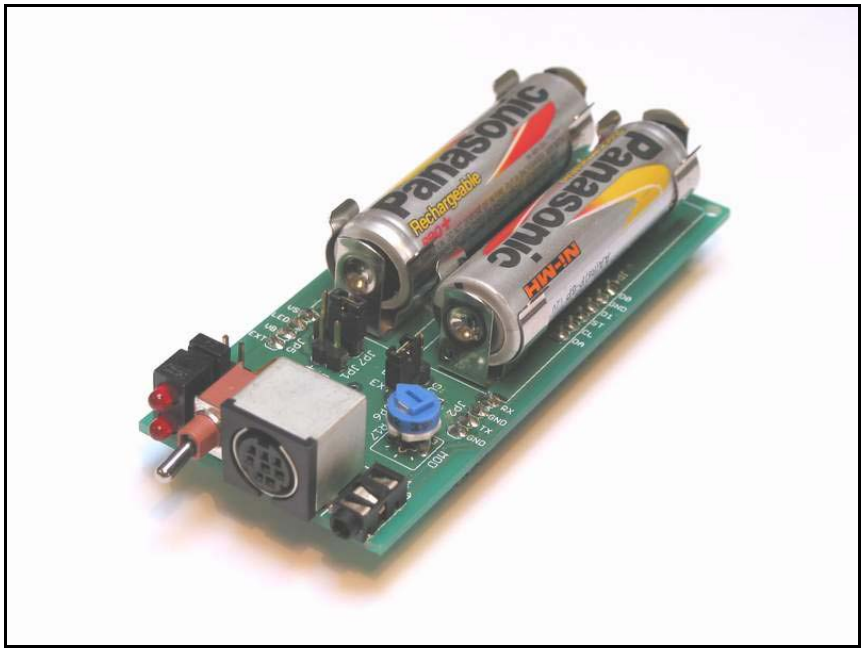
# Mobiler Controller TC100

## Benutzerhandbuch

HW Vers.1.2  SW Vers.1.01




Kurzbeschreibung
Inbetriebnahme
Spezifikationen
Anschlussbeschreibung
Programmierbefehle
Erweiterungsplatine

# Inhalt

# Kurzbeschreibung

Der Universal Controller TC100 ist eine freiprogrammierbare Steuerungseinheit, optimiert für die mobile Informationsübertragung in Funksystemen. Er enthält einen 8051 kompatiblen Prozessorkern mit 2x 12 Bit DAC zur Erzeugung der Modulationssignale sowie insgesamt 4x ADC Eingänge zur Verarbeitung analoger Eingangssignale. Über die RS-232 kompatible Schnittstelle kann ein anwender-spezifisches Programm in das prozessorinterne EEPROM geladen werden. Unter Verwendung eines Standard - Terminalprogramms korrespondiert der PC mit dem TC100, wobei eine einfache Basic-ähnliche Programmiersprache zum Einsatz kommt. Der Controller läuft danach eigenständig mit dem gespeicherten Programm. Durch die geringe Leistungsaufnahme ist das Modul ideal für batteriebetriebene Anwendungen geeignet.

Es stehen 2 Schnittstellen nach außen zur Verfügung:

1. 8-polige Mini-DIN Buchse mit:
   - Externem Versorgungseingang ( +8…+15VDC )
   - Ausgang für interne Betriebsspannung ( 3,3VDC / max. 10mA )
   - PTT Schaltausgang ( max. 25VDC / 100mA )
   - Modulationsausgang ( 0…250mV$_{PP}$ )
   - 2x Messspannungseingang ( 0…2,5VDC / 1MΩ )
   - RXD Signaleingang

2. 3-polige 2,5mm Klinkenbuchse mit RS-232 kompatibler Schnittstelle ( TXD, RXD, GND )

Weiters kann über 4 Stiftleisten (gesamt 20 Pins) eine Erweiterungsplatine aufgesteckt werden, die es ermöglicht, den Controller an alle denkbaren Aufgaben anzupassen. Auf diesem Erweiterungsboard kann u.a. eine weitere Spannungserzeugung für zusätzliche Hardware ( z.B. GPS RX-Modul ) oder ein Sendemodul untergebracht werden. Der interne Datenbus ist dort ebenfalls zugänglich ( LCD-Ansteuerung möglich ).

Programmiermöglichkeiten:

- Ausgabe von Fixtexten in Morsetelegrafie ( A1, F1, F2 )
- Ausgabe von Fixtexten in BPSK31
- Ausgabe von Fixtexten im AX-25 Protokoll ( AFSK, FSK )
- Ausgabe von Ziffern in DTMF Codierung
- Ausgabe von Tönen ( Fixfrequenzen )
- Ausgabe der Temperatur des On - Board Temperatur-Sensors in Morsetelegraphie, BPSK31, DTMF, AX-25, an LCD´s oder seriell
- Ausgabe jeglicher Analogspannungswerte wie oben
- Einlesen von Analogspannungen, arithmetische Bearbeitung und Ausgabe dieser Werte wie oben
- Umwandlung von Analogspannungen in Audiofrequenzen
- Ausgabe von Kombinationen aus Fixtexten und Messwerten
- Senden von Meldungen bei Überschreiten von Grenzwerten
- Kombinationen aus oben aufgeführten Punkten
- Arbeiten in einer Programmschleife oder Aktivierung durch ein externes DC oder NF Signal
- Vielfältige weitere Anwendungen durch freie Programmiermöglichkeit

Weiters können die wichtigsten Parameter wie Modulationspegel, DC – Anteil, Morsetempo, Baudraten bei AX-25, AFSK Frequenzen etc. durch einfache Befehle beliebig konfiguriert und dadurch sehr einfach an bestehende Sender angepasst werden.

Der TC100 Universalcontroller erlaubt außerhalb der aktiven Phasen einen Stromsparmodus, in welchem die Stromaufnahme auf wenige µA reduziert wird und dadurch die Batterie-Lebensdauer extrem verlängert. Das laufende Programm wird dadurch nicht beeinflusst. Während der Stromsparperiode steht ein Shutdown – Signal zur Verfügung, um eventuelle zusätzliche Hardware ( Sensorik etc.) ebenfalls in der Stromaufnahme zu reduzieren.

