



DL7BJ

Amateur Radio Station

BJ-Keyer

Dokumentation

ab Version 1.00 vom 15. Juli 2023

Tom, DL7BJ

Mail tom@dl7bj.de

Site <https://isnix.de>

Die Morsetelegrafie ist der
Urknall des Internets.

(DK5KE)

Vorwort

Was ist ein elektronischer Morsezeichengeber? Das ist ein Gerät, welches wir Funkamateure besser unter dem Namen Morse-Keyer kennen. Kurz gesagt, ein Morse-Keyer erzeugt elektronisch Punkte, Striche und Pausen. Während dies mit der Handtaste zum Morsen manuell gemacht werden muss, wird ein Morse-Keyer in Verbindung mit Ein- oder Zweihebeln verwendet und erzeugt bei Betätigung die Punkte und Striche sowie die Pausen selbständig.

Ist das neu?


Nein, Morse-Keyer gibt es schon sehr lange. Als Fertigeräte, als Bausätze und auch nur als Bauanleitungen in vielen verschiedenen Varianten. Etwas, das man quasi an jeder Straßenecke bekommt, in unterschiedlichen Preisklassen.

Warum noch ein Morse-Keyer?

Einige der erhältlichen Morse-Keyer sind in großen Gehäusen untergebracht, mit vielen Funktionen, Abschluß für eine Tastatur, dutzende Speicher und LC-Display und kosten viel Geld. Andere sind sehr günstig, haben aber nur einen Anschluß für eine Taste. Wenn man nicht gerade der Contester und DX-Jäger ist, gerne mal diverse Tasten an mehr als einem Transceiver verwendet und weder Steuerung über den PC noch Anschluß für Tastaturen benötigt, findet fast nichts am Markt.

Deswegen der BJ-Keyer, einfach, simpel, klein und trotzdem können mehrere Tasten und 2 Transceiver angeschlossen werden. Alles, was ich nicht benötige, habe ich auch weggelassen. Wer also auf der Suche nach einem Morse-Keyer mit ganz vielen Funktionen ist, dem empfehle ich eher sich woanders umzuschauen.

Wer aber einen kleinen Keyer mit wenigen aber praktischen Funktionen sucht, sollte hier weiterlesen.

Alle Unterlagen, wie Dokumentation, Schaltpläne und Software - kurz gesagt das gesamte Werk mit allem was dazu gehört, unterliegt der Attribution-NonCommercial-ShareAlike 4.0 International  Lizenz, wenn nicht anders angegeben.

Viel Spaß! Tom, DL7BJ

Inhalt

1 Funktionen	5
2 Hinweise zur Dokumentation	7
3 Grundlagen	9
3.1 Betriebsarten eines Morse-Keyers	9
3.1.1 Zeitverhalten	9
4 Bedienung	11
4.1 Tastaturbelegung	11
4.1.1 Übersicht	11
5 Schaltung	13
5.1 Beschreibung	13
6 Beschreibung der Hardware	15
7 Beschreibung der Software	17
7.1 Timer 1	17
7.1.1 Timer einstellen	17
7.2 Sinus Mithörton durch Pulsweitenmodulation	17
7.2.1 Grundlagen	17
7.2.2 Sinustabelle	21
8 Entwicklungsumgebung	23
Tabellen	25
Abbildungen	27

1 Funktionen

- BJ-Keyer Funktionsübersicht
 - Iambic A und Iambic B Mode
 - Anschluß für Handtaste und Paddle
 - Ausgang für Key-Eingang TRX
 - Ausgang für PTT
 - Stromversorgung 7-15V
 - Integrierter Lautsprecher für Mithörton
 - Mithörton als Sinussignal

2 Hinweise zur Dokumentation

In dieser Dokumentation werden diverse gleichbleibende Darstellungsweisen verwendet. Dies erleichtert Dir das Verständnis der Bedeutung. Texte, die auf dem Display erscheinen, werden in der Bedienungsanleitung so **dargestellt**. Quellcode wird in einer farbigen Code-Darstellung eingebunden.

