DL7BJ Amateur Radio Station

# BJ-Keyer Dokumentation

ab Version 1.00 vom June 28, 2023

Tom, DL7BJ

Mail tom@dl7bj.de
Site https://isnix.de

# Vorwort

Was ist ein elektronischer Morsezeichengeber? Das ist ein Gerät, welches wir Funkamateure besser unter dem Namen Morse-Keyer kennen. Kurzgesagt, 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 Zweihebeltasten 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 Fertiggeräte, 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.



Viel Spaß! Tom, DL7BJ

# Inhalt

# 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-Keyers

### 3.1.1. Zeitverhalten

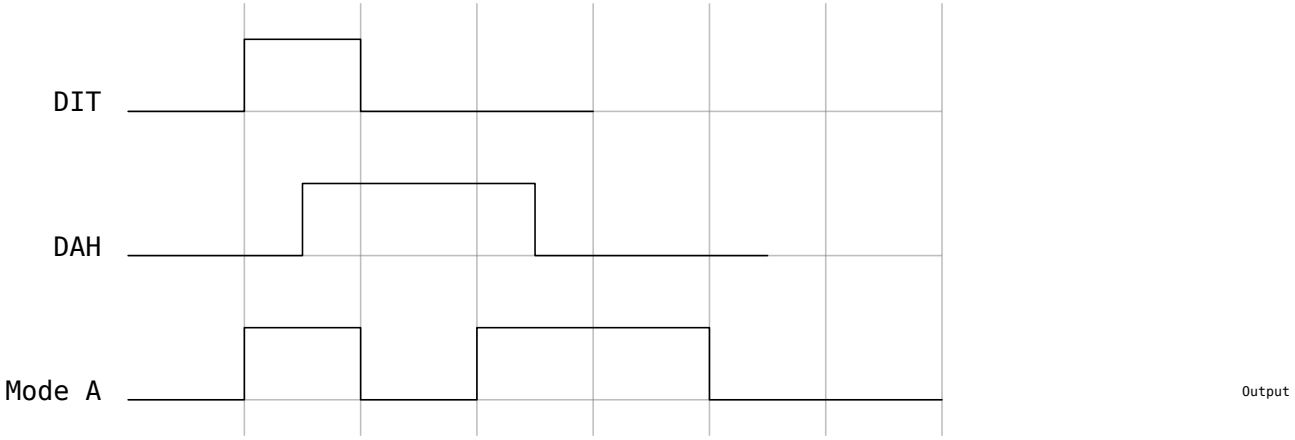


Figure 3.1.: Diagramm Mode A

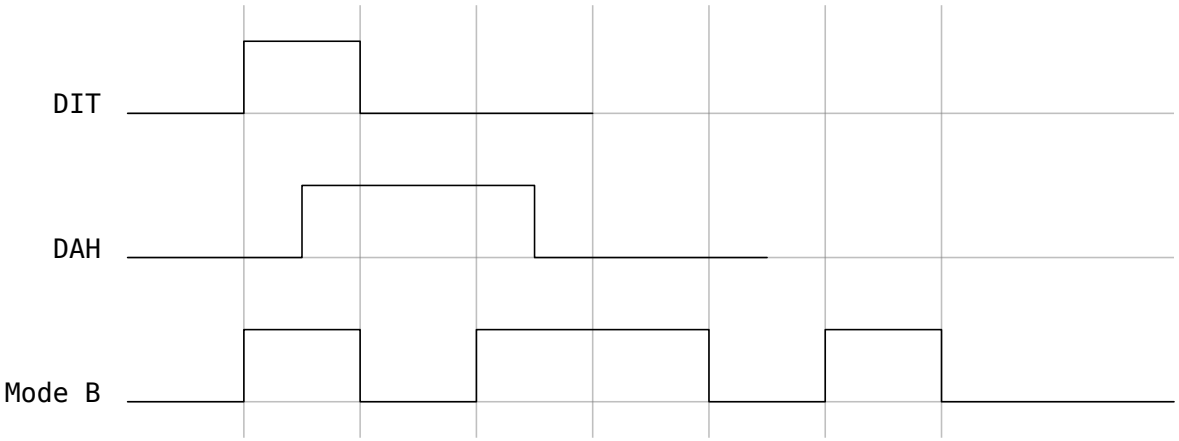


Figure 3.2.: Diagramm Mode B

# 4. Bedienung

## 4.1. Tastaturbelegung

### 4.1.1. Übersicht

# 5. Schaltung

## 5.1. Beschreibung

## 5.2. Schaltplan

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

Table 5.1.: Programmierpunkte Teil 1

# 6. Beschreibung der Hardware

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

Table 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

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)} \tag{7.1}$$

## 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.

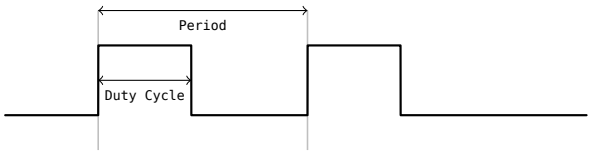


Figure 7.1.: PWM Ausgangssignal

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

Bei einer Rechteckschwingung gilt für den Tastgrad D:

