaltung etwas vereinfacht, ohne dass sich an dem Ergebnis etwas geändert hat.

6 Schaltbilder

Die Verbindungen zur Stromversorgung erscheinen nicht auf den Schaltbildern. Das gilt auch für die Taktleitungen auf dem Schaltbild der Logikeinheit.

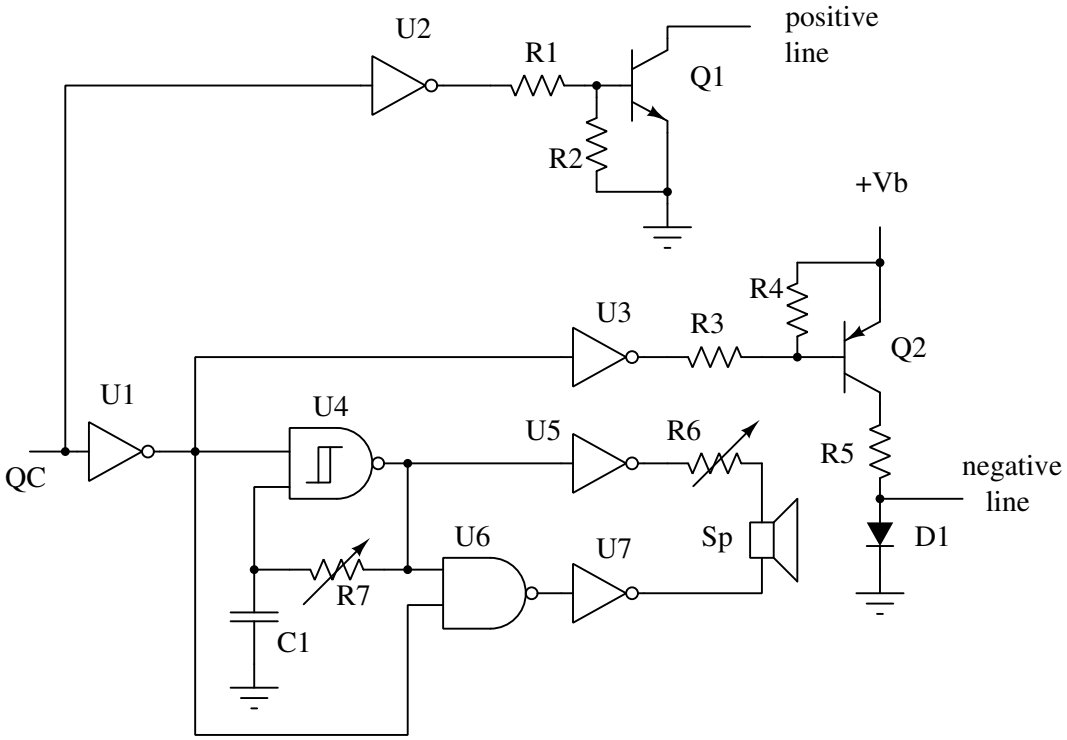

Abbildung 7: Das Schaltbild der Schnittstellenschaltungen

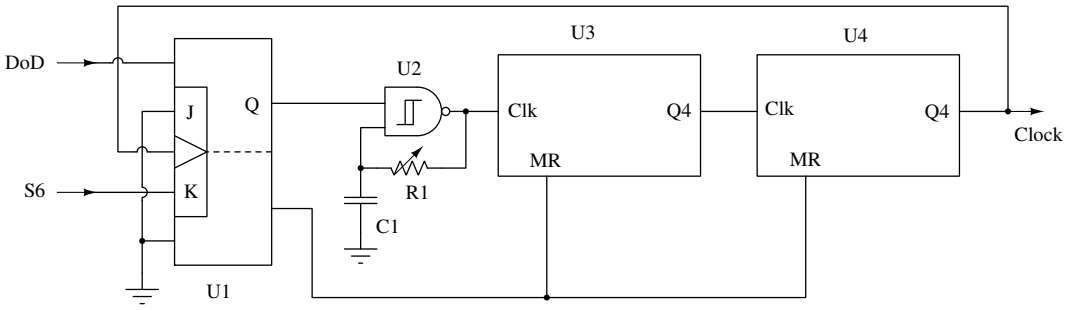

Abbildung 8: Das Schaltbild des Taktgenerators

Tabelle 1: Bauelemente der Schnittstellenschaltungen

Symbol	Typ oder Wert	Bemerkungen
U1	MC14010B	Einfacher Inverter
U2, U3, U5, U7	SCL4441UB	Invertierende Treiber
U4	MC14093B	NAND Gatter mit Hysterese, 2 Eingänge
U6	MC14011B	NAND Gatter, 2 Eingänge
Q1	BF471	NPN Transistor
Q2	BF472	PNP Transistor
D1	1N647	Silizium Diode
R1	4,7 kΩ	Schichtwiderstand
R2	10 kΩ	Schichtwiderstand
R3	4,7 kΩ	Schichtwiderstand
R4	10 kΩ	Schichtwiderstand
R5	1 kΩ	Schichtwiderstand
R6	100 kΩ	Logarithmisches Potentiometer, verkehrt montiert, als Lautstärkeregler
R7	150 kΩ	Schichtwiderstand, eventuell regelbar, als Tonhöheregler
C1	10 nF	Polyesther-Folienkondensator
Sp	600 Ω	Telefonhörer, hier als Lautsprecher verwendet

Tabelle 2: Bauelemente des Taktgenerators

Symbol	Typ oder Wert	Bemerkungen
U1	MC14027B	JK-Flip-Flop
U2	MC14093B	NAND Gatter mit Hysterese, 2 Eingänge
U3, U4	MC14526B	4-stufiger binärer Abwärtszähler
R1	500 kΩ	Logarithmisches Potentiometer mit einem festen 82 kΩ Schichtwiderstand in Serie, als Geschwindigkeitsregler
C1	1,2 nF	Folienkondensator, z.B. Polystyrol oder Polypropylen