3 Grundlagen

3.1 Betriebsarten eines Morse-Keysers

3.1.1 Zeitverhalten

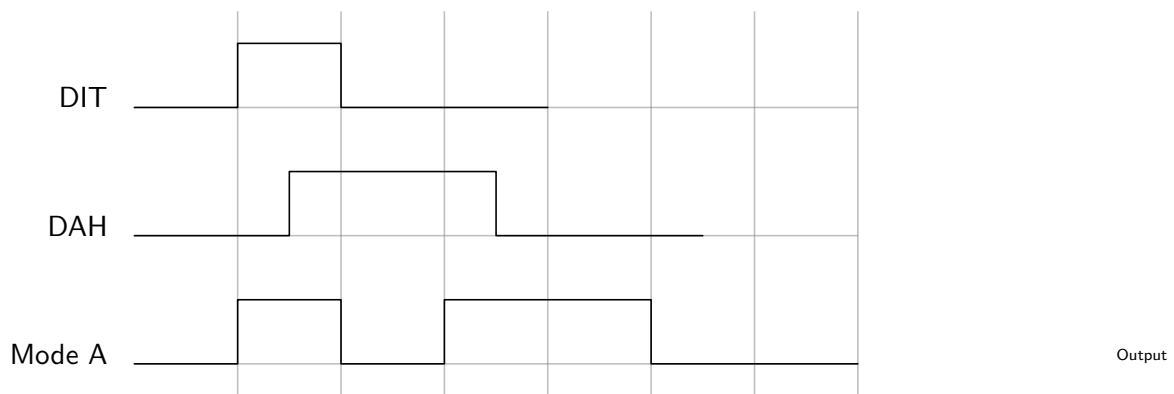


Abb. 3.1: Diagramm Mode A

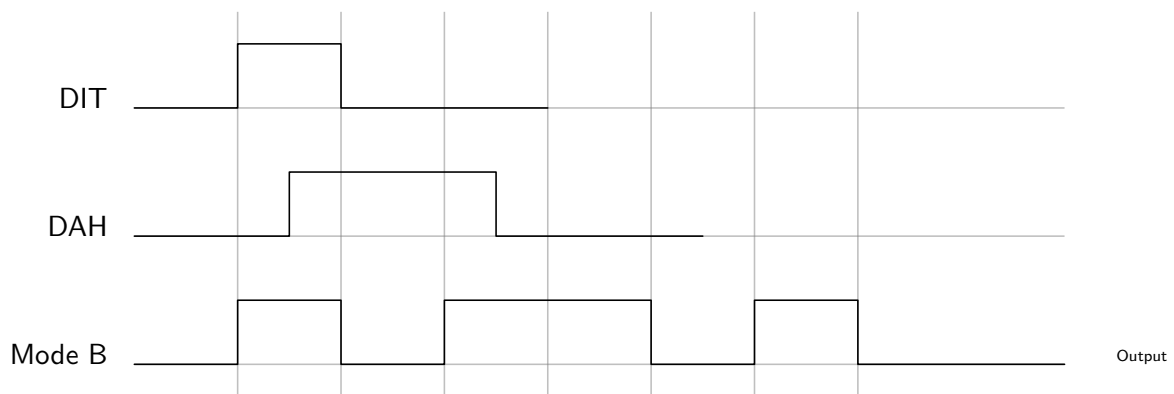


Abb. 3.2: Diagramm Mode B

4 Bedienung

4.1 Tastaturbelegung

4.1.1 Übersicht

5 Schaltung

5.1 Beschreibung

Prg.	Beschreibung	Wertebereich	Standard
------	--------------	--------------	----------

Tab. 5.1: Programmierpunkte Teil 1

6 Beschreibung der Hardware

Klemme	Funktion	Beschreibung	Prg.-Punkt
--------	----------	--------------	------------

Tab. 6.1: Klemmenbelegung

7 Beschreibung der Software

7.1 Timer 1

Der Timer 1 ist ein 16Bit Timer. Dieser wird für die Erzeugung von 2 Zeiten verwendet. Der Timer löst jeweils beim Erreichen der Zeit einen Interrupt aus. Die Interrupts werden alle 1ms und 20ms ausgelöst. So können einfach Interrupt gesteuerte Zeiten verwendet werden.