# Inbetriebnahme

## 1. Verbindung zum PC

TXD ( Pin 2 ), RXD ( Pin 3 ) und GND ( Pin 5 ) der 9-poligen SubD - Buchse des seriellen Schnittstellenkabels mit den gleichnamigen Pins des TC100 verbinden ( JP2 oder X1 )

## 2. PC – Software

Sollte mit dem TC100 eine passende Terminalsoftware mitgeliefert worden sein, kann diese am PC installiert und damit die gesamte Kommunikation und Programmierung vorgenommen werden.
Die für Installation und Betrieb notwendigen Informationen können der entsprechenden Readme - Datei entnommen werden.
Es können auch andere Programme zur Datenkommunikation verwendet werden. Notwendig ist dafür ein Standard Terminal Programm zur Kommunikation auf einem seriellen Com – Port.
Einstellung: 1200 Bps, 8 Datenbits, 1 Stoppbit, kein Paritybit, kein Echo, kein Hardware Handshake.
Nützlich ist auch ein Standard-Texteditor zum Download und Verwalten von bereits geschriebenen Userprogrammen.

## 3. Datenaustausch

Wenn die Datenleitung korrekt mit dem Controller verbunden ist, kann die Betriebsspannung angelegt und der Einschalter betätigt werden. Damit wird ein Reset Impuls erzeugt und ein eventuell bereits im Flash-Speicher ( EEPROM ) geladenes User - Programm startet.
Durch kurzes Überbrücken der Stiftleiste JP8 (PGM) auf dem Board kann der Controller in den Programmier - Mode geschalten werden. Anzeige: **ram mode (list|run|line|ram|test|burn)**
Der Controller steigt damit aus dem im EEPROM abgelegten Userprogramm aus und kann entweder mit der PC-Tastatur Zeile für Zeile oder durch Download eines im Texteditor gespeicherten Programms mit einem Userprogramm geladen werden. Dieses Programm wird im internen RAM abgelegt und geht beim Abschalten wieder verloren. Zur dauerhaften Programmierung dient der Befehl **burn** an erster Stelle.

## 4. Erstellung eines User – Programms

### 4.1 RAM – Mode

Diese Betriebsart kann zum schnellen Test von kleineren Userprogrammen verwendet werden. Hierbei werden die einzelnen Befehlszeilen im prozessorinternen RAM gespeichert. Durch den begrenzten Speicherplatz erlaubt dieser Betrieb allerdings nur ca. 30% des möglichen Userprogrammumfangs. Außerdem gehen die gespeicherten Befehle beim Abschalten der Betriebsspannung des TC100 wieder verloren.

#### 4.1.1 Zeilenweise Eingabe der Befehle mit der Tastatur

Zu Beginn den Befehl **ram** eingeben oder Stiftleiste JP8 (PGM) kurz überbrücken. Nun kann ein Userprogramm im RAM Mode erstellt werden, indem Befehle aus der Liste der möglichen Instruktionen eingegeben werden. Die notwendige Syntax muss dabei beachtet werden. Wenn nicht, erscheint nach betätigen der Enter-Taste in der folgenden Zeile eine Error Meldung. Die Programmeingabe kann jedoch danach ohne Probleme mit der Eingabe der korrekten Programmzeile fortgesetzt werden.
Sollte der zur Verfügung stehende Speicherplatz im RAM überschritten werden, erscheint die Anzeige „memory full". Wenn dennoch im RAM Betrieb weiter gearbeitet werden soll, muss die Anzahl der Programmzeilen reduziert werden. Andernfalls muss das Userprogramm über den Burn Mode in den Controller geschrieben werden. Hierbei steht der gesamte Speicherplatz von 640 Byte zur Verfügung.
Ein Programm, welches im RAM geschrieben und ausgeführt werden soll, wird mit dem Befehl **run** in der letzten Zeile gestartet. Gestoppt wird es wiederum durch kurzes Überbrücken der Stiftleiste JP8 (PGM). Es erscheint wieder das Grundmenü des Programmiermodus **ram mode (list|run|line|ram|test|burn)**
Der Befehl **list** listet die im RAM abgelegten Userprogramm-Befehle zur Überprüfung auf.

4.1.2 Download des Userprogramms als fertiges File

Das Userprogramm kann auch in einem Standard Texteditor geschrieben und als txt - File in den TC100 geladen werden. Damit müssen bei mehrmaligem Umändern des Programms nicht jedes Mal alle Befehlszeilen neu eingegeben werden. Man ändert die jeweilige Zeile im Editor und führt einen File - Download über das Terminalprogramm durch. Durch die schnelle Datenübertragung zum Controller muss jedoch in diesem Fall das sonst in der Übertragung vorgesehene Echo ausgeschaltet werden. Das geschieht einfach durch Verwendung des leicht abgeänderten Befehls **ram&** in der ersten Zeile.