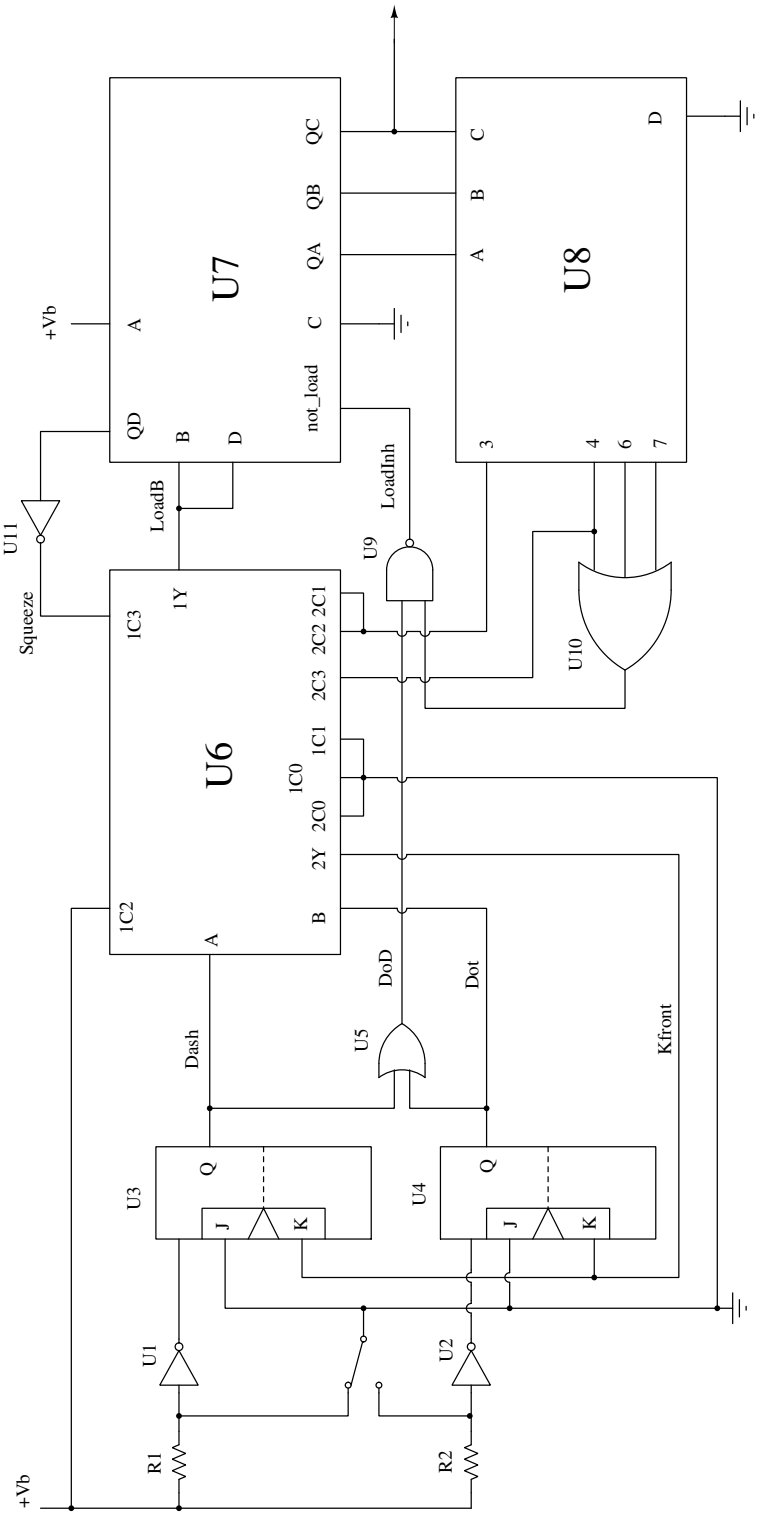

Abbildung 9: Das Schaltbild der Logikeinheit

Tabelle 3: Bauelemente der Logikeinheit

Symbol	Typ oder Wert	Bemerkungen
U1, U2, U11	MC14010B	Einfacher Inverter
U9	MC14011B	NAND Gatter, 2 Eingänge
U3, U4	MC14027B	JK-Flip-Flop
U5, U10	MC14075B	ODER-Gatter, 3 Eingänge
U6	MC14539B	Doppelter Selektor mit 4 Kanäle
U7	MC14161B	4-stufiger synchroner, binärer Aufwärtszähler mit synchronem Laden
U8	MC14028B	BCD Decoder
R1, R2	220 kΩ	Schichtwiderstand

Das CMOS IC MC14539B ist vom TTL IC SN74153 abgeleitet. Je nach Datenblatt, können die Signale jedoch anders bezeichnet werden.