7.1.1 Timer einstellen

$$f_{OCnA} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRnA)} \quad (7.1)$$

7.2 Sinus Mithörton durch Pulsweitenmodulation

Der BJ-Keyer erzeugt einen Mithörton mit Sinuskurve, statt dem vielfach verwendeten Rechtecksignal. Der Klang eines Sinussignals ist angenehmer. Um mit dem Mikrocontroller ein Sinussignal zu erzeugen, wird die Pulsweitenmodulation verwendet.

7.2.1 Grundlagen

Die Pulsweitenmodulation, kurz PWM genannt, ist eine digitale Modulationsart, bei der eine Spannung zwischen zwei Werten wechselt.

Mit einer konstanten Frequenz wird ein Rechteckimpuls moduliert, bei dem die Weite variiert. Das Verhältnis zwischen Impuls und Pause wird Tastgrad (Duty Cycle) genannt.

7 Beschreibung der Software

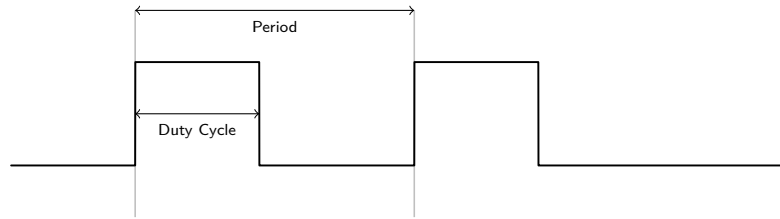


Abb. 7.1: Pulsweitenmodulation

Bei einer Rechteckschwingung gilt für den Tastgrad D :

$$D = \frac{\tau}{T} \quad (7.2)$$

mit τ als Impulsdauer und T als Periodendauer. Mit einem Tastgrad $D = 0,5 = 50\%$ würde ein symmetrischer Impuls erzeugt werden. Der Mikrocontroller schaltet den Ausgang zwischen V_{SS} und V_{DD} .

Die resultierende Ausgangsspannung berechnet sich wie folgt:

$$U_{Out} = \frac{\tau}{T} \cdot U_{In} \quad (7.3)$$

Dabei ist U_{In} gleich V_{SS} . Bei einem Tastgrad von 50% und einer Spannung V_{SS} von 5V beträgt $U_{Out} = 2,5V$. Je länger die Einschaltzeit ist, desto höher ist die effektive Spannung des erzeugten Rechtecksignals, bis zu V_{SS} bei einem Tastgrad von 100%.

Pulsweitenmodulation

Das PWM Signal wird mit Timer 2 des ATmega328P erzeugt. Das PWM Signal wird an PortB Pin 3, OC2A ausgegeben. Es wird der Fast PWM Mode 7 des Controllers verwendet, dabei ist der obere Wert des Timers der Wert im Register OCR2A. Der Ausgang OC2A wird auf den Ausgangswert Toggle konfiguriert, d.h. jedes Mal, wenn der Timer 2 den Wert in OCR2A erreicht, wird der Port umgeschaltet. Es wird ein Rechteck-Signal an PB3 erzeugt, dessen Tastgrad durch OCR2A eingestellt wird. Als Taktquelle wird der CPU Takt verwendet. Dies bedeutet, der Timer 2 läuft ohne einen Vorteiler.