$$D = \frac{\tau}{T} \tag{7.2}$$

mit $\tau$ 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:

$$U_{Out} = \frac{\tau}{T} \cdot U_{In} \tag{7.3}$$

Dabei ist $U_{In}$ gleich $V_{SS}$. Bei einem Tastgrad von 50% und einer Spannung $V_{SS}$ von 5V beträgt $U_{Out}$ = 2,5V.

Pulsweitenmodulation

Das PWM Signal wird mit einem Timer erzeugt. Um die Frequenz des PWM Signals zu verändern, wird die Taktrate und der obere Grenzwert des Zählers eines Timers verändert. Eine Änderung des Output- Compare-Registers ändert das Pausenverhältnis. Der PWM Ausgang des ATMega328 ist High bis der Wert im zugehörigen OCR erreicht ist und Low bis der obere Zählwert erreicht wird. Das ist der Fast-PWM Mode des ATMega328.

Für die Erzeugung des Sinussignals wird der Timer 2 im Fast-PWM Mode verwendet. Der PWM Ausgang des Timers 2 ist OC2A. Die Taktquelle des Timers 2 wird eingestellt und der PWM Mode ausgewählt, so dass OC2A geschaltet wird. Weiter wird der Overflow-Interrupt aktiviert.

Der maximale Wert für FastPWM berechnet sich wie folgt:

$$f = \frac{f_{Quarz}}{2 \cdot 1 \cdot 256} \tag{7.4}$$

Der maximale Wert bei einem Quarz mit 8MHz und der minimalen Vorteilung von 1 beträgt somit:

$$\frac{8.000000Hz}{2 \cdot 1 \cdot 256} = 15.625Hz \tag{7.5}$$

15625Hz entspricht einer Periodendauer von 64µs.

Diese 15.625Hz wären die Samplerate. Für einen Sinuston von 800Hz bei 256 Schritten für die Einzelwerte der PWM wären aber 800 ·256 = 204.800Hz erforderlich. Die einzige Möglichkeit, mit dieser niedrigen Samplerate ist die Verringerung der Schritte. Auf jeden Fall wird die Pulsbreitenänderung im hörbaren Bereich liegen, dies soll das RLC Filter am Ausgang des Controllers bereinigen.

Bei 16 Schritten würde eine Samplerate von 800Hz ·16 = 12.800Hz erforderlich sein. Da ich 800Hz als Mithörton zu hoch empfinde und damit die PWM und die Nachladefrequenz möglichst synchron laufen, habe ich 32 Schritte gewählt. Dies führt dann dazu, dass bei 32 Teilschritten der Sinusperiode pro Teilschritt 4 PWM Impulse erzeugt werden. Damit ist die Nachladefrequenz und die PWM Frequenz synchron und es ergibt sich ein Mithörton mit einer Frequenz von 488.28Hz.

$$\frac{\frac{15.625}{32}}{=} 488.28Hz \tag{7.6}$$

### 7.2.2. Sinustabelle

Über den Overflow-Interrupt vom Timer 2 wird der jeweils nächste Wert einer Sinustabelle in OCR2A geladen. Die Sinustabelle wurde

mit einem einfachen Perl-Script erzeugt und wird als Include Datei eingebunden. Bei jedem Interrupt wird der nächste Wert nach OCR2A geladen. Die Sinustabelle hat 32 Werte. Die Frequenz für das Sinussignal berechnet sich wie folgt:

$$f = \frac{Samplerate}{32} \tag{7.7}$$

Jeder Eintrag der Sinustabelle bestimmt das Pausenverhältnis von Timer 2. Jetzt müssen die 32 Werte zum richtigen Zeitpunkt in OCR2A geladen werden. Dies übernimmt der Timer 0. Timer 0 ist ein 8-Bit Timer und der nötige Werte wären

$$\frac{8.000000Hz}{15625Hz} = 512 \tag{7.8}$$

.

# ListofTables

# ListofFigures

# A. 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

- 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 ;-)