

# TPS / MyCo Data Collection

## December 2021 Juergen Pintaske

Recently I fell over Burkhard's [Christmas Caledar 2021 – a veeeery basic robot.](#) This was love at first sight. I loved, it so translated the text into English – see on our [MyCo facebook.](#) There ist no controller and no programming involved – yet. But this kit of parts would be very nice to first work through the 24 experiments, and then add a controller to achieve more – where you would need the same components mostly as interface anyway.

I have used TPS / MyCo quite a bit, and as well microbit and Arduino / nano.

Books related: [Translation](#) – [Including Willie's Emulator](#) – [TPS/MYCO all on Paper](#)

So I wanted to go back to the routes and started to collect data from Burkhard's website.

Next would be to try to understand Bascom – some parts attached here.

But I gave up for now, as the info for Beginners is not good enough at BASCOM for me.

But I have the 4 Parts of TPS code to help. And Burkhard's explanations how it works.

Most of Burkhard's information is unfortunately in German, so I started some translations, to make this interesting project accessible to the English speaking community. These parts can be used as example to program in other languages, e.g. C, (Arduino), Forth, or Assembler.

### Contents of this document:

<a href="#">TPS1 German</a>	2
<a href="#">TPS2 German</a>	7
<a href="#">TPS3 German</a>	12
<a href="#">TPS4 German</a>	16
<a href="#">Mini-TPS ATmega8 German</a>	21
<a href="#">Corrections German</a>	33
<a href="#">TPS1 English</a>	45
<a href="#">TPS2 English</a>	50
<a href="#">TPS3 English</a>	55
<a href="#">TPS4 English</a>	59
More translations to come	
<a href="#">ATmega 8 for translation</a>	64
<a href="#">TPS1-4 just the code</a>	87
<a href="#">Bascom 1</a>	98
<a href="#">Bascom getting started</a>	104
<a href="#">Bascom and Arduino Uno</a>	117

# Die Tasten-programmmierbare Steuerung TPS (TPS 1)

<http://www.elektronik-labor.de/Projekte/TPS1.html>

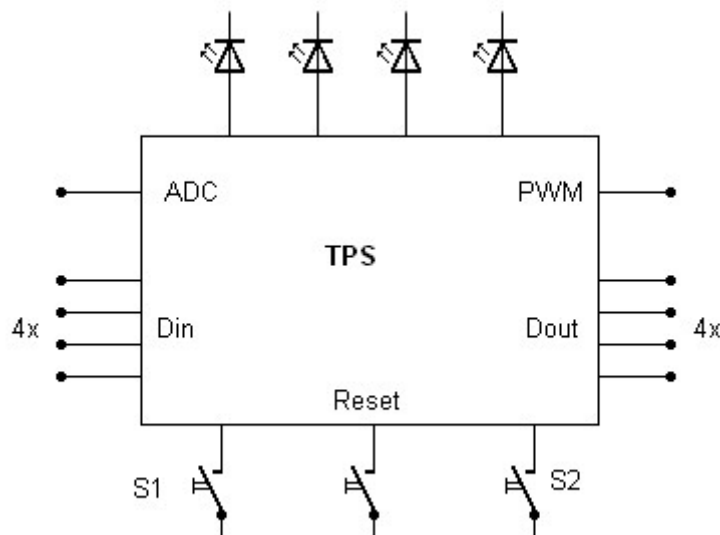
BASCOM <https://avrhelp.mcselec.com/>

English Translation after the German Version

In this document. Work in progress

[Elektronik-Labor](#) [Projekte](#) [TPS](#)

---



<a href="#">TPS 2: Der Programmiermodus</a>	<a href="#">Link hier</a>	<a href="#">German</a>	<a href="#">English</a>
<a href="#">TPS 3: Rechnen mit Variablen</a>	<a href="#">Link hier</a>	<a href="#">German</a>	<a href="#">English</a>
<a href="#">TPS 4: Sprünge und Verzweigungen</a>	<a href="#">Link hier</a>	<a href="#">German</a>	<a href="#">English</a>
<a href="#">Mini-TPS mit ATmega8</a>	<a href="#">Link hier</a>	<a href="#">German</a>	<a href="#">English</a>

Die Idee ist einfach: Ein kleiner Steuercomputer soll ohne PC oder Programmiergerät nur über ein paar Tasten programmiert werden können. Der Befehlsvorrat soll so einfach sein, dass man ihn im Kopf behalten kann, um notfalls ganz ohne Unterlagen ein Steuerprogramm zu entwickeln. Damit alles möglichst klein und überschaubar ist, soll es ein 4-Bit-System werden.

Es gibt vier digitale Ausgänge,  
vier digitale Eingänge,  
intern verarbeitete Daten haben eine Breite von vier Bit,  
und auch die Befehle sind nur mit vier Bit kodiert,  
d.h. es gibt maximal 16 Befehle, die man sich merken muss.

Bei der Programmeingabe hat man ein Display aus vier LEDs, das abwechselnd Befehle und Daten anzeigen soll.

Zur Eingabe braucht man zwei Tasten:

S1 dient zur Eingabe von Daten,

S2 zum Programmieren.

Außerdem muss es noch eine Reset-Taste geben, mit der man wahlweise das Programm startet oder den Programmiermodus einschaltet.

Dieser kleine Steuercomputer kann ganz unterschiedliche Aufgaben erfüllen, von der Alarmanlage bis zur automatischen Akku-Ladestation oder zum Solarregler. Die Idee ist mir im Urlaub gekommen. Ich hatte keinen PC dabei, aber so ein kleines Platinchen passt in jede Reisetasche.

Da könnte man mal dies und das eintippen und dabei seinen Geist trainieren.

Tipp, teripp, tiptipp, schon ist ein kleines Reaktionsspiel programmiert.

Oder man könnte einen Programmierwettbewerb veranstalten, wer löst eine Aufgabe mit den wenigsten Programmschritten...

Bisher gibt es nur die Idee und eine grobe Aufstellung der wichtigsten Befehle.

Etwas ähnliches habe ich vor einiger Zeit schon einmal entwickelt:

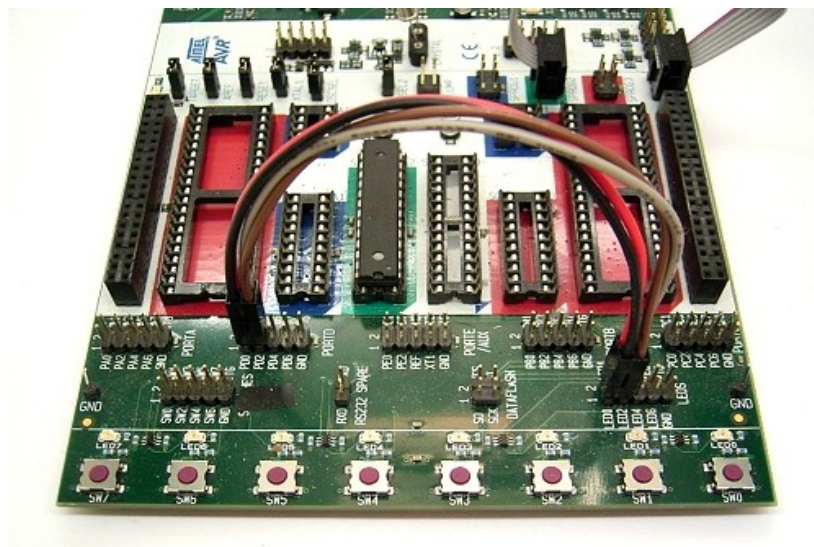
Den [Umwelt-Spion](#) mit mehreren Sensoren und einer einfachen Interpretersprache (Spion-Basic).

Später kam dann noch der [Kosmos-Mikrocontroller](#) mit seinem Kosmos-Basic.

Beides waren 8-Bit-Systeme, die über den PC programmiert werden mussten.

Diesmal will ich es noch einfacher halten, also nur vier Bit und direkte Programmeingabe. Die ganze Entwicklung soll hier in allen Phasen vorgestellt werden.

Damit kann dann jeder noch seine eigenen Befehle hinzufügen und das System an eigene Bedürfnisse anpassen.



Wie fängt man am besten an?

Man könnte natürlich erst mal einen Controller nehmen, auf eine Lochrasterplatine setzen und mit den nötigen drei Tastern und vier LEDs verbinden.

Aber im Moment ist noch nicht einmal klar, welcher Controller es am Ende werden soll.

Deshalb habe ich mich dafür entschieden, die ersten Schritte auf dem STK500 zu entwickeln.

Zufällig war gerade ein ATmega168 in der Fassung, mit dem wird jetzt erst mal gearbeitet.

Und es sollen die Tasten und die LEDs auf dem STK500 verwendet werden.

Zwei Kabel verbinden den Port D mit den LEDs. Verwendet wird D0 bis D4.

Weil die LEDs bei diesem System gegen VCC geschaltet sind, muss die Ausgabe in dieser Entwicklungsphase invertiert werden.

### **Die Firmware soll in Bascom entwickelt werden.**

Im ersten Schritt soll der Interpreter mit nur drei Befehlen (Portausgabe, Wartezeit, Sprungbefehl) angefangen werden.

Den Programmiermodus gibt es noch nicht, aber dafür wird das erste Miniprogramm mit nur fünf Programmschritten direkt ins EEPROM geschrieben.

### **Die ersten drei Befehle lauten:**

**1: Direkte Portausgabe    0...15**

**2: Wartezeit                    0...15**

(1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 30000, 60000 ms)

**3: Sprung zurück            0...15**

Jeder dieser Befehle hat direkt eingegebene 4-Bit-Daten.

Zusammen mit den Daten wird damit jeweils ein Byte belegt.

In hexadezimaler Schreibweise stellt daher das obere Nibble den Befehl und das untere Nibble die Daten dar.