Der maximale Wert für FastPWM berechnet sich wie folgt:

$$f = \frac{f_{Quarz}}{N \cdot 256} \quad (7.4)$$

Der maximale Wert bei einem Quarz mit 16MHz und der minimalen Verteilung von $N = 1$ beträgt somit:

$$\frac{16MHz}{1 \cdot 256} = 62,5kHz \quad (7.5)$$

7.2 Sinus Mithörton durch Pulsweitenmodulation

Am Ausgang von PB3 liegt so bei einem Tastgrad von 50% ein symmetrisches Rechtecksignal mit 62,5kHz an. Der Effektivwert beträgt bei einer Betriebsspannung V_{SS} von 5V = 2,5V. Die 256-1 sind der maximale Wert (256 Werte von 0-255), den OCR2A haben kann (Timer 2 ist ein 8 Bit Timer). Das ist aber nicht das Ziel, da der Keyer

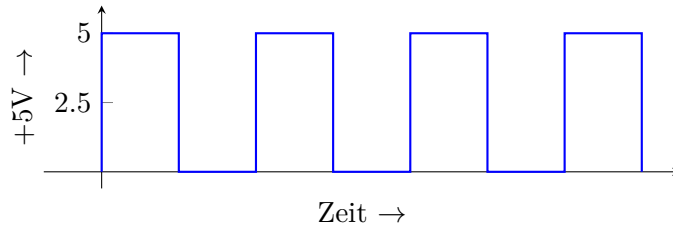


Abb. 7.2: Symmetrisches Rechtecksignal an PB3

ein sinusförmiges Signal ausgeben soll. Um das zu erreichen, muss der Effektivwert der Rechteckspannung an PB3 veränderbar sein. Dies erreicht man durch eine Änderung des Tastgrades. Nun kann ein Mikrocontroller an einem digitalen Ausgangsport keinen Sinus erzeugen, einzig eine Treppe mit einer bestimmten Anzahl an Stufen, an- und absteigend ist möglich.

Wenn der obere Zählwert des Timers 2 in OCR2A verändert wird, ändert sich auch der Effektivwert der Rechteckspannung, durch die Änderung des Tastgrades. Lässt man OCR2A von 0 bis 255 zählen, ändert der Tastgrad sich von 0% bis 100%. Wenn dies über eine Zeitdauer t durch Änderung von OCR2A passiert, dann steigt die effektive Spannung über diese Zeitdauer t von $0V$ - V_{SS} .

$$V_{eff} = U_{max} \cdot \sqrt{Tastgrad} \quad (7.6)$$

Da der digitale Port nur zwischen Low und High wechseln und keine negativen Spannungen erzeugen kann, legt man eine virtuelle Nulllinie auf die Mitte, also auf $2,5V$ ¹. Die $2,5V$ werden bei einem Tastgrad von 50% erreicht, entsprechend einem Wert von 128 in OCR2A.

Erhöht man den Wert von OCR2A in Form einer Sinusfunktion von 128 auf 255 über eine Zeitdauer t , ergibt sich eine ansteigende effektive Spannung in Form einer Sinusfunktion von $2,5V$ auf $5,0V$. Verringert man den Wert von OCR2A von 255 auf 0 in Form einer Sinusfunktion, fällt die effektive Spannung auf $0V$. Durch die passende Änderung von OCR2A in Form einer Sinusfunktion über die Zeitdauer t können somit Effektivspannungen mit 256 Werten dargestellt werden. Je mehr Werte es über die Zeitdauer t sind, umso genauer ist die resultierende Hüllkurve in Form eines Sinus. Die Zeitdauer t , mit der OCR2A mit den Werten einer Sinusfunktion geladen wird, legt die Frequenz des Mithörtons fest. Für die Zeitdauer t wird ein weiterer Timer benötigt. Für einen Mithörton von 600Hz müssen $600 \cdot 256$ Werte pro Sekunde über einen Interrupt in OCR2A geladen