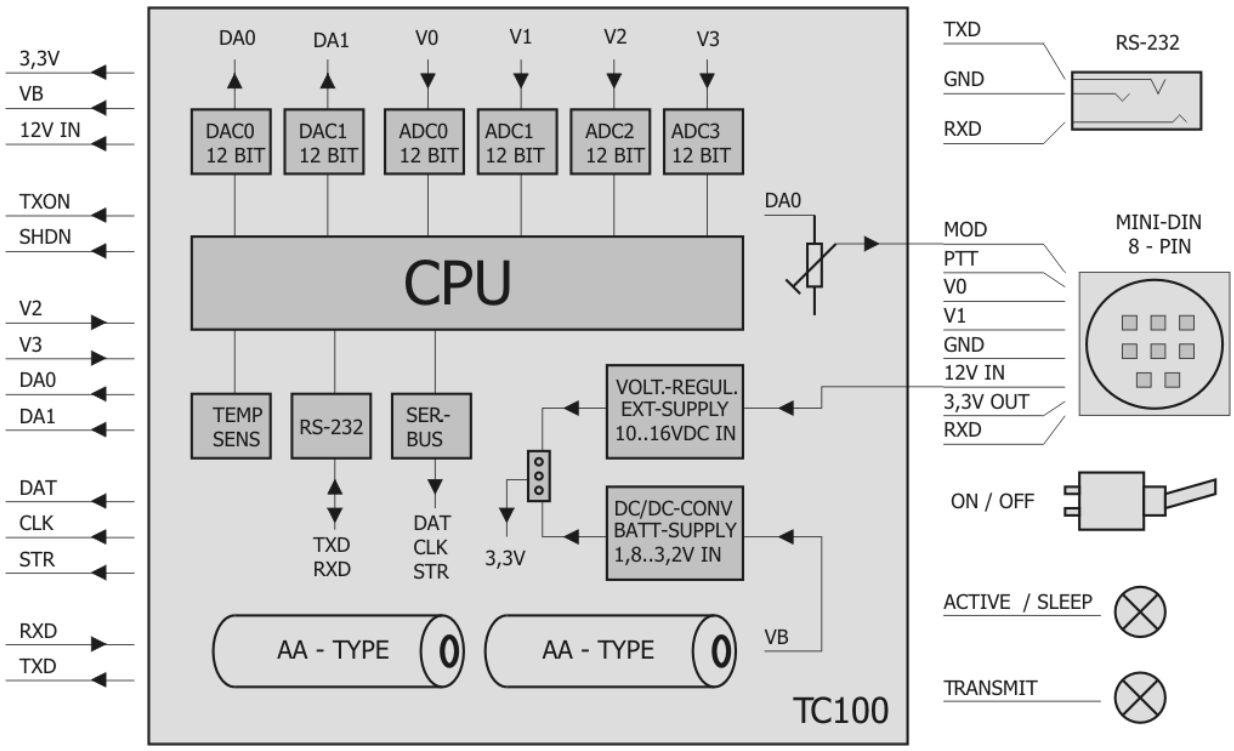
## 4.2 BURN – Mode

Zum Abspeichern eines Userprogramms im EEPROM verwendet man den Befehl **burn**. Alle nun folgenden Programmzeilen werden Zeichen für Zeichen in das EEPROM gebrannt. Durch diese Betriebsart wird nicht nur das Userprogramm unverlierbar in den TC100 geschrieben, sondern es steht auch der gesamte mögliche Speicherplatz zur Verfügung.
Tritt bei dieser Art der Programmierung ein Syntax Fehler auf, erscheint eine Error Meldung und es wird in den Test Mode geschaltet. So kann problemlos die korrekte Eingabeform getestet und danach weiterprogrammiert werden.
Durch Eingabe von **list** im Burn Mode besteht die Möglichkeit, die Anzahl der Programm Bytes abzufragen und das gesamte Userprogramm aufzulisten. Es stehen insgesamt 640 Byte zur Verfügung, die jedoch auch sehr komplexe Programmabläufe ermöglichen
In diesem Betrieb empfiehlt es sich, hier immer den Download aus dem Texteditor zu verwenden. Dazu schreibt man ähnlich wie im RAM Mode in der ersten Zeile den Befehl **burn&**, danach die Befehle des Userprogramms, und beendet mit dem Befehl **end** in der letzten Zeile. Am Bildschirm erscheint danach die Anzahl der programmierten Zeilen.
Das Programm kann nun entweder durch Eingabe von **run** oder durch Ab- und wiederum Einschalten des TC100 gestartet werden. Es startet ab nun immer nach Anlegen der Betriebsspannung.