Das erste Testprogramm erzeugt einen einfachen Wechselblinker:

<b>&amp;H 1 1</b>	Portausgabe	1	0001	<b>0001</b>
<b>&amp;H 2 9</b>	Wartezeit	1 s	0010	1001
<b>&amp;H 1 8</b>	Portausgabe	8	<b>1000</b>	
<b>&amp;H 2 9</b>	Wartezeit	1 s		
<b>&amp;H 3 4</b>	Sprung zurück	4		

Download: [TPS1](#)

```

'-----
' Tasten-programmierbare Steuerung TPS
' Test 1: Interpreter, die ersten drei Befehle
'-----

$regfile    = "m168def.dat"      '
$crystal     = 11059200          '
$hwstack    = 32                 '
$swstack    = 64                 '
$framesize  = 64                 '

Dim Addr    As Byte             '
Dim Eebyte  As Byte             '
Dim Dat     As Byte             '
Dim Kom     As Byte             '

Ddrd  = &HFF                    'D0...D1 Outputs
Portd = &H0F                    'STK500 invertiert
Portc = &H0F                    'C0..C3 Inputs mit Pullup

S1 Alias Pinc.3                 'Dateneingabe
S2 Alias Pinc.0                 'Programmieren

Waitms 200                      '

Dat = &H11 : Writeeprom Dat , 0      'Dout=1
Dat = &H29 : Writeeprom Dat , 1      '1000 ms
Dat = &H18 : Writeeprom Dat , 2      'Dout=8
Dat = &H29 : Writeeprom Dat , 3      '1000 ms
Dat = &H34 : Writeeprom Dat , 4      'Adr = Adr - 4

Waitms 200                      '

If S2 = 0 Then                   '
    Goto Programmieren          '
Else                             '
Ausfuehren:                     '
    Addr = 0                    '
    Do                          '
        Readeeprom Eebyte , Addr '
        Addr = Addr + 1          '
        Dat = Eebyte And 15      '
        Kom = Eebyte / 16        '
        If Kom = 1 Then         '1: Direkte Portausgabe

```

```

    Portd = 255 - Dat  'invertierte Portausgabe wegen STK500
End If
If Kom    = 2  Then                                '2: Wartezeit
    If Dat = 0  Then Waitms 1                      '
    If Dat = 1  Then Waitms 2                      '
    If Dat = 2  Then Waitms 5                      '
    If Dat = 3  Then Waitms 10                     '
    If Dat = 4  Then Waitms 20                     '
    If Dat = 5  Then Waitms 50                     '
    If Dat = 6  Then Waitms 100                    '
    If Dat = 7  Then Waitms 200                    '
    If Dat = 8  Then Waitms 500                    '
    If Dat = 9  Then Waitms 1000                   '
    If Dat = 10 Then Waitms 2000                   '
    If Dat = 11 Then Waitms 5000                   '
    If Dat = 12 Then Waitms 10000                  '
    If Dat = 13 Then Waitms 20000                  '
    If Dat = 14 Then Waitms 30000                  '
    If Dat = 15 Then Waitms 60000                  '
End If
If Kom = 3 Then                                    '3: Sprung - relativ
    Addr = Addr - 1                                '
    Addr = Addr - Dat                              '
End If
Loop
End If

Programmieren:                                     '
Do                                                    '
Loop                                                '

End

```

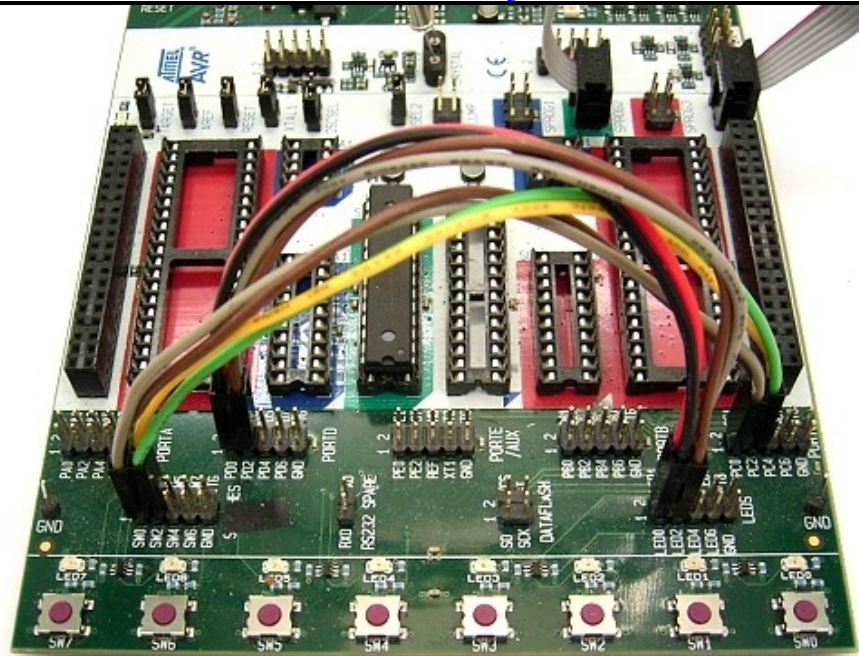
Und tatsächlich, es blinkt. Fertig ist der Mini-Interpreter mit nur drei Befehlen. Damit kann man schon viele unterschiedliche Programme schreiben, vom Lauflicht bis zur Schrittmotorsteuerung.

[weiter](#)



## TPS 2: Der Programmiermodus

[Elektronik-Labor Projekte TPS](#)



Um den Controller über die Tasten programmieren zu können, werden zwei weitere Kabel auf das STK gesetzt. Für die Programmierung werden die Tasten S1 (C3) und S2 (C0) gebraucht. Die Lage der Tasten entspricht dann dem ursprünglichen Entwurf: Dateneingabe links, Programmieren rechts.

In den Programmiermodus gelangt man mit einem Reset bei gedrückter Programmier Taste S2. Man kann nun allein mit der Taste S2 das ganze Programm ansehen. Jede Adresse erfordert dazu zwei Tastenbetätigungen an S2. So wechselt man jeweils in die Anzeige des Befehls und der Daten. Außerdem wird jeweils für kurze Zeit die aktuelle Adresse angezeigt.

- **Erster Tastendruck S2:**
  - Adresse (untere vier Bit) anzeigen, 300 ms
  - Anzeige aus, 300 ms
  - Befehl anzeigen
- **Zweiter Tastendruck S2**
  - Daten anzeigen
- **Dritter Tastendruck S2**
  - Nächste Adresse anzeigen, 300 ms
- usw.

Will man z.B. ein bestehendes Programm mit fünf Schritten nur ansehen, aber nicht verändern, dann gelangt man mit insgesamt zehn Betätigungen von S2 bis ans Ende. Weil jeweils die aktuelle Adresse kurz eingeblendet wird, fällt die Orientierung leicht. Man weiß immer, ob die Anzeige gerade einen Befehl oder Daten darstellt.

Die Taste S1 kommt nur zur Anwendung, wenn man einen Befehl oder seine Daten verändern will. Grundsätzlich können nur Zahlenwerte zwischen Null und 15 eingegeben werden. Mit dem ersten Druck auf S1 wird eine Null eingestellt. Jeder folgende Tastendruck erhöht die Zahl um Eins. Der aktuelle Stand wird jeweils über die vier LEDs binär angezeigt. Will man z.B. eine Vier eingeben, drückt man insgesamt fünfmal auf S1: 0, 1, 2, 3, 4

Wenn auf diese Weise entweder der Befehl oder die Daten oder beides neu eingegeben wurde, führt der zweite Tastendruck auf S2 dazu, dass dieses Byte ins EEPROM programmiert wird. Um das zu verdeutlichen wird die LED-Anzeige für 600 ms abgeschaltet, bevor die nächste Adresse und danach der nächste Befehl angezeigt wird. Diese kleine Pause soll intuitiv als Programmiervorgang verstanden werden. Man kann im Hinterkopf die Vorstellung aufbauen, dass das System die Energie für die Anzeige einspart und für die Programmierung des EEPROMs verwendet. Sowas kennt man ja schon vom Auto: Wenn der Anlasser betätigt wird, geht für einen kurzen Moment das Licht und das Radio aus. Das hilft sehr, wenn man ein schon bestehendes Programm nur an einer Stelle verändern möchte. Mit S2 scrollt man dann bis zur gewünschten Stelle und verändert mit S1 den Befehl oder die Daten, um sie dann mit S2 zu speichern.

Will man ein Programm mit z.B. zwei Bytes (&H17, &H30) komplett neu eingeben, dann muss so getippt werden:

### **Reset+S2    to get into the Programming Mode**

2 x S1	the first one to get to 0, the second one to get to	<b>1</b>
S2	program the first nibble of the first instruction	
8 x S1	the first one to get to 0, and then another 7 to get to	<b>7</b>
S2	program the second nibble of the first instruction	
4 x S1	the first one to get to 0, and then another 3 to get to	<b>3</b>
S2	program the first nibble of the second instruction	
1 x S1	the first one to get to 0,	
S2	program the second nibble of the second instruction	

Wenn man mal zu oft getippt hat, muss man noch einmal ganz rum. Das kennt



man ja von Digitaluhren. Allerdings ist einmal rum nicht 60, sondern in diesem Fall nur 16.

Übrigens, ein Programm mit nur zwei Bytes, was kann das wohl sein. Wenn man sich die Befehle und Daten genau ansieht, tut es dies: Drei LEDs einschalten und dann Ende in Form einer Endlosschleife mit Sprung auf sich selbst.