¹es wird im weiteren Verlauf immer von V_{ss} von 5V ausgegangen

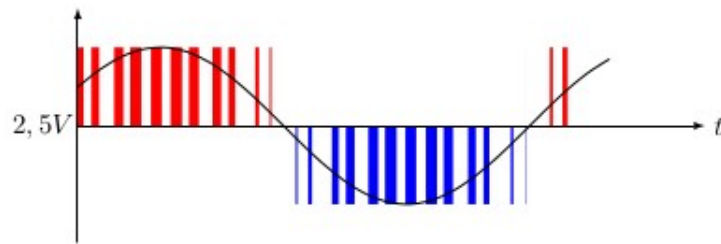


Abb. 7.3: PWM - Tastgrad - Sinus

werden. Je mehr Werte es sind, um so genauer ist die Kurvenform. Für den Timer kann die Zeit wie folgt berechnet werden:

$$600\text{Hz} \cdot 256 = 153,6\text{kHz} \quad (7.7)$$

Die Timer 1 läuft im CTC Modus und es wird ein Output Compare Match Interrupt ausgelöst. Das bedeutet, der Timer läuft bis zum Wert in OCR1A hoch und dann erfolgt der Interrupt. Der Wert für OCR1A wird wie folgt berechnet:

$$f_{OCR1A} = \frac{f_{clk_{I/O}}}{N \cdot (1 + OCR1A)} \quad (7.8)$$

Daraus folgt, daß sich ein Wert von

$$OCR1A = \left(\frac{16\text{MHz}}{8 \cdot 153,6\text{kHz}} \right) - 1 = 12 \quad (7.9)$$

12 für OCR1A ergibt. Allerdings bedeutet eine Frequenz von 153,6kHz für den Timer, dass alle 6,5µs ein Interrupt ausgelöst wird, bei 800Hz Mithörton sind es alle 5µs.

Der Controller läuft mit 16MHz, ein Taktzyklus dauert 62,5ns. Damit bleiben ungefähr 80 Taktzyklen für alle restlichen Aufgaben, wie Tasten abfragen, Display ansteuern, Drehgeber abfragen, CW Zeichen ausgeben. Das ist sehr wenig und die Gefahr besteht, dass Interrupts der Tasteneingänge verloren gehen². Der einzige Weg zur Verlängerung der Zeit zwischen 2 Interrupts besteht in einer Verringerung der Werte für die Sinusfunktion. Mit den 256 Werten ist die Kurvenform zwar fein abgestuft, aber der Controller ist damit am Limit.

Bei einer Verringerung auf 64 Werte für den Sinus ergibt sich dann bei 600Hz eine Zeit von 26µs, das entspricht ungefähr 416 Taktzyklen, was mehr als ausreichend ist. Das der Mithörton dadurch etwas rauher im Klang wird, muss das nachfolgende RLC Filter ausgleichen.

$$600\text{Hz} \cdot 64 = 38,4\text{kHz} \quad (7.10)$$

²was auch in der Praxis bei den Tests so war

7.2 Sinus Mithörton durch Pulsweitenmodulation

Mit einer Tabelle von 64 Werten ergibt sich für den Timer 1 eine Frequenz von 38,4kHz und für OCR1A ein Wert von 51 bei einem Prescaler von 8.

$$OCR1A = \left(\frac{16\text{MHz}}{8 \cdot 38,4\text{kHz}} \right) - 1 = 51 \quad (7.11)$$

Die Berechnung der Werte für OCR1A für unterschiedliche Frequenzen des Mithörtons erfolgt nach dieser Formel mit f_{Sinus} als gewünschte Frequenz des Mithörtons.

$$OCR1A = \left(\frac{f_{clk_I/O}}{N \cdot 64 \cdot f_{Sinus}} \right) - 1 \quad (7.12)$$

f_{Sinus}	OCR1A
1000Hz	30
800Hz	38
600Hz	51
400Hz	77

Tab. 7.1: OCR1A Werte für verschiedene Frequenzen des Mithörtons

7.2.2 Sinustabelle

Über den Overflow-Interrupt vom Timer 1 wird der jeweils nächste Wert einer Sinustabelle in OCR2A geladen. Die Sinustabelle wurde mit einem einfachen Python3 Script erzeugt.