7 Hardwarebeschreibung in verilog

Folgender Code wurde für die Simulation der Schaltung verwendet:

```
// ‘timescale 1ns / 1ps
module counter (clock, in, _load, out);
   // MC14161B
   input wire clock;
   input wire _load;
   input wire [3:0] in;
   output reg [3:0] out;
   initial begin
      out = 4’b0111 ;
      end
   always @ ( posedge clock)
     begin : COUNTER
        if (_load == 1’b0) begin
           out <= #2 in ;
        end
        else begin
           out <= #2 out + 4’b1 ;
        end
     end
endmodule // counter

// ‘timescale 1ns / 1ps
module flipflop (clock, J, K, set, clear, Q, _Q);
   // MC14027B
   input wire clock, J, K, set, clear;
   output reg Q, _Q ;
   initial begin
      Q = 1’b0 ;
      _Q = 1’b1 ;
   end
   always @ ( posedge clock or posedge set or posedge clear)
     begin : FF
        if (set === 1’b1 || set === 1’b1) begin
           Q <= #2 set ;
           _Q <= #2 clear ;
        end
        else begin
           if (J === 1’b1 && K === 1’b1 ) begin
              Q <= #1 ~Q ;
              _Q <= #1 ~_Q ;
```

```
            end
            else if (J === 1'b1) begin
               Q <= #1 1'b1 ;
               _Q <= #1 1'b0 ;
            end
            else if (K === 1'b1) begin
               Q <= #1 1'b0 ;
               _Q <= #1 1'b1 ;
            end
         end
      end
endmodule // flipflop

module clkgen (Enable, Stop, OutClock) ;
   input wire Enable, Stop ;
   output reg OutClock ;
   reg [7:0] Divider ;
   reg       running ;

   initial begin
      Divider = 8'b0 ;
      OutClock = 1'b0 ;
      running = 1'b0 ;
   end

   always begin
      #1 if ( Enable || running ) begin
         Divider = Divider - 8'b1 ;
         OutClock = Divider[7] ;
         running = 1'b1 ;
      end
      if (Divider == 8'b11111111 && Stop ) begin
         running = 1'b0 ;
         OutClock = 1'b0 ;
         Divider = 8'b0 ;
        end
   end // always begin
   endmodule // clkgen

// `timescale 1ms / 1us
module keyer () ;
//   input wire PaddleDot, PaddleDash, clock ;
//   output reg CWsigs ;
```

```
// Declare inputs as regs and outputs as wires
   reg PaddleDot, PaddleDash ;
   wire CWsigs ;
   wire QC, QB, QA;
   wire Dash, Dot;
   wire DoD, Kfront;
   wire S6, S3, S4, S7, LoadEn, LoadInh, LoadB;
   wire Squeeze;
   wire [3:0] Qcounter;
   wire ClkGo, clock ;

   initial begin
      $dumpfile("Keyer.vcd");
      $dumpvars;
      // $display ( "time\t clock, PaddleDot, Dot, PaddleDash, Dash, DoD,
      // Kfront, LoadEn, S3, S4, S6, S7, QA, QB, QC" ) ;
      // $monitor ( "%g\t %b  %b %b  %b %b  %b %b %b  %b %b %b %b  %b %b %b",
      // $time, clock, PaddleDot, Dot, PaddleDash, Dash, DoD, Kfront, LoadEn,
      // S3, S4, S6, S7, QA, QB, QC) ;
      PaddleDot = 0;
      PaddleDash = 0;
      #692 PaddleDot = 1 ;
      #1714 PaddleDash = 1;
      #1230 PaddleDot = 0 ;
      #2700 PaddleDash = 0 ;
      #5000 $finish ;
   end

   assign Squeeze = ~Qcounter[3];
   assign QC = Qcounter[2];
   assign QB = Qcounter[1];
   assign QA = Qcounter[0];
   assign S6 = ~QA & QC & QB;
   assign S3 = QA & ~QC & QB;
   assign S4 = ~QA & QC & ~QB;
   assign S7 = QA & QC & QB;
   assign DoD = Dash | Dot;
   assign LoadEn = DoD & (S6 | S4 | S7);
   assign LoadInh = ~LoadEn;
   assign LoadB = Dot & (~Dash | (Dash & Squeeze));
   assign Kfront = Dot ? (Dash ? S4 : S3) : (Dash & S3);

   flipflop DashFF (clock, 1'b0, Kfront, PaddleDash, 1'b0, Dash, );
```

```
   flipflop DotFF  (clock, 1'b0, Kfront, PaddleDot, 1'b0, Dot, );
   counter Generator (clock, {LoadB, 1'b0, LoadB, 1'b1}, LoadInh,
                      Qcounter); // [0] to [3]
   clkgen MainClock (DoD, S6, clock) ;
endmodule // keyer
```