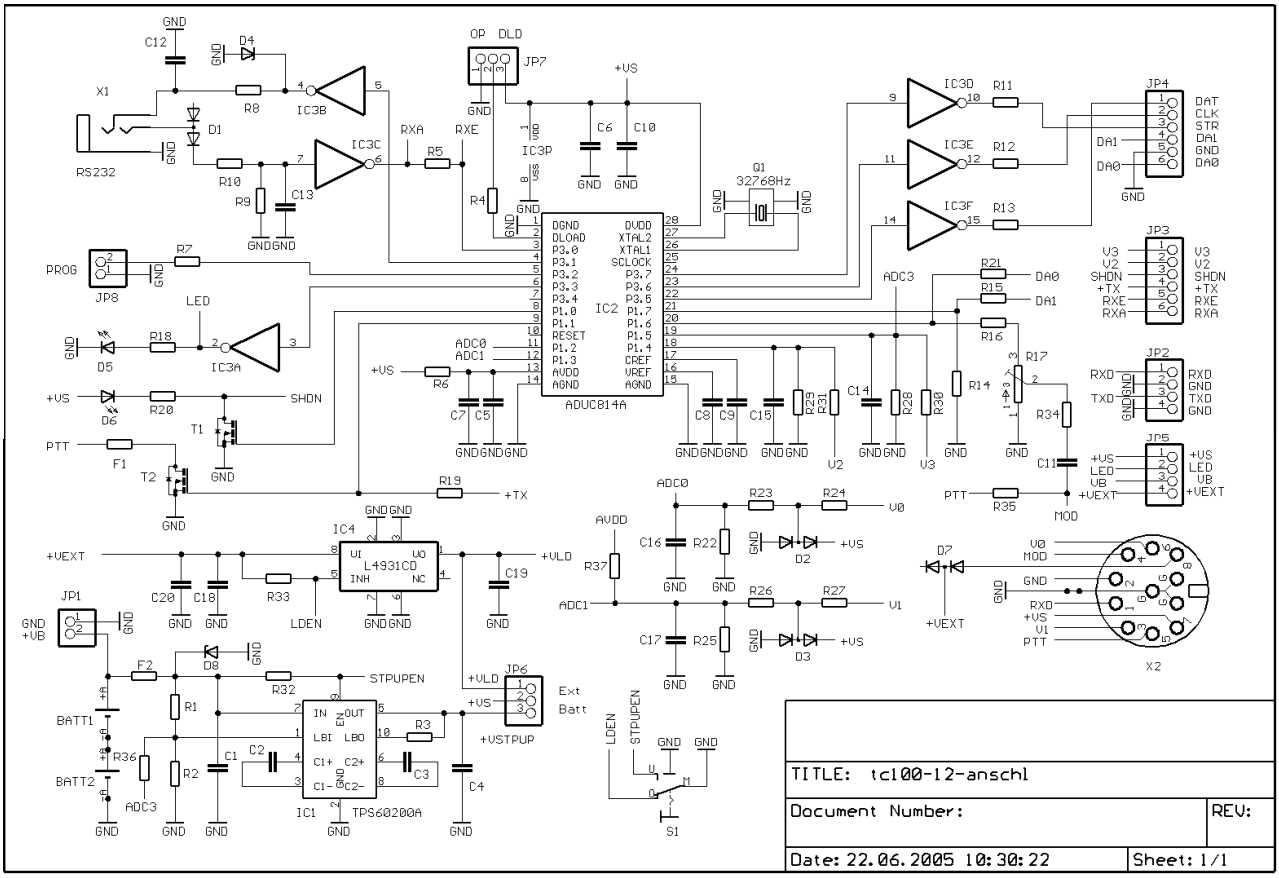
# Spezifikationen

| | |
|---|---|
| Betriebsspannung extern | 8,0 ... 15 VDC |
| Betriebsspannung intern ( Batt. ) | 1,8…3,2 VDC |
| Stromaufnahme ( extern ) | ca. 15 mA |
| Stromaufnahme ( Sleep-Mode ) | ca. 150 µA |
| Stromaufnahme ( Batt. ) | ca. ?mA |
| Eingangsimpedanz der AD-Wandler | ca. 1 MOhm |
| Eingangsspannung der AD-Wandler | max. +2,5 VDC |
| Ausgangsimpedanz der DA-Wandler | ca. 200 Ohm |
| Ausgangsspannung der DA-Wandler | max. +2,5 VDC |
| Schaltstrom bei PTT und SHDN | max. 200 mA ( gegen GND ) |
| Ausgangsimpedanz der Bus-Signale | ca. 470 Ohm |
| Modulationssignal | 0…250m$V_{SS}$ |

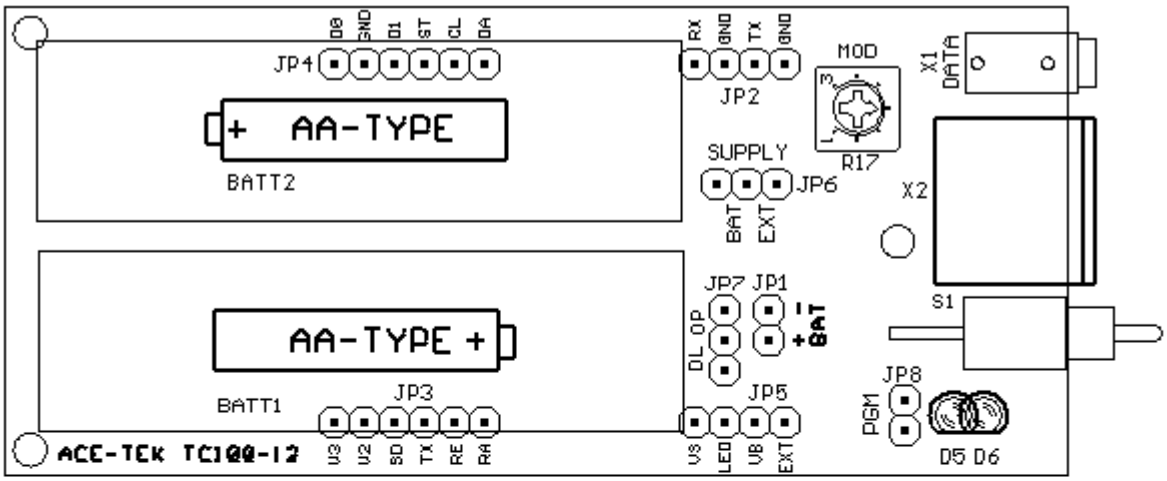
# TC100 Blockdiagramm



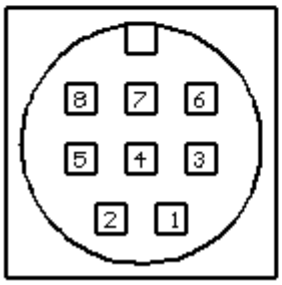
# TC100 Schaltplan



# TC100 Anschlussbeschreibung




X2 ( Mini-DIN 8-pin ):




1...RXD / GPS Data in ( +-3...15V )
2...GND
3...V1 ( 0...2,5VDC )
4...MOD ( 0...250mVpp )
5...PTT ( max. 25V, 200mA )
6...V0 ( 0...2,5VDC )
7...+Vs out ( 3,3VDC / max. 10mA )
8...Vext ( +8...15VDC )

## JP1 ( Anschluss für externe Batterie ):

Direkter Zugang zu den Batterieanschlüssen, max. 3,2VDC!

## JP2 ( Anschluss für Erweiterungsplatine ):

GND:
Allgemeine Versorgungsmasse, ist direkt mit dem Minuspol der Betriebsspannungsquelle verbunden
TXD:
TX-Daten der RS-232 kompatiblen Schnittstelle, Pegel 0/+3,3V, Ausgangswert der Baudrate 1200
RXD:
RX-Daten der RS-232 kompatiblen Schnittstelle, möglicher Pegel ±3..15V, Ausgangswert der Baudrate 1200

## JP3 ( Anschluss für Erweiterungsplatine ):

RXA:
Ausgang für invertiertes RXD Signal zur XT Platine, Pegel 0/+3,3V
RXE:
Eingang für invertiertes RXD Signal von der XT Platine, Pegel 0/+3,3V
+TX:
+3,3V bei PTT=ein
SHDN:
Shutdown Pin für zusätzliche Elektronik. Open Drain Ausgang zur Reduktion des Stromverbrauchs
während der „Sleep" – Phase ( schaltet im Betrieb gegen GND ), maximal +24VDC / 200mA
V2:
Eingang in AD-Wandler ( Port 18 ), Eingangsimpedanz statisch ca. 1M // 30pF, Vin > +VS und < 0V vermeiden
V3:
Eingang in AD-Wandler ( Port 19 ), Eingangsimpedanz statisch ca. 1M // 30pF, Vin > +VS und < 0V vermeiden

## JP4 ( Anschluss für Erweiterungsplatine ):

DAT:
Daten – Signal des seriellen TX – Steuerbus ( kann auch als digitales Ausgangssignal frei programmiert werden )
CLK:
Clock - Signal des seriellen TX – Steuerbus ( kann auch als digitales Ausgangssignal frei programmiert werden )
STR:
Strobe - Signal des seriellen TX – Steuerbus ( kann auch als digitales Ausgangssignal frei programmiert werden )
DA0:
Analogausgang für Sendemodulation ( Hauptsignal ), Ausgangswid. 200 Ohm, Vmax = Vref (2,5V) oder +VS ( einstellbar )
DA1:
Analogausgang für Sendemodulation ( Hilfssignal ), Ausgangswid. 200 Ohm, Vmax = Vref (2,5V) oder +VS ( einstellbar )
GND:
Allgemeine Versorgungsmasse, ist direkt mit dem Minuspol der Betriebsspannungsquelle verbunden

## JP5 ( Anschluss für Erweiterungsplatine ):

+VS:
Interne Versorgungsspannung des TC100, +3,3V
LED:
+3,3V bei LED=ein, Ausgangszustand ist: LED=ein bei PTT=ein, kann im Userprogramm geändert werden
VB:
Batteriespannung über F2 gesichert
+VEXT:
Externe Betriebsspannung von X2 nach Diode D7, +8…15VDC/max.200mA

## JP6 ( Jumper für Versorgungsauswahl ):

EXT:
Jumper auf EXT zur Versorgung des TC100 mit +8…15VDC über Mini-DIN Buchse ( X2 )
BAT:
Jumper auf BAT zur Versorgung des TC100 mit 2xAA Batterien ( TC100B )

## JP7 ( Jumper für Firmware-Download ):

OP:
Jumper auf OP zum normalen Betrieb des TC100
DL:
Jumper auf DL zum Laden einer neuen Firmwareversion ( nicht durch Anwender )

## JP8 ( Jumper für Programmiermode ):

Jumper kurz überbrückt schaltet den TC100 in den Programmiermode ( Menüaufruf )

# Programmierbefehle

## Konfigurationsbefehl

**cfg** Dieser Befehl kann zu Beginn des Userprogramms bestimmte voreingestellte Werte verändern. Er braucht nicht verwendet zu werden, wenn die Standardeinstellung für die Anwendung verwendbar ist.

Ausgangswerte:

- Schaltsignal SDN kommt bei Sleep Mode
- LED leuchtet wenn PTT schaltet
- Programmiermode auch über Terminalbefehl einschaltbar
- max. Ausgangsspannung der D/A-Wandler = Vref ( 2,5V )

Syntax: **cfg** <**Binärwert**> z.B. **cfg 1100000111b** ( 10 Bit, davon 5 verwendet, restliche 5 Bits = 0 )

| | | |
|---|---|---|
| Bit 0 ( **000000000Xb** ) : | 0: SDN Signal im Sleep Mode | 1: SDN durch Userprogramm bestimmt |
| Bit 1 ( **00000000X0b** ) : | 0: LED leuchtet bei PTT | 1: LED durch Userprogramm bestimmt |
| Bit 2 ( **0000000X00b** ) : | 0: Prog.-Mode zus. über Terminal | 1: Prog.-Mode nur über Jumper JP8 |
| Bit 3…7 ( **00XXXXX000b** ) : | nicht verwendet, bleiben auf 0 | |
| Bit 8 ( **0X00000000b** ) : | 0: DAC0 max. bis Vref ( 2,5V ) | 1: DAC0 max. bis +Vs |
| Bit 9 ( **X000000000b** ) : | 0: DAC1 max. bis Vref ( 2,5V ) | 1: DAC1 max. bis +Vs |

## Schreib-/ Lese – Variable

**a** … **j** Universell einsetzbare Variable SYNTAX : **a=<x>** ( 16 Bit ) oder Verknüpfung aus zwei 16 Bit Variablen oder Konstanten ( z.B. **a=b+c** oder **e=e+1** )

**p0**, **p1**, **p2**, **p3** Portadressen wie 8051 – Standard, hier p1 und p3 extern vorhanden
SYNTAX: wie bei den universellen Variablen
**Achtung:** Vorgang kann unzulässige Zustände an den Ports zur Folge haben! Nur verwenden, wer genau mit der Hardware vertraut ist und über fundierte Kenntnisse der 8051 Programmierung verfügt

**x**, **xl** String Variable, Länge der Stringvariablen, siehe auch Befehle **rdln, rs** und **pr**, Lese - Variable **xm** bzw. Beispielprogramme

**lev** Änderung der Modulationsamplitude SYNTAX: **lev=0** bis **lev=255** für 0…2,5V an DA0

**spd** Änderung der Gebegeschwindigkeit bei CW oder Dauer des Beep – Signals
( siehe auch **cwa1, cwf1, cwf2** und **beep** ) , SYNTAX: **spd=0** bis **spd=255**

**dc** Gleichspannungsanteil am Modulationssignal SYNTAX: **dc=0** bis **dc=255** für 0..2,5V

**f0** Fixiert eine Audiofrequenz zur Ausgabe an DA0, Schrittweite=0,2Hz, z.B. **f0=5000** für eine Audiofrequenz von 1 kHz, Ausgabe erfolgt über den **tone** Befehl ( gilt nicht für AFSK )

**f1** Definiert die Umtastfrequenz bei AFSK

**txd** TX – Delay für die Betriebsarten PR ( AX-25 ), PSK und DTMF, allerdings unterschiedliche Einheiten für jede Betriebsart SYNTAX: **txd=0** bis **txd=255**

**tail** Nachlaufzeit für die Betriebsart PR ( AX-25 ) SYNTAX: **tail=0** bis **tail=255**

**devi** Spannungshub bei analoger FSK ( von Null Volt aufwärts ) SYNTAX: **devi=0** bis **devi=255**

**port** Selektiert den gewünschten Analogport, 0=V0, 8=Chip-Temp-Sensor, 17…19=V1…V3
SYNTAX: **port=0, port=8** bzw. **port=17** bis **port=19**
Ausgangsport ist port0, der zuletzt gewählte port wird beibehalten

**rs** Letztes Byte der Daten an der seriellen Schnittstelle. Kann auch verwendet werden, um Bytes zu senden

**th0** Samplingrate für Modulation, verändert alle Gebegeschwindigkeiten und Frequenzen

**shft** Sendet Daten an ein externes Schieberegister über den seriellen Port (CLK, DAT, STR)
SYNTAX: **shft=0** bis **shft=255** bzw. **shft=00000000b** bis **shft=11111111b**
Ansteuerung des Schieberegisters siehe unter Anschaltungsvarianten, 5. Anschaltung eines LC Displays ( Handbuch UC100 )

## Lese – Variable

**adc** Stellt den Spannungswert am selektierten Port dar, kann beliebig weiterverarbeitet werden
Ausgabewert (dezimal ) liegt zwischen 0 und 4095 für Spannungen zwischen 0 und 2,5V DC.

**temp** Chiptemperatur 3-stellig auf 0.1 Grad C aufgelöst, ohne Komma ( ist der bereits intern umgerechnete Messwert an Port 8 ) d.h. zur korrekten Ausgabe dieses Wertes muss eine Kommastelle gesetzt werden (z.B. **pr %d31:temp** gibt die Chiptemperatur über die RS-232 Schnittstelle folgendermaßen aus: **21.3**) Toleranzbereich ca. +- 1,5°C, abhängig von der momentanen Stromaufnahme der CPU.

**xm** maximale Größe der Stringvariablen ( Größe des FIFO Puffers )

## Schreib – Variable

**dac0** Gibt am Ausgang DA0 eine DC zwischen 0 und 2,5V aus
SYNTAX: **dac0=0** bis **dac0=4095** für 0 bis 2,5V

**dac1** Gibt am Ausgang DA1 eine DC zwischen 0 und 2,5V aus
SYNTAX: **dac1=0** bis **dac1=4095** für 0 bis 2,5V

## Procedures

**pr** Gibt eine Datenzeile über die RS-232 Schnittstelle aus
SYNTAX für Fixdaten: **pr <x>** ( x=auszugebende Information )
SYNTAX für Variable: **pr %d<y>:<x>** ( d=dez, y=Ausgabeformat, x=auszugebende Variable )
Ausgabeformatierung :
Immer beginnend mit dem Formatierungszeichen **%**
Danach **d** ( dezimal ), **h** ( hex ), **$** ( Stringvariable x ) oder **n** ( neue Zeile )
Bei Dezimalausgabe **d** folgt die Angabe der benötigten Stellen und bei Bedarf die Kommaposition
z.B. **%d31:temp** schreibt die Chiptemperatur 3-stellig und setzt das Komma auf 0,1 Grad ( xx,x )
bei **%d51:temp** bleibt die Anzeige gleich, jedoch werden zusätzlich 2 Leerstellen vorne mit eingeschoben
**%konstante** und **%variable** geben die betreffenden Werte als ASCII Byte aus

**rdln 0** Startet unlimitierten Einlesevorgang aus der RS-232 Schnittstelle ( FIFO ), kein EOL, stoppt durch **rdln -1**

**rdln 1** Startet Einlesevorgang aus der RS-232 Schnittstelle in den String **x** mit der Länge **xl**, Ende durch CR (13)

**rdln -1** Stoppt den Einlesevorgang, der mit **rdln 0** gestartet wurde

**rdln 2 .. 65535** Wie **rdln 1** jedoch Ende durch CR (13) oder timeout in ms
Achtung: EOL (0Dh) im String mitgespeichert. Löschen durch **xl=xl-1**
Im Falle von timeout wird ein String mit **xl=0** zurückgesendet

**rsbd** Einstellung der RS-232 Baudrate. Ausgangswert ist 1200Bd
SYNTAX: **rsbd=<x>** ( x=Baudrate )

**cwa1** Sendet Fixtext oder Variable in Morsetelegrafie ( Tastung des PTT – Signals ). Gebegeschwindigkeit ist ca. 60 BpM, kann mit dem Befehl **spd** verändert werden.
(**spd=70** für ca. 45 BpM, 60 für ca. 55 BpM, 50 für ca. 65 BpM, 40 für ca. 80 BpM, 30 für ca. 105 BpM )
SYNTAX bei Fixtext: **cwa1 <x>** ( x kann Buchstaben, Zahlen oder Sonderzeichen enthalten )
SYNTAX bei Variablen: **cwa1 %d<y>:<x>** ( d=dez, y=Ausgabeformat, x=Variable oder Port )
Genauere Angaben zum Ausgabeformat unter Befehl **pr,** siehe auch Befehle **spd** und **init**

**cwf1** Sendet Fixtext oder Variable in Morsetelegrafie ( Tastung der DC - Spannung auf DA1 ).
Gebegeschwindigkeit ist ca. 60 BpM, kann mit dem Befehl **spd** verändert werden. Ansonsten wie **cwa1**.

**cwf2** Sendet Fixtext oder Variable in Morsetelegrafie ( Tontastung mit Frequenz **f0** ). Gebegeschwindigkeit ist ca. 60 Bpm, kann mit dem Befehl **spd** verändert werden. Ansonsten wie **cwa1**.

**h12** Sendet Fixtext oder Variable im AX-25 Protokoll ( Packet Radio ), AFSK, 1200 Bps
SYNTAX bei Fixtext: **h12 :<X> <Y>:<z>** ( X=Empfänger-Call, Y=Sender-Call, z=beliebiger Text )
SYNTAX bei Variablen: **h12 :<X> <Y>:%d<z>:<a>** ( X,Y wie oben, d=dez, z=Ausgabeformat, a=Variable ) Beispiel : Ausgabe der Chiptemperatur **h12 :RXCALL TXCALL:%d31:temp**
Eingabe von bis zu 8 Digipeatern möglich, z.B. **h12 :RXCALL TXCALL DIGI1 DIGI2:test**
( auf Großschreibung der Calls achten! ).

**h24** Sendet Fixtext oder Variable im AX-25 Protokoll, AFSK, 2400 Bps, SYNTAX wie bei **h12**

**h96** Sendet Fixtext oder Variable im AX-25 Protokoll, FSK, 9600 Bps, SYNTAX wie bei **h12**

**dtmf** Sendet Fixzahlenfolgen oder Variable in DTMF codierten Tönen aus
SYNTAX bei Fixzahlen: **dtmf <x>** ( x = beliebige Zahlenfolge )
SYNTAX bei Variablen: **dtmf %<x>** ( x = definierte Variable, wie bei Befehl **pr** )

**psk** Sendet Fixtext oder Variable in BPSK31
SYNTAX bei Fixtext: **psk <x>** ( x = beliebige Text- oder Zahlenfolge )
SYNTAX bei Variablen: **psk %<x>** ( x = definierte Variable, wie bei Befehl **pr** )

**beep** Sendet einen kurzen Ton SYNTAX: **beep <x>** (x=Tonfrequenz in Hz), Einstellung der Dauer über den Befehl **spd** in 10ms Schritten, z.B. **init beep** gefolgt von **spd=50** für 500ms, danach z.B. **beep 1000**

**tone** Sendet dauernd die Frequenz **f0**, die auch als Variable definiert sein kann. Funktion wie V/f - Converter
SYNTAX: **tone** ( in einer Programmzeile ohne Zusatz, danach ein Label und eine Schleife zur Abfrage der gewünschten Variable. Siehe Beispielprogramme ). **tx 0** beendet die Ausgabe. Bei Verwendung der Befehle **sls** oder **slms** bleibt die zuletzt ausgegebene Tonfrequenz für die gewünschte Dauer erhalten

**f3** Gibt den an einem ausgewählten Port anstehenden Spannungswert 1:1 an DA0 wieder aus

**init** Setzt Ausgangswerte ( Parameter ) für die angegebene Modulation, gilt solange, bis eine andere Modulation aufgerufen wird SYNTAX: **init <x>** ( z.B. **init cwa1** )

**sls** Fügt eine Wartezeit in den Programmablauf ein, UC100 geht in den „Sleep" – Mode
SYNTAX: **sls <x>** ( x = Wartezeit in Sekunden, nur ganze Sekunden möglich, z.B. **sls 2** )

**slms** Wie Befehl **sls**, jedoch Eingabe von Millisekunden

**tx 1** Schaltet den Sender ein

**tx 0** Schaltet den Sender aus

**led 1** Schaltet die LED ein, z.B. für Signalisierungszwecke ( siehe auch Befehl **cfg**)

**led 0** Schaltet die LED aus ( siehe auch Befehl **cfg**)

**shdn 1** Schaltet den SDN Pin durch ( siehe auch Befehl **cfg**)

**shdn 0** Schaltet den SDN Pin wieder aus ( siehe auch Befehl **cfg**)

**wait** Stoppt den Programmablauf, Programm wird erst bei erkennen eines Steuerimpulses am angegebenen Port fortgesetzt. Start durch positive oder negative Flanke kann ausgewählt werden ( Triggerschwelle 1,25V ).
SYNTAX bei steigender Flanke **wait <x>** ( x = Portnummer 0...7, 17..19, z.B. **wait 0** )
SYNTAX bei fallender Flanke **wait <-x>** ( x = Portnummer 0...7, 17..19, z.B. **wait -1**, bei Port 0 nicht möglich, dort nur positive Flanken )
Start durch Tonburst: **wait <frequ>** ( frequ = auszuwertende Tonfrequenz in Hz ), das Programm stoppt, bis am ausgewählten Port die gewünschte Tonfrequenz erscheint und wird danach fortgesetzt.

**lcd** Schreibt Fixtexte oder Variable auf Standard-LCDs
Für diese Funktion ist ein einfaches Hardware Interface in Form eines Schieberegisters notwendig, um die Daten des seriellen Busses in parallele Informationen umzuwandeln
( siehe Handbuch UC100 „Anschaltungsvarianten" ).

Prinzipiell lassen sich die meisten LCD Anzeigen über die verfügbaren Möglichkeiten ansteuern, am günstigsten ist jedoch die Verwendung von zweizeiligen Anzeigen mit je 16 bis 40 Zeichen.

SYNTAX: **lcd** <**string**> zum Schreiben an die erste Position der ersten Zeile ( z.B. **lcd Hallo** )
**lcd %128+x** <**string**> zum Schreiben an eine beliebige Position in der ersten Zeile
z.B. **lcd %130 Hallo** schreibt „Hallo" an die 3. Position der ersten Zeile
**lcd %192+x** <**string**> zum Schreiben an eine beliebige Position in der zweiten Zeile
SYNTAX für Variable: z.B. Temperatur in Zeile 2, Spalte 3 : **lcd %194 %d31:temp C**

Steuerbefehle: Dez. 0…15 je Steuer-Nippel können an das angeschlossene Display gesendet werden.
z.B. Schirm löschen: **lcd %0 %1**
nach manchen Kommandos kann es notwendig sein, eine Pause von mehreren Millisekunden einzuschalten, bis wieder etwas ausgegeben werden kann ( **slms** )
Initialisierung: Initialisierungsstring wird automatisch beim ersten Aufruf des **lcd** Befehls nach dem Einschalten übertragen.

**app** Hängt den folgenden String an den bestehenden String **x** bei Position **xl** ( **xl** wird dabei um die Stringlänge inkrementiert )

**#** Fügt eine Kommentarzeile ein, gilt nicht als Programmschritt, SYNTAX: **#<text>** ( kein Leerzeichen )
Benötigt keinen Speicherplatz im Chip

## Mathematische / Logische 2 Parameter Operanden

| | |
|---|---|
| **+** | Addiert 2 Konstanten oder Variable ( 16 Bit ), z.B. **a=b+1** |
| **-** | Subtrahiert 2 Konstanten oder Variable ( 16 Bit ), z.B. **a=b-1** |
| ***** | Multipliziert 2 Konstanten oder Variable ( je 16 Bit, Ergebnis auf 16 Bit begrenzt ), z.B. **a=adc*2** |
| **/** | Dividiert 2 Konstanten oder Variable ( 16 Bit ), z.B. **b=a/3** |
| **~** | Rechnen mit Bruchzahlfaktor, dient zur vereinfachten Eingabe folgender Operationen:<br>**a=b*~c** entspricht **a=b*(c/65536)** und<br>**a=b~/c** entspricht **a=(b*65536)/c** für positive Zahlen und b<c |
| **#** | Divisionsrest, z.B. **c=a#b** ( 16 Bit ) |
| **&** | Logische UND Verknüpfung, z.B. **a=b&c** ( 16 Bit ) |
| **\|** | Logische ODER Verknüpfung, z.B. **a=b\|c** ( 16 Bit ) |
| **%** | Logische Exklusiv ODER Verknüpfung, z.B. **a=b%c** ( 16 Bit ) |
| **<** | Entscheidungskriterium KLEINER z.B. **if a<b** ( 16 Bit ) |
| **>** | Entscheidungskriterium GRÖSSER z.B. **if a>b** ( 16 Bit ) |
| **=** | Entscheidungskriterium GLEICH z.B. **if a=b** ( 16 Bit ) |
| **<>** | Entscheidungskriterium UNGLEICH z.B. **if a<>b** ( 16 Bit ) |
| **>=** | Entscheidungskriterium GRÖSSER-GLEICH z.B. **if a>=b** ( 16 Bit ) |
| **<=** | Entscheidungskriterium KLEINER-GLEICH z.B. **if a<=b** ( 16 Bit ) |
| **^0** | Bit 0 der rechten Variablen, Bits 1..15 der linken Variablen |
| **^1** | Bit 1 der rechten Variablen, Bits 0, 2..15 der linken Variablen |
| **^2** | Bit 2 der rechten Variablen, Bits 0..1, 3..15 der linken Variablen |
| **^3** | Bit 3 der rechten Variablen, Bits 0..2, 4..15 der linken Variablen |
| **^4** | Bit 4 der rechten Variablen, Bits 0..3, 5..15 der linken Variablen |
| **^5** | Bit 5 der rechten Variablen, Bits 0..4, 6..15 der linken Variablen |
| **^6** | Bit 6 der rechten Variablen, Bits 0..5, 7..15 der linken Variablen |
| **^7** | Bit 7 der rechten Variablen, Bits 0..6, 8..15 der linken Variablen |
| **+^** | Setzt das Bit n der linken Variablen ( z.B. **p3=p3+^a** ) |
| **-^** | Löscht das Bit n der linken Variablen ( z.B. **p3=p3-^a** ) |

## Programm Befehle

| | |
|---|---|
| **if** | Wenn folgender Ausdruck nicht zutrifft, nächste Zeile überspringen ( z.B. **if a=10** ) |
| **goto** | Sprungbefehl zu Label Nr. 1…255 ( z.B. **goto 10** ) |
| **loop** | Sprung zurück zum Programmstart |
| **halt** | Stoppt das laufende Programm, UC100 geht in den „Sleep-Mode" |
| **end** | Letzte Zeile des Userprogramms im BURN - Mode |
| <label>**:** | Definiert den Beginn eines bestimmten Programmabschnitts für Sprungbefehl<br>SYNTAX **<x>:** INFO: x = Beliebige Zahl zwischen 1 und 255 ( z.B. **10:** )<br>Label benötigt eine Programmzeile, nächster Programmschritt in neue Zeile |

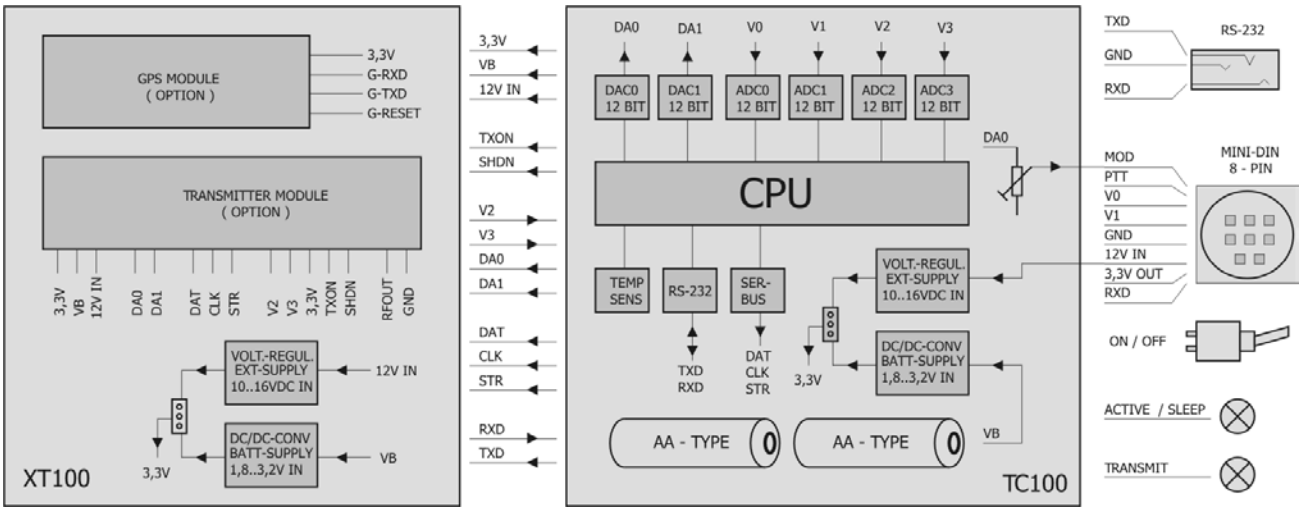
## Interaktive Befehle

| | |
|---|---|
| **list** | Listet das Userprogramm aus dem RAM auf dem Bildschirm |
| **run** | Startet das Userprogramm im RAM |
| **line** | Eingabe einer Programmzeile und sofortige Ausführung |
| **test** | Syntax-Test einer Programmzeile, Echo ein für Tastatureingabe |
| **test&** | Syntax Test einer Programmzeile, Echo aus für Eingabe aus Datei |
| **burn** | Speichern der folgenden Programmzeilen in das EEPROM, Echo ein für Tastatureingabe. Programmeingabe durch den Befehl **end** abschließen. |
| **burn&** | Speichern der folgenden Programmzeilen in das EEPROM, Echo aus für Eingabe aus Datei. Programmeingabe durch den Befehl **end** abschließen. |
| **ram** | Löscht die Daten im RAM und ermöglicht die Eingabe eines neuen Userprogramms, erscheint nach kurzem Überbrücken von JP8 (PGM). Echo ein für Tastatureingabe. |
| **ram&** | Löscht die Daten im RAM und ermöglicht die Eingabe eines neuen Userprogramms, erscheint nach kurzem Überbrücken von JP8 (PGM). Echo aus für Eingabe aus Datei. |

# Erweiterungsplatine XT100
( in Vorbereitung )

Um in Kombination mit dem TC100 zusätzliche Funktionseinheiten betreiben zu können, wurde die Erweiterungsplatine XT100 entwickelt. Sie wird über 4 Stück 6- bzw. 4-polige Stiftleisten mit dem TC100 verbunden und hat damit Zugang zu dessen wichtigsten Schnittstellen.
Es wird dadurch z.B.möglich, eine GPS-RX Maus anzuschalten und zu versorgen, ein internes GPS-RX Modul oder einen internen Sender ( oder beides ) zu betreiben. Damit wird der TC100 bei Bedarf zu einer kompletten Funktionsbaugruppe ohne der Notwendigkeit einer externen Anschaltung.

## Blockschaltbild TC100 + XT100




Weitere Informationen in Vorbereitung