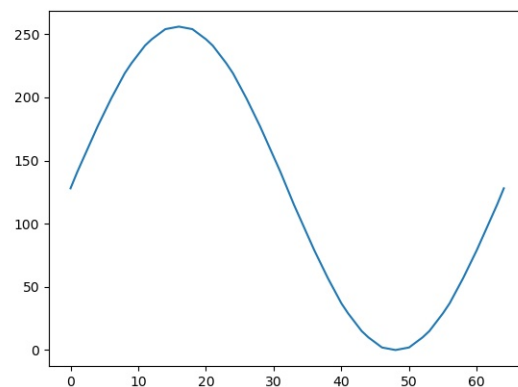


Abb. 7.4: Sinus nach Tabelle vom Python3 Script als Linie

Die 64 Werte vom Python3 Script ergeben sich wie folgt:

7 Beschreibung der Software

```
const unsigned char sinewave[] PROGMEM = {  
0x80,0x8d,0x99,0xa5,0xb1,0xbc,0xc7,0xd1,0xdb,0xe3,0xea,0xf1,0xf6,0xfa,0xfe,0xff, //  
0x100,0xff,0xfe,0xfa,0xf6,0xf1,0xea,0xe3,0xdb,0xd1,0xc7,0xbc,0xb1,0xa5,0x99,0x8d, //  
0x80,0x73,0x67,0x5b,0x4f,0x44,0x39,0x2f,0x25,0x1d,0x16,0x0f,0x0a,0x06,0x02,0x01, //  
0x00,0x01,0x02,0x06,0x0a,0x0f,0x16,0x1d,0x25,0x2f,0x39,0x44,0x4f,0x5b,0x67,0x73, //  
0x80 };
```

In dieser Grafik sind die 64 Werte als Stützpunkte eingezeichnet.

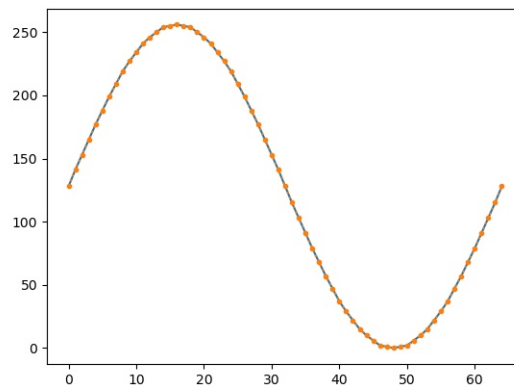


Abb. 7.5: Sinus nach der Tabelle vom Python3 Script mit Stützpunkten

8 Entwicklungsumgebung

Als Entwicklungsumgebung verwende ich mehrere, ausschließlich kostenfreie und überwiegend Open Source Programme:

- Editor vim & neovim
- Shell bash
- Filemanager mc
- Terminalmultiplexer tmux
- RS232 Terminal minicom
- Dokumentation lualatex
- PDF Reader zathura
- Compiler avr-gcc
- Flashprogrammer avrdude
- Layout & Schaltplan KiCad 7.xx
- Bohrschablonen FrontDesigner
- Softwareverwaltung Git
- Softwaredokumentation Doxygen
- Website Nginx & Dokuwiki
- Website Sourcecode Gitea
- Betriebssystem Entwicklung MX-Linux

8 Entwicklungsumgebung

- Betriebssystem Webserver Debian

Wie man sieht, sind das bis auf die CAD Anwendungen und dem PDF Reader alles Anwendungen für die Textconsole. Ich finde, richtig produktiv kann man nur mit der Textconsole arbeiten ;-)

Tabellen

Tab. 5.1: Programmierpunkte Teil 1	13
Tab. 6.1: Klemmenbelegung	15
Tab. 7.1: OCR1A Werte für verschiedene Frequenzen des Mithörtons	21

Abbildungen

Abb. 3.1: Diagramm Mode A	9
Abb. 3.2: Diagramm Mode B	9
Abb. 7.1: Pulswellenmodulation	18
Abb. 7.2: Symmetrisches Rechtecksignal an PB3	19
Abb. 7.3: PWM - Tastgrad - Sinus	20
Abb. 7.4: Sinus nach Tabelle vom Python3 Script als Linie	21
Abb. 7.5: Sinus nach der Tabelle vom Python3 Script mit Stützpunkten	22