Download: [TPS2](#)

Programmieren:

```

Addr  = 0
Prog = 0
Do
  Adrlo = Addr And 15           'Adresse anzeigen
  Portd = 255 - Addr
  Waitms 300
  Portd = 255 - 0
  Waitms 200
  Readeeprom Eebyte , Addr
  Dat = Eebyte And 15
  Kom = Eebyte / 16
  Portd = 255 - Kom           'Befehl anzeigen
  Do
    Loop Until S2 = 1
    Waitms 50

  Prog = 1                   'Phase 1: Befehl anzeigen
  Do
    If S1 = 0 Then
      If Prog = 1 Then
        Prog = 2
        Kom = 15
      End If
      If Prog = 2 Then           'Phase 2: Befehl verändert
        Kom = Kom + 1
        Kom = Kom And 15
        Portd = 255 - Kom
      End If
      If Prog = 3 Then :
        'Phase 3: Befehl unverändert, Daten ändern
        Prog = 5
        Dat = 15
      End If
      If Prog = 4 Then
        'Phase 4: Befehl und Daten geändert

```

```

    Prog = 5          '
    Dat = 15          '
End If
If Prog = 5 Then      'Phase 5: Daten verändert
    Dat = Dat + 1     '
    Dat = Dat And 15  '
    Portd = 255 - Dat '
End If
Waitms 50            '
Do
    Loop Until S1 = 1 '
    Waitms 50         '
End If

If S2 = 0 Then        '
    If Prog = 3 Then Prog = 7 '
                                'nur angezeigt, nicht verändert

    If Prog = 1 Then
        Portd = 255 - Dat      '
        Prog = 3              '
    End If
    If Prog = 4 Then
        Portd = 255 - Dat      '
        Prog = 6              '
    End If
    If Prog = 2 Then
        Portd = 255 - Dat      '
        Prog = 4              '
    End If
    If Prog = 6 Then          'nur Kommando wurde verändert
        Dat = Dat And 15      '
        Eebyte = Kom * 16     '
        Eebyte = Eebyte + Dat '
        Writeeprom Eebyte , Addr '
        Portd = 255 - 0       '
        Waitms 600            '
        Addr = Addr + 1       '
        Prog = 0              '
    End If
    If Prog = 5 Then          'Daten wurden verändert
        Dat = Dat And 15      '
        Eebyte = Kom * 16     '
        Eebyte = Eebyte + Dat '
        Writeeprom Eebyte , Addr '
        Portd = 255 - 0       '
        Waitms 600            '
        Addr = Addr + 1       '
        Prog = 0              '

```

```

End If
If Prog = 7 Then      '
    Addr = Addr + 1   '
    Prog = 0          '
End If               '
Waitms 50            '
Do                   '
    Loop Until S2 = 1  '
    Waitms 50         '
End If               '
Loop Until Prog = 0   '
Loop

```

**End**

Das Bascom-Programm arbeitet im Programmiermodus wie eine State-Machine. Es durchläuft bei der Eingabe mehrere Phasen Prog = 0 bis Prog = 7, in Abhängigkeit davon, ob der Speicherringhalt nur angezeigt oder auch verändert werden soll. Die Tasten sind durch eine Wartezeit von 50 ms entprellt.

Und hier einige Programme, die man in diesem Stadium eingeben und ausführen kann:

### **Lauflicht 1:**

&H11, &H28, &H12, &H28, &H14, &H28, &H18, &H28, &H38

### **Lauflicht 2:**

&H11, &H28, &H12, &H28, &H14, &H28, &H18, &H28, &H14, &H28, &H12, &H28, &H3C

### **Zeitschalter, eine Minute:**

&H1F, &H2F, &H10, &H30

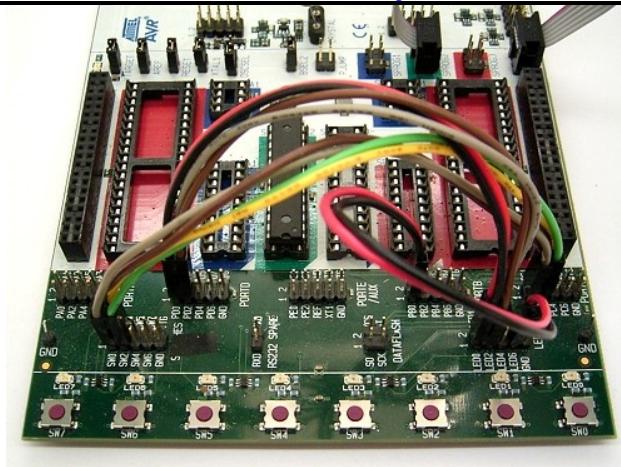
Am Anfang schreibt man sich die Programme vielleicht noch mit Kommentaren auf, aber nach einiger Zeit lernt man digital zu denken und zu fühlen. Mensch und Maschine werden eins, so wie beim Motorradfahren.

[zurück](#)

[weiter](#)

# TPS 3: Rechnen mit Variablen

[Elektronik-Labor](#) [Projekte](#) [TPS](#)



Die Tasten-programmierbare Steuerung soll nun drei Variablen A, B, C und D erhalten. Außerdem kommt ein analoger Eingang und ein PWM-Ausgang hinzu. Beide werden auf 4 Bit begrenzt und sind nur über die Variable A zugänglich ( $A = \text{ADC}$ ,  $\text{PWM} = A$ ). A kann auch direkt mit einer Zahl geladen werden. Um B, C oder D zu füllen, muss man zuerst A laden und den Inhalt dann der anderen Variablen zuweisen. Mit A und B können einige Rechenschritte durchgeführt werden. C und D können als Zwischenspeicher dienen und werden später noch als Zähler für Zählschleifen gebraucht.

**4: A = 0...15**

**5: Ziel 1...9 = A**

(B, C, D,  
Dout, Dout.0, Dout.1, Dout.2, Dout.3,  
PWM)

**6: A = Quelle 1...9**

(B, C, D,  
Din, Din.0, Din.1, Din.2, Din.3,  
ADC)

**7: A = Ausdruck 1...10**

( $A + 1$ ,  $A = A - 1$ ,  $A + B$ ,  $A = A - B$ ,  $A * B$ ,  $A / B$ ,  
 $A \text{ And } B$ ,  $A \text{ Or } B$ ,  $A \text{ Xor } B$ ,  $\text{Not } A$ )

Die folgenden Beispiele stehen auskommentiert am Anfang des Quelltextes. Man kann die Programme über die Tasten eingeben oder (solange der Controller noch auf dem STK steckt) die Kommentarzeichen entfernen und das jeweilige Programm per Bascom ins EEPROM übertragen.

Diese zeitsparende Methode hat sich während der Firmware-Entwicklung bewährt.

'Binärzähler:

```
'Dat = &H71 : Writeeprom Dat , 0      'A = A + 1
'Dat = &H54 : Writeeprom Dat , 1      'Dout = A
'Dat = &H29 : Writeeprom Dat , 2      '1000 ms
'Dat = &H33 : Writeeprom Dat , 3      'Adr = Adr - 3
```

'Analog-Digitalwandler und Ausgangsport

```
'Dat = &H69 : Writeeprom Dat , 0      'A = ADC
'Dat = &H54 : Writeeprom Dat , 1      'Dout = A
'Dat = &H29 : Writeeprom Dat , 2      '1000 ms
'Dat = &H33 : Writeeprom Dat , 3      'Adr = Adr - 3
```

'Digitalwandler und PWM

```
'Dat = &H69 : Writeeprom Dat , 0      'A = ADC
'Dat = &H59 : Writeeprom Dat , 1      'PWM =A
'Dat = &H54 : Writeeprom Dat , 2      'Dout = A
'Dat = &H29 : Writeeprom Dat , 3      '1000 ms
'Dat = &H34 : Writeeprom Dat , 4      'Adr = Adr - 3
```

Download: [TPS3](#)

```
'-----
' Tasten-programmierbare Steuerung TPS
' Test 3: Variablen und Rechenbefehle
'-----
```

```
Dim A As Byte      '
Dim B As Byte      '
Dim C As Byte      '
Dim D As Byte      '
```

```
Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare A Pwm =
Clear Down , Compare B Pwm = Clear Down
Start Timer1
```

...

```
Ausfuehren:      '
    Addr = 0      '
    Do
```

```

Readeeprom Eebyte , Adr      '
Addr = Addr + 1              '
Dat = Eebyte And 15          '
Kom = Eebyte / 16            '

If Kom = 1 Then                '1: Direkte Portausgabe
    Portd = 255 - Dat
                                'invertierte Portausgabe wegen STK500
End If

If Kom = 2 Then                '2: Wartezeit
    If Dat = 0 Then Waitms 1    '
    If Dat = 1 Then Waitms 2    '
    If Dat = 2 Then Waitms 5    '
    If Dat = 3 Then Waitms 10   '
    If Dat = 4 Then Waitms 20   '
    If Dat = 5 Then Waitms 50   '
    If Dat = 6 Then Waitms 100  '
    If Dat = 7 Then Waitms 200  '
    If Dat = 8 Then Waitms 500  '
    If Dat = 9 Then Waitms 1000 '
    If Dat = 10 Then Waitms 2000 '
    If Dat = 11 Then Waitms 5000 '
    If Dat = 12 Then Waitms 10000 '
    If Dat = 13 Then Waitms 20000 '
    If Dat = 14 Then Waitms 30000 '
    If Dat = 15 Then Waitms 60000 '
End If

If Kom = 3 Then                '3: Sprung - relativ
    Addr = Addr - 1            '
    Addr = Addr - Dat          '
End If

If Kom = 4 Then                '
    A = Dat
End If

If Kom = 5 Then                '
    If Dat = 1 Then B = A      'Variablen
    If Dat = 2 Then C = A
    If Dat = 3 Then D = A
    If Dat = 4 Then Portd = 255 - A      'Port
    If Dat = 5 Then
        If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1
                                                'Portbits
    End If
    If Dat = 6 Then            '
        If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1

```

```

End If
If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If
If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If
If Dat = 9 Then
    Dd = A * 17                                ' PWM
    Pwm1a = Dd                                ' PWM
End If
End If

If Kom = 6 Then
    If Dat = 1 Then A = B                      'Variablen
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinb                    'Port
    If Dat = 5 Then A = Portb.0                'Portbits
    If Dat = 6 Then A = Portb.1
    If Dat = 7 Then A = Portb.2
    If Dat = 8 Then A = Portb.3
    If Dat = 9 Then
        Dd = Getadc(4)                        'ADC
        Dd = Dd / 64
        A = Dd
    End If
End If

If Kom = 7 Then
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B
    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B
    If Dat = 9 Then A = A Xor B
    If Dat = 10 Then A = Not A
    A = A And 15
End If

Loop
End If

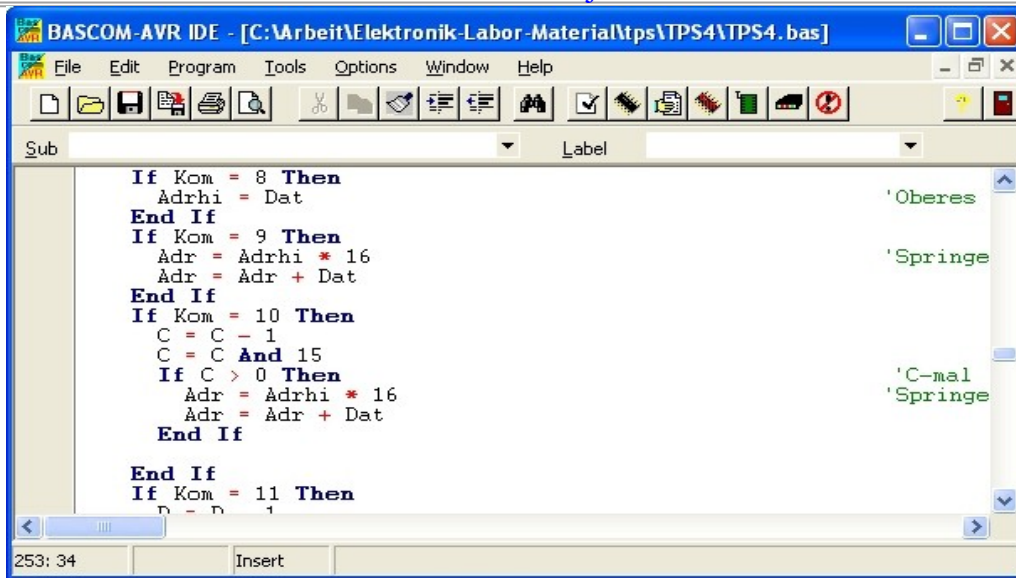
```

[zurück](#)  
[weiter](#)



# TPS 4: Sprünge und Verzweigungen

[Elektronik-Labor Projekte TPS](#)



Bisher gab es nur einen einfachen Rücksprung (Befehl 3), der maximal 15 Adressen zurück reichte. Nun kommt ein absoluter Sprung hinzu.

Da das Sprungziel nur mit 4 Bit angegeben werden kann, gibt es einen zusätzlichen Befehl, der die 4-Bit-Seite, also das High-Nibble der Adresse festlegt. Damit hat man den Adressraum 0...255. Das dürfte für die meisten Anwendungen ausreichen.

Zwei Zählschleifen mit den Variablen C und D führen ebenfalls absolute Sprünge aus, wobei zuvor das High-Nibble der Adresse geladen werden kann.

Die bedingten Sprünge lehnen sich an die Skip-Befehle in AVR-Assembler an. Wenn die jeweilige Bedingung wahr ist, wird eine Adresse übersprungen. Dort könnte z.B. ein Sprungbefehl oder ein Unterprogrammaufruf stehen. Als Bedingungen stehen Vergleiche zwischen A und B sowie direkte Bit-Abfragen des Eingangsports zur Verfügung.

Außerdem gibt es noch einen Unterprogramm-Aufruf und den dazu gehörenden Return-Befehl. Es werden zwar mehrere Unterprogramme erlaubt, aber aus einem Unterprogramm darf kein weiteres Unterprogramm aufgerufen werden, weil der Interpreter sich immer nur eine Rücksprungsadresse merkt.

**8: Adr-high = 0...15**

**9: Direkter Sprung auf Adr-high, Adr-low (0..15)**

**10: Zählschleife C-mal** Adr-high, Adr-low (0..15)

**11: Zählschleife D-mal** Adr-high, Adr-low (0..15)

**12: Bedingte Sprünge:** If (Bedingung 1...11) then Adr = Adr + 1

(A > B,    A > B,    A = B,  
Din.0 = 1, Din.1 = 1, Din.2 = 1, Din.3 = 1,  
Din.0 = 0, Din.1 = 0, Din.2 = 0, Din.3 = 0)

**13: Unterprogrammaufruf** Adr-high, Adr-low (0..15)

**14: Return**

Damit ist die Programmiersprache erst mal vollständig. Erweiterungen sind aber möglich, denn zwei mögliche Befehle (0 und 15) sind noch nicht definiert. Und einige Befehle wie z.B. der Befehl 12 schöpfen noch nicht den möglichen Umfang ihrer Parameter aus. So könnte z.B. der bedingte Sprungbefehl (12) noch einige weitere Bedingungen bekommen. Ähnliches gilt für die Befehle 5, 6 und 7. Man könnte da noch weitere Rechenschritte, mehr AD-Kanäle und einen zweiten PWM-Ausgang definieren.

**Beispiele:**

**'Test: Zählschleife**

```
'Dat = &H45 : Writeeprom Dat , 0           'A=5
'Dat = &H52 : Writeeprom Dat , 1           'C=A
'Dat = &H11 : Writeeprom Dat , 2           'Port=1
'Dat = &H27 : Writeeprom Dat , 3           '200 ms
'Dat = &H10 : Writeeprom Dat , 4           'Port=0
'Dat = &H27 : Writeeprom Dat , 5           '200 ms
'Dat = &HA2 : Writeeprom Dat , 6           'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 7           'Ende
```

**'Test: Unterprogramm**

```
'Dat = &H4F : Writeeprom Dat , 0           'A=15
'Dat = &H52 : Writeeprom Dat , 1           'C=A
'Dat = &H82 : Writeeprom Dat , 2           'Adr 32...
'Dat = &HD0 : Writeeprom Dat , 3           'Call 32+0
'Dat = &H80 : Writeeprom Dat , 4           'Adr 00...
'Dat = &HA2 : Writeeprom Dat , 5           'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 6           'Ende
```

```

'Unterprogramm ab Adresse 32
'Dat = &H11 : Writeeprom Dat , 32      'Port=1
'Dat = &H27 : Writeeprom Dat , 33      '200 ms
'Dat = &H10 : Writeeprom Dat , 34      'Port=0
'Dat = &H27 : Writeeprom Dat , 35      '200 ms
'Dat = &H27 : Writeeprom Dat , 35      '200 ms
'Dat = &HE0 : Writeeprom Dat , 36      'Return

```

```

'Test: Verzweigung, Blinken wenn ADC < 7

```

```

Dat = &H47 : Writeeprom Dat , 0      'A=7
Dat = &H51 : Writeeprom Dat , 1      'B=A
Dat = &H69 : Writeeprom Dat , 2      'A=ADC
Dat = &H82 : Writeeprom Dat , 3      'Adr 32...
Dat = &HC2 : Writeeprom Dat , 4      'if A<B then Adr=Adr+1
Dat = &HD0 : Writeeprom Dat , 5      'Call 32+0
Dat = &H80 : Writeeprom Dat , 6      'Adr 00...
Dat = &H90 : Writeeprom Dat , 7      'Jump 00

```

```

'Unterprogramm ab Adresse 32

```

```

Dat = &H11 : Writeeprom Dat , 32      'Port=1
Dat = &H27 : Writeeprom Dat , 33      '200 ms
Dat = &H10 : Writeeprom Dat , 34      'Port=0
Dat = &H27 : Writeeprom Dat , 35      '200 ms
Dat = &H27 : Writeeprom Dat , 35      '200 ms
Dat = &HE0 : Writeeprom Dat , 36      'Return

```

Download: [TPS4](#)

```

'-----
' Tasten-programmierbare Steuerung TPS
' Test 4: Sprünge und Verzweigungen
'-----

```

```

$regfile    = "m168def.dat"      '
$crystal    = 11059200           '
$hwstack    = 32                 '
$swstack    = 64                 '
$framesize  = 64                 '

```

```

Dim Addr    As Byte             '
Dim Eebyte  As Byte             '
Dim Dat     As Byte             '
Dim Kom     As Byte             '
Dim Adrhi   As Byte             '
Dim Adrlo   As Byte             '
Dim Adrret  As Byte             '

```

```

Dim Prog    As Byte    '
Dim Dd      As Word    '

Waitms 200    '

If S2 = 0 Then
    Goto Programmieren    '
Else
Ausfuehren:    '
    Addr = 0    '
    Do
        Readeeprom Eebyte , Addr    '
        Addr = Addr + 1    '
        Dat = Eebyte And 15    '
        Kom = Eebyte / 16    '
    ...
    If Kom = 8 Then
        Adrhi = Dat    'Oberes Nibble der Adresse
    End If

    If Kom = 9 Then
        Addr = Adrhi * 16    'Springe absolut 0...255
        Addr = Addr + Dat
    End If

    If Kom = 10 Then
        C = C - 1
        C = C And 15
        If C > 0 Then
            Addr = Adrhi * 16    'Springe absolut 0...255
            Addr = Addr + Dat
        End If
    End If

    If Kom = 11 Then
        D = D - 1
        D = D And 15
        If D > 0 Then
            Addr = Adrhi * 16    'Springe absolut 0...255
            Addr = Addr + Dat
        End If
    End If

    If Kom = 12 Then
        If Dat = 1 Then
            If A > B Then Addr = Addr + 1
        End If
    End If

```

```

If Dat = 2 Then
    If A < B Then Addr = Addr + 1
End If
If Dat = 3 Then
    If A = B Then Addr = Addr + 1
End If
If Dat = 4 Then
    If Pinc.0 = 1 Then Addr = Addr + 1
End If
If Dat = 5 Then
    If Pinc.1 = 1 Then Addr = Addr + 1
End If
If Dat = 6 Then
    If Pinc.2 = 1 Then Addr = Addr + 1
End If
If Dat = 7 Then
    If Pinc.3 = 1 Then Addr = Addr + 1
End If
If Dat = 8 Then
    If Pinc.0 = 0 Then Addr = Addr + 1
End If
If Dat = 9 Then
    If Pinc.1 = 0 Then Addr = Addr + 1
End If
If Dat = 10 Then
    If Pinc.2 = 0 Then Addr = Addr + 1
End If
If Dat = 11 Then
    If Pinc.3 = 0 Then Addr = Addr + 1
End If
End If

If Kom = 13 Then
    Addrret = Addr
    Addr = Adrhi * 16      'Call Unterprogramm absolut 0...255
    Addr = Addr + Dat
End If
If Kom = 14 Then
    Addr = Addrret        'Return
End If
Loop
End If

```

[zurück](#)  
[weiter](#)

# Mini-TPS mit ATmega8

[Elektronik-Labor](#) [Projekte](#) [TPS](#)

---



Dies ist eine Miniversion mit dem ATmega8. Alle Ein- und Ausgänge liegen am oberen Rand und wurden mit Serienwiderständen gegen Überlastung geschützt, die Eingänge 2,2 k, die Ausgänge mit 1 k. Die Ausgänge liegen wie bisher auf D0 bis D3, die Eingänge wie bisher auf C0 bis C3.

Abweichend vom ersten Entwurf mit dem Mega168 gibt es diesmal zwei PWM-Ausgänge (PWM1, PWM2) und zwei AD-Eingänge (AD1= ADC4 und AD2 = ADC5). Die Taste S1 liegt jetzt an D6 und S2 an D7. Anders als beim STK500 werden die Portausgänge nicht mehr invertiert. Aber man muss darauf achten, dass die beiden Tasten am gleichen Port liegen und die Pullups eingeschaltet bleiben.

Da die Tasten jetzt nicht mehr am Eingangsport liegen, bietet sich eine direkte Tastenabfrage über die bedingten Sprungbefehle an. Deshalb gibt es jetzt vier neue Optionen für den Befehl 12 (CC bis CF).

## Die erweiterte Befehlsliste

**10 ... 1F: Direkte Portausgabe** 0...15

**20 ... 2F: Wartezeit** 0...15

(1, 2, 5, 10, 20, 50, 100, 200, 500,  
1000, 2000, 5000, 10000, 20000, 30000, 60000 ms)

**30 ... 3F: Sprung zurück** 0...15

**40 ... 4F: A =** 0...15

**51...5A: Ziel** 1...10 = A

51: B = A

52: C = A

53: D = A

54: Dout = A

55: Dout.0 = A.0

56: Dout.1 = A.0

57: Dout.2 = A.0

58: Dout.3 = A.0

59: PWM1 = A

5A: PWM2 = A

**61 ..6A: A = Quelle** 1...10

61: A = B

62: A = C

63: A = D

64: A = Din

65: A = Din.0

66: A = Din.1

67: A = Din.2

68: A = Din.3

69: A = ADC1

6A: A = ADC2

**71 ...7A: A = Ausdruck** 1...10

71: A = A + 1

72: A = A - 1

73: A = A + B

74: A = A - B

75: A = A \* B

76: A = A / B

77: A = A And B

78: A = A Or B

79: A = A Xor B

7A: A = Not A)



**80 ... 8F: Adr-high = 0...15**

**90 ... 0F: Direkter Sprung** auf Adr-high, Adr-low (0..15)

**A0 ... AF: Zählschleife C-mal** Adr-high, Adr-low (0..15)

**B0 ... BF: Zählschleife D-mal** Adr-high, Adr-low (0..15)

**C1 ... CF: Bedingter Sprung:** If (Bedingung 1...15) then Adr = Adr + 1

C1: if A > B then Adr = Adr + 1

C2: if A < B then Adr = Adr + 1

C3: if A = B then Adr = Adr + 1

C4: if Din.0 = 1 then Adr = Adr + 1

C5: if Din.1 = 1 then Adr = Adr + 1

C6: if Din.2 = 1 then Adr = Adr + 1

C7: if Din.3 = 1 then Adr = Adr + 1

C8: if Din.0 = 0 then Adr = Adr + 1

C9: if Din.1 = 0 then Adr = Adr + 1

CA: if Din.2 = 0 then Adr = Adr + 1

CB: if Din.3 = 0 then Adr = Adr + 1

CC: if S1 = 0 then Adr = Adr + 1

CD: if S2 = 0 then Adr = Adr + 1

CE: if S1 = 1 then Adr = Adr + 1

CF: if S2 = 1 then Adr = Adr + 1

**D0 ... DF Unterprogrammaufruf** Adr-high, Adr-low (0..15)

**E0 ... EF: Return**

**Ein Beispiel zur Tastenabfrage S1:** Solange man die Taste drückt, läuft ein schneller Zähler, die LEDs leuchten scheinbar mit halber Helligkeit. Lässt man die Taste los, friert der Zählerstand ein. Damit hat man praktisch ein Zufallsprogramm bzw. einen Würfel von Null bis 15.

'Dat = &HCC : Writeeprom Dat , 0

'Dat = &H71 : Writeeprom Dat , 1

'Dat = &H54 : Writeeprom Dat , 2

'Dat = &H33 : Writeeprom Dat , 3

'if S1 = 1 then Adr = Adr + 1

'A = A + 1

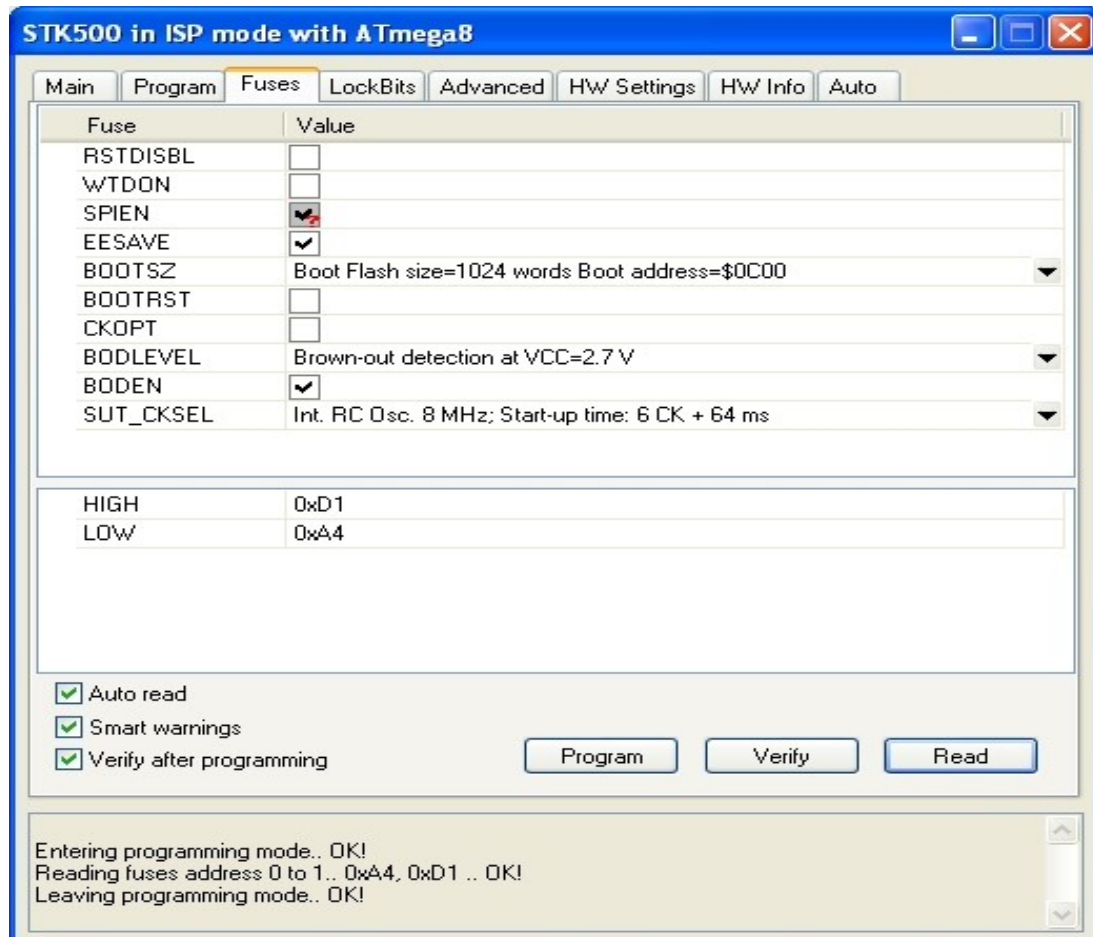
'Dout = A

'Adr = Adr - 3

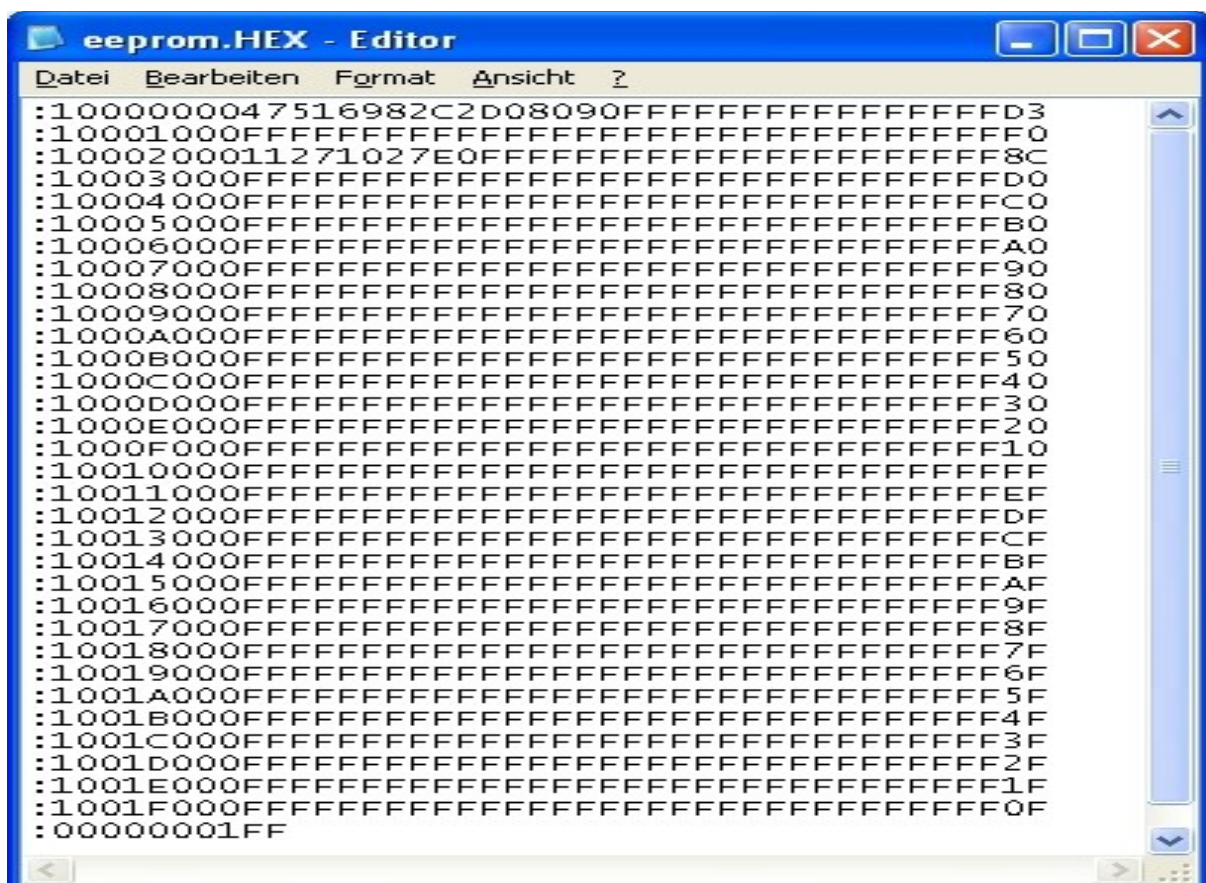
## Die Befehle in einer Programmierkarte

	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	Dout	1 ms	jmp -	A=				AdrHi	jmp	C*	D*	if ...	call	ret
1		2 ms			B=A	A= B	A = A + 1					A > B		
2		5 ms			C=A	A = C	A = A - 1					A < B		
3		10			D=A	A = D	A = A + B					A = B		
4		20			Dout = A	A = Din	A = A - B					Din.0 = 1		
5		50			Dout.0 = A.0	A = Din.0	A = A * B					Din.1 = 1		
6		100			Dout.1 = A.0	A = Din.1	A = A / B					Din.2 = 1		
7		200			Dout.2 = A.0	A = Din.2	A = A And B					Din.3 = 1		
8		500			Dout.3 = A.0	A = Din.3	A = A Or B					Din.0 = 0		
9		1 s			PWM1 = A	A = ADC1	A = A Xor B					Din.1 = 0		
A		2 s			PWM2 = A	A = ADC2	A = Not A					Din.2 = 0		
B		5 s										Din.3 = 0		
C		10										S1 = 0		
D		20										S2 = 0		
E		30										S1 = 1		
F		60										S2 = 1		

Der Controller läuft jetzt mit 8 MHz intern. EESAVE ist aktiviert. Das bedeutet, man kann beim ersten Brennen ein Programm im EEPROM mitgeben, es dann im Quelltext auskommentieren oder löschen und noch einmal brennen.



Insbesondere nützliche Unterprogramme lassen sich auf diese Weise für die spätere Verwendung bereithalten. Das Bild zeigt ein Programm ab 00 in ein Unterprogramm ab &H20, ausgelesen mit dem STK500.



Download: [TPSm8.zip](#)

Und hier das komplette Listing mit allen Anpassungen für den Mega8.

```

'-----
'  Tasten-programmierbare Steuerung TPS
'  ATmega8, intern 8 MHz, 2 ADC, 2 PWM
'-----

$regfile      = "m8def.dat"      '
$crystal      = 8000000          '
$hwstack      = 32               '
$swstack      = 64               '
$framesize    = 64               '


Dim Adr      As Byte      '
Dim Eebyte As Byte      '
Dim Dat      As Byte      '
Dim Kom      As Byte      '
Dim Adrhi    As Byte      '
Dim Adrlo    As Byte      '
Dim Adrret   As Byte      '
Dim Prog     As Byte      '
Dim Dd       As Word      '


Dim A        As Byte      '
Dim B        As Byte      '
Dim C        As Byte      '
Dim D        As Byte      '


Ddrd  = &H0F                'D0...D1 Outputs
Portd = &HF0                'Pullup D4...D7
Portc = &H0F                'C0..C3 Inputs mit Pullup


S1 Alias Pind.6              'Dateneingabe
S2 Alias Pind.7              'Programmieren


Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare A Pwm =
Clear Down , Compare B Pwm = Clear Down
Start Timer1


If S2 = 0 Then

```

Goto Programmieren

Else

Ausfuehren:

Adr = 0

Do

Readeeprom Eebyte , Adr

Adr = Adr + 1

Dat = Eebyte And 15

Kom = Eebyte / 16

If Kom = 1 Then

1: Direkte Portausgabe

Portd = Dat Or &HF0

End If

If Kom = 2 Then

'2: Wartezeit

If Dat = 0 Then Waitms 1

If Dat = 1 Then Waitms 2

If Dat = 2 Then Waitms 5

If Dat = 3 Then Waitms 10

If Dat = 4 Then Waitms 20

If Dat = 5 Then Waitms 50

If Dat = 6 Then Waitms 100

If Dat = 7 Then Waitms 200

If Dat = 8 Then Waitms 500

If Dat = 9 Then Waitms 1000

If Dat = 10 Then Waitms 2000

If Dat = 11 Then Waitms 5000

If Dat = 12 Then Waitms 10000

If Dat = 13 Then Waitms 20000

If Dat = 14 Then Waitms 30000

If Dat = 15 Then Waitms 60000

End If

If Kom = 3 Then

'3: Sprung - relativ

Adr = Adr - 1

Adr = Adr - Dat

End If

If Kom = 4 Then

'4:

A = Dat

End If

If Kom = 5 Then

'5:

If Dat = 1 Then B = A

'Variablen

If Dat = 2 Then C = A

If Dat = 3 Then D = A

If Dat = 4 Then Portd = A Or &HF0

'Port

If Dat = 5 Then

```

    If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1
                                                    'Portbits
End If
If Dat = 6 Then
    If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1
End If
If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If
If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If
If Dat = 9 Then
    Dd = A * 17
                                                    'PWM
    Pwmla = Dd
                                                    'PWM
End If
End If
If Kom = 6 Then
                                                    '6:
    If Dat = 1 Then A = B
                                                    'Variablen
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinb
                                                    'Port
    If Dat = 5 Then A = Portb.0
                                                    'Portbits
    If Dat = 6 Then A = Portb.1
    If Dat = 7 Then A = Portb.2
    If Dat = 8 Then A = Portb.3
    If Dat = 9 Then
        Dd = Getadc(4)
                                                    'ADC
        Dd = Dd / 64
        A = Dd
    End If
    If Dat = 10 Then
        Dd = Getadc(5)
                                                    'ADC
        Dd = Dd / 64
        A = Dd
    End If
End If

If Kom = 7 Then
                                                    '7:
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B
    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B
    If Dat = 9 Then A = A Xor B

```

```

    If Dat = 10 Then A = Not A    '
    A = A And 15
End If

```

```

If Kom = 8 Then                                '8:
    Adrhi = Dat                                'Oberes Nibble der Adresse
End If

```

```

If Kom = 9 Then                                '9:
    Adr = Adrhi * 16                           'Springe absolut  0...255
    Adr = Adr + Dat
End If

```

```

If Kom = 10 Then                               '10:
    C = C - 1
    C = C And 15
    If C > 0 Then                                'C-mal
        Adr = Adrhi * 16                         'Springe absolut  0...255
        Adr = Adr + Dat
    End If
End If

```

```

End If

```

```

If Kom = 11 Then                               '11:
    D = D - 1
    D = D And 15
    If D > 0 Then                                'D-mal
        Adr = Adrhi * 16                         'Springe absolut  0...255
        Adr = Adr + Dat
    End If
End If

```

```

End If

```

```

If Kom = 12 Then                               '12:
    If Dat = 1 Then
        If A > B Then Adr = Adr + 1
    End If
    If Dat = 2 Then
        If A < B Then Adr = Adr + 1
    End If
    If Dat = 3 Then
        If A = B Then Adr = Adr + 1
    End If
    If Dat = 4 Then
        If Pinc.0 = 1 Then Adr = Adr + 1
    End If
    If Dat = 5 Then
        If Pinc.1 = 1 Then Adr = Adr + 1
    End If
End If

```



```

End If
If Dat = 6 Then
    If Pinc.2 = 1 Then Adr = Adr + 1
End If
If Dat = 7 Then
    If Pinc.3 = 1 Then Adr = Adr + 1
End If
If Dat = 8 Then
    If Pinc.0 = 0 Then Adr = Adr + 1
End If
If Dat = 9 Then
    If Pinc.1 = 0 Then Adr = Adr + 1
End If
If Dat = 10 Then
    If Pinc.2 = 0 Then Adr = Adr + 1
End If
If Dat = 11 Then
    If Pinc.3 = 0 Then Adr = Adr + 1
End If
If Dat = 12 Then
    If Pind.6 = 1 Then Adr = Adr + 1
End If
If Dat = 13 Then
    If Pind.7 = 1 Then Adr = Adr + 1
End If
If Dat = 14 Then
    If Pind.6 = 0 Then Adr = Adr + 1
End If
If Dat = 15 Then
    If Pind.7 = 0 Then Adr = Adr + 1
End If

```

```
End If
```

```

If Kom = 13 Then                                '13:
    Adrret = Adr
    Adr = Adrhi * 16    'Call Unterprogramm absolut  0...255
    Adr = Adr + Dat
End If
If Kom = 14 Then
    Adr = Adrret                                'Return
End If

```

```

Loop
End If

```

Programmieren:

```

Adr = 0
Prog = 0

```

```

Do
  Adrlo = Adr And 15                                'Adresse anzeigen
  Portd = Adr Or &HF0
  Waitms 300
  Portd = 0 Or &HF0
  Waitms 200
  Readeeprom Eebyte , Adr
  Dat = Eebyte And 15
  Kom = Eebyte / 16
  Portd = Kom Or &HF0                                'Befehl anzeigen
Do
  Loop Until S2 = 1
  Waitms 50

Prog = 1                                              'Phase 1: Befehl anzeigen
Do
  If S1 = 0 Then

    If Prog = 1 Then
      Prog = 2
      Kom = 15
    End If
    If Prog = 2 Then                                'Phase 2: Befehl verändert
      Kom = Kom + 1
      Kom = Kom And 15
      Portd = Kom Or &HF0
    End If
    If Prog = 3 Then :
      'Phase 3: Befehl unverändert, Daten ändern
      Prog = 5
      Dat = 15
    End If
    If Prog = 4 Then                                'Phase 4: Befehl und Daten geändert
      Prog = 5
      Dat = 15
    End If
    If Prog = 5 Then                                'Phase 5: Daten verändert
      Dat = Dat + 1
      Dat = Dat And 15
      Portd = Dat Or &HF0
    End If
    Waitms 50
  Do
    Loop Until S1 = 1
    Waitms 50
  End If

```

```

If S2 = 0 Then
  If Prog = 3 Then Prog = 7
                                'nur angezeigt, nicht verändert
  If Prog = 1 Then
    Portd = Dat Or &HF0
    Prog = 3
  End If
  If Prog = 4 Then
    Portd = 255 - Dat
    Prog = 6
  End If
  If Prog = 2 Then
    Portd = Dat Or &HF0
    Prog = 4
  End If
  If Prog = 6 Then
                                'nur Kommando wurde verändert
    Dat = Dat And 15
    Eebyte = Kom * 16
    Eebyte = Eebyte + Dat
    Writeeprom Eebyte , Adr
    Portd = 255 - 0
    Waitms 600
    Adr = Adr + 1
    Prog = 0
  End If
  If Prog = 5 Then
                                'Daten wurden verändert
    Dat = Dat And 15
    Eebyte = Kom * 16
    Eebyte = Eebyte + Dat
    Writeeprom Eebyte , Adr
    Portd = 0 Or &HF0
    Waitms 600
    Adr = Adr + 1
    Prog = 0
  End If
  If Prog = 7 Then
    Adr = Adr + 1
    Prog = 0
  End If
  Waitms 50
  Do
    Loop Until S2 = 1
  Waitms 50
End If
Loop Until Prog = 0
Loop

End

```

---

## Korrektur der Firmware, von Michael Gaus

Müsste das statt Pinb nicht Pinc und statt Portb.0...Portb.3 nicht Pinc.0...Pinc.3 heißen, es sollen ja die Eingänge abgefragt werden?

```

    If Kom = 6 Then
        If Dat = 1 Then A = B
'Variablen
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinc
'Port
    If Dat = 5 Then A = Portc.0
'Portbits
    If Dat = 6 Then A = Portc.1
    If Dat = 7 Then A = Portc.2
    If Dat = 8 Then A = Portc.3

```

Antwort: Das stimmt wohl, die Eingaben müssen auf Port C geändert werden. Diese Version für den Mega8 wurde nicht so sorgfältig geprüft wie die Holtek-Version für die Lernpakete. Es würde mich nicht wundern, wenn noch mehr Fehler auftauchen.

Weitere Korrekturen, von Michael Gaus

Ich habe entsprechende Änderungen vorgenommen und mit der neuesten Bascom-AVR Version 2.0.7.5 Demo kompiliert. Dabei wurde angemeckert, dass "Adr" ein "reserved Word" sei, ich habe dann alle "Adr" ersetzt durch "Addr".

Änderungen:

- 1) PINB ersetzt durch PINC, PORTB.x ersetzt durch PINC.x , s.o.
- 2) Skip if S1/S2=0/1 waren alle verdreht abgefragt => korrigiert
- 3) Beim Label "Programmieren" muss es in der 2. Zeile nach dem "Do" Befehl heißen: PORTD = Adrlo OR &HF0 statt Addr
- 4) Bei PWM Config: clear\_down geändert in clear\_up
5. C- und D-Zählschleife am die Holtek-Version angeglichen

Down load: [Tpsm8\\_korrigiert\\_2.zip](#)

'-----

```
' Tasten-programmierbare Steuerung TPS
' ATmega8, intern 8 MHz, 2 ADC, 2 PWM
'-----
```

```
$regfile = "m8def.dat"
$crystal = 8000000
$hwstack = 32
$swstack = 64
$framesize = 64
```

```
Dim Addr As Byte
Dim Eebyte As Byte
Dim Dat As Byte
Dim Kom As Byte
Dim Adrhi As Byte
Dim Adrlo As Byte
Dim Adrret As Byte
Dim Prog As Byte
Dim Dd As Word
```

```
Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim D As Byte
```

```
Ddrd = &H0F                                'D0...D1 Outputs
Portd = &HF0                                'Pullup D4...D7
Portc = &H0F                                'C0..C3 Inputs mit Pullup
```

```
S1 Alias Pind.6                            'Dateneingabe
S2 Alias Pind.7                            'Programmieren
```

```
Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare_a_pwm =
Clear_up , Compare_b_pwm = Clear_up
Start Timer1
```

```
Waitms 200
```

```
'Test: Zählschleife
'Dat = &H45 : Writeeprom Dat , 0           'A=5
'Dat = &H52 : Writeeprom Dat , 1           'C=A
'Dat = &H11 : Writeeprom Dat , 2           'Por=1
'Dat = &H27 : Writeeprom Dat , 3           '200 ms
```

```

'Dat = &H10 : Writeeprom Dat , 4          'Port=0
'Dat = &H27 : Writeeprom Dat , 5          '200 ms
'Dat = &HA2 : Writeeprom Dat , 6          'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 7          'Ende

'Test: Unterprogramm
'Dat = &H4F : Writeeprom Dat , 0          'A=15
'Dat = &H52 : Writeeprom Dat , 1          'C=A
'Dat = &H82 : Writeeprom Dat , 2          'Adr 32...
'Dat = &HD0 : Writeeprom Dat , 3          'Call 32+0
'Dat = &H80 : Writeeprom Dat , 4          'Adr 00...
'Dat = &HA2 : Writeeprom Dat , 5          'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 6          'Ende
'
'                                     Unterprogramm ab Adresse 32
'Dat = &H11 : Writeeprom Dat , 32         'Port=1
'Dat = &H27 : Writeeprom Dat , 33         '200 ms
'Dat = &H10 : Writeeprom Dat , 34         'Port=0
'Dat = &H27 : Writeeprom Dat , 35         '200 ms
'Dat = &H27 : Writeeprom Dat , 35         '200 ms
'Dat = &HE0 : Writeeprom Dat , 36         'Return

'Test: Verzweigung, Blinken wenn ADC < 7
'Dat = &H47 : Writeeprom Dat , 0          'A=7
'Dat = &H51 : Writeeprom Dat , 1          'B=A
'Dat = &H69 : Writeeprom Dat , 2          'A=ADC
'Dat = &H82 : Writeeprom Dat , 3          'Adr 32...
'Dat = &HC2 : Writeeprom Dat , 4          'if A<B then Adr=Adr+1
'Dat = &HD0 : Writeeprom Dat , 5          'Call 32+0
'Dat = &H80 : Writeeprom Dat , 6          'Adr 00...
'Dat = &H90 : Writeeprom Dat , 7          'Jump 00
'
'                                     Unterprogramm ab Adresse 32
'Dat = &H11 : Writeeprom Dat , 32         'Port=1
'Dat = &H27 : Writeeprom Dat , 33         '200 ms
'Dat = &H10 : Writeeprom Dat , 34         'Port=0
'Dat = &H27 : Writeeprom Dat , 35         '200 ms
'Dat = &H27 : Writeeprom Dat , 35         '200 ms
'Dat = &HE0 : Writeeprom Dat , 36         'Return

'Test : Tastenabfrage S1
'Dat = &HCC : Writeeprom Dat , 0          'if S1 = 1 then Adr = Adr+1
'Dat = &H71 : Writeeprom Dat , 1          'A = A + 1
'Dat = &H54 : Writeeprom Dat , 2          'Dout =A
'Dat = &H33 : Writeeprom Dat , 3          'Adr = Adr - 3

```

```

If S2 = 0 Then
  Goto Programmieren
Else
Ausfuehren:
  Addr = 0
  Do
    Readeeprom Eebyte , Addr
    Addr = Addr + 1
    Dat = Eebyte And 15
    Kom = Eebyte / 16

    If Kom = 1 Then                                     '1: Direkte Portausgabe
      Portd = Dat Or &HF0
    End If

    If Kom = 2 Then                                     '2: Wartezeit
      If Dat = 0 Then Waitms 1
      If Dat = 1 Then Waitms 2
      If Dat = 2 Then Waitms 5
      If Dat = 3 Then Waitms 10
      If Dat = 4 Then Waitms 20
      If Dat = 5 Then Waitms 50
      If Dat = 6 Then Waitms 100
      If Dat = 7 Then Waitms 200
      If Dat = 8 Then Waitms 500
      If Dat = 9 Then Waitms 1000
      If Dat = 10 Then Waitms 2000
      If Dat = 11 Then Waitms 5000
      If Dat = 12 Then Waitms 10000
      If Dat = 13 Then Waitms 20000
      If Dat = 14 Then Waitms 30000
      If Dat = 15 Then Waitms 60000
    End If

    If Kom = 3 Then                                     '3: Sprung - relativ
      Addr = Addr - 1
      Addr = Addr - Dat
    End If

    If Kom = 4 Then
      A = Dat
    End If

    If Kom = 5 Then                                     'Variablen
      If Dat = 1 Then B = A
      If Dat = 2 Then C = A
      If Dat = 3 Then D = A

```



```

If Dat = 4 Then Portd = A Or &HF0                                'Port
If Dat = 5 Then
    If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1                'Portbits
End If
If Dat = 6 Then
    If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1
End If
If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If
If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If
If Dat = 9 Then
    Dd = A * 17                                                  ' PWM
    Pwm1a = Dd                                                  ' PWM
End If
End If

If Kom = 6 Then
    If Dat = 1 Then A = B                                        'Variablen
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinc                                    'Port
    If Dat = 5 Then A = Pinc.0                                'Portbits
    If Dat = 6 Then A = Pinc.1
    If Dat = 7 Then A = Pinc.2
    If Dat = 8 Then A = Pinc.3
    If Dat = 9 Then
        Dd = Getadc(4)                                          'ADC
        Dd = Dd / 64
        A = Dd
    End If
    If Dat = 10 Then
        Dd = Getadc(5)                                          'ADC
        Dd = Dd / 64
        A = Dd
    End If
End If

If Kom = 7 Then
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B

```

```

    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B
    If Dat = 9 Then A = A Xor B
    If Dat = 10 Then A = Not A
    A = A And 15
End If

If Kom = 8 Then
    Adrhi = Dat                                'Oberes Nibble der Adresse
End If

If Kom = 9 Then
    Addr = Adrhi * 16                          'Springe absolut 0...255
    Addr = Addr + Dat
End If

If Kom = 10 Then
    If C > 0 Then                                'C-mal
        C = C - 1
        C = C And 15
        Addr = Adrhi * 16                      'Springe absolut 0...255
        Addr = Addr + Dat
    End If
End If

If Kom = 11 Then
    If D > 0 Then                                ' D-mal
        D = D - 1
        D = D And 15
        Addr = Adrhi * 16                      'Springe absolut 0...255
        Addr = Addr + Dat
    End If
End If

If Kom = 12 Then
    If Dat = 1 Then
        If A > B Then Addr = Addr + 1
    End If
    If Dat = 2 Then
        If A < B Then Addr = Addr + 1
    End If
    If Dat = 3 Then
        If A = B Then Addr = Addr + 1
    End If
    If Dat = 4 Then
        If Pinc.0 = 1 Then Addr = Addr + 1
    End If
End If

```

```

End If
If Dat = 5 Then
    If Pinc.1 = 1 Then Addr = Addr + 1
End If
If Dat = 6 Then
    If Pinc.2 = 1 Then Addr = Addr + 1
End If
If Dat = 7 Then
    If Pinc.3 = 1 Then Addr = Addr + 1
End If
If Dat = 8 Then
    If Pinc.0 = 0 Then Addr = Addr + 1
End If
If Dat = 9 Then
    If Pinc.1 = 0 Then Addr = Addr + 1
End If
If Dat = 10 Then
    If Pinc.2 = 0 Then Addr = Addr + 1
End If
If Dat = 11 Then
    If Pinc.3 = 0 Then Addr = Addr + 1
End If
If Dat = 12 Then
    If Pind.6 = 0 Then Addr = Addr + 1
End If
If Dat = 13 Then
    If Pind.7 = 0 Then Addr = Addr + 1
End If
If Dat = 14 Then
    If Pind.6 = 1 Then Addr = Addr + 1
End If
If Dat = 15 Then
    If Pind.7 = 1 Then Addr = Addr + 1
End If

```

```
End If
```

```

If Kom = 13 Then
    Addrret = Addr
    Addr = Addrhi * 16
                                'Call Unterprogramm absolut 0...255
    Addr = Addr + Dat
End If

```

```

If Kom = 14 Then
    Addr = Addrret
                                'Return
End If

```

```
Loop
```

**End If**

## Programmieren:

**Addr = 0**

**Prog = 0**

Do

$$\text{Adrlo} = \text{Addr} \text{ And } 15$$

## 'Adresse anzeigen

```
Portd = Adrlo Or &HF0
```

Waitms 300

Portd = 0 Or &HF0

Waitms 200

Readeeprom Eebyte , Addr

Dat = Eebyte And 15

$$\text{Kom} = \text{Eebyte} / 16$$

Portd = Kom Or &amp;HF0

' Befehl anzeigen

**Do**

Loop Until S2 = 1

Waitms 50

**Prog = 1**

## 'Phase 1: Befehl anzeigen

Do

**If S1 = 0 Then**

If Prog = 1 Then

Prog = 2

**Kom = 15**

End If

If Prog = 2 Then

## 'Phase 2: Befehl verändert

$$\text{Kom} = \text{Kom} + 1$$

Kom = Kom And 15

```
Portd = Kom Or &HF0
```

End If

### 'Phase 3: Befehl unverändert, Daten ändern

Prog = 5

**Dat = 15**

End If

**If Prog = 4 Then**

### 'Phase 4: Befehl und Daten geändert

Proq = 5

**Dat = 15**

End If

**If Prog = 5 Then**

'Phase 5: Daten verändert

**Dat = Dat + 1**

```

    Dat = Dat And 15
    Portd = Dat Or &HF0
End If
Waitms 50
Do
    Loop Until S1 = 1
    Waitms 50
End If

If S2 = 0 Then
    If Prog = 3 Then Prog = 7
                                'nur angezeigt, nicht verändert

    If Prog = 1 Then
        Portd = Dat Or &HF0
        Prog = 3
    End If

    If Prog = 4 Then
        Portd = 255 - Dat
        Prog = 6
    End If

    If Prog = 2 Then
        Portd = Dat Or &HF0
        Prog = 4
    End If

    If Prog = 6 Then
                                'nur Kommando wurde verändert

        Dat = Dat And 15
        Eebyte = Kom * 16
        Eebyte = Eebyte + Dat
        Writeeprom Eebyte , Addr
        Portd = 255 - 0
        Waitms 600
        Addr = Addr + 1
        Prog = 0
    End If

    If Prog = 5 Then
                                'Daten wurden verändert

        Dat = Dat And 15
        Eebyte = Kom * 16
        Eebyte = Eebyte + Dat
        Writeeprom Eebyte , Addr
        Portd = 0 Or &HF0
        Waitms 600
        Addr = Addr + 1
        Prog = 0
    End If

    If Prog = 7 Then
        Addr = Addr + 1
        Prog = 0
    End If

```

```

    End If
    Waitms 50
    Do
    Loop Until S2 = 1
    Waitms 50
    End If
    Loop Until Prog = 0
Loop

```

End

---

Korrektur zum Port-Lesen, von Manfred Tischer

Im Programm muss man bei Verarbeitung des Befehls A=Din beachten, dass ja nur das untere Nibble des Bytes genutzt wird. Die beiden AD-Eingänge sind ja auch auf PortC (Teile des oberen Nibble). Von daher müssen beim Verarbeiten dieses Befehls die oberen Bits ausgeblendet werden. Ansonsten funktionieren anschließende Vergleiche ggf. nicht korrekt.

z.B.:

A=Din

B=A

A=1110

A=B? (Ergebnis des Vergleichs hängt vom Eingangssignal an den Analogeingängen ab)

Mit folgender Änderung für das Kommando 6 ist das Problem aus meiner Sicht behoben.

```

If Kom = 6 Then
    If Dat = 1 Then A = B
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = &H0F And Pinc
    If Dat = 5 Then A = Pinc.0
    If Dat = 6 Then A = Pinc.1
    If Dat = 7 Then A = Pinc.2
    If Dat = 8 Then A = Pinc.3
    If Dat = 9 Then
        Dd = Getadc(4)
        Dd = Dd / 64
        A = Dd
    End If
    If Dat = 10 Then
        Dd = Getadc(5)
        Dd = Dd / 64
        A = Dd
    End If

```

**End If**

Download der korrigierten Version, kompiliert mit Bascom 1.11.9.8:

[Tpsm8\\_n3.zip](#)

(Hinweis von B.K: Daraufhin habe ich noch mal in den Quelltext der Holtek-Version geschaut, dort war es schon genauso wie jetzt hier koorigiert:

In Holtek-C: if (Dat == 4) A = \_pa & 15;)

Translation into English by Juergen Pintaske

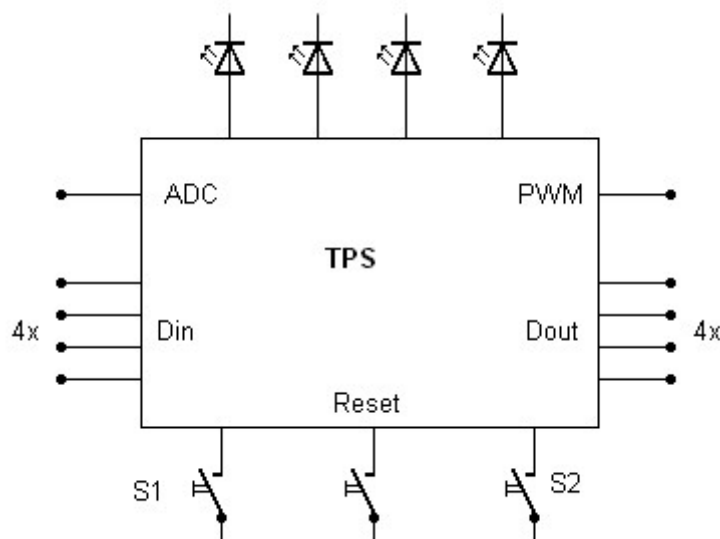
# The Pushbutton Programmable Controller (TPS 1 in German)

<http://www.elektronik-labor.de/Projekte/TPS1.html>

BASCOM <https://avrhelp.mcselec.com/>

[Elektronik-Labor](#) [Projekte](#) [TPS](#)

---



[TPS 2: The Programming Mode](#)

[Link in this document](#)

[TPS 3: Calculating with Variables](#)

[Link in this document](#)

[TPS 4: Sprünge und Verzweigungen](#)

[Link in this document](#)

[Mini-TPS mit ATmega8](#)

[Link in this document](#)

**The basic idea is simple:** How to program a small computer control system without the need for a PC or any programming device - using just a few pushbuttons.

The command set should be so simple, that you can keep it in your head, in order to develop a small control program without any documents if necessary. As everything should be as small and manageable as possible, it should be a 4-bit system.

There are

**4 digital outputs,**

**4 digital inputs,**

internally processed data are 4 bit wide,



and the commands are only coded with 4 bits,  
which means, that there is a maximum of **16 commands** you have to remember.

When entering programs, you have a display via the four OUTPUT LEDs,  
which will alternately display commands and related data.

**Two keys** are required for input:

**S1** is used to **enter data**,

**S2** for **programming**.

There must also be a **reset button** with which you can use  
either **start the program** or **switch to the programming mode**.

This small computer can perform a wide variety of tasks, from alarm systems to automatic battery charging stations or solar controllers. The idea developed while on vacation. I didn't have a PC with me, but realized, that such a small system would fit into every travel bag.

You could type in programs and train your mind in the process. Push, push, push, and a little reaction game is programmed.

Or you could have a programming contest, to find out  
who solves a task with the fewest program steps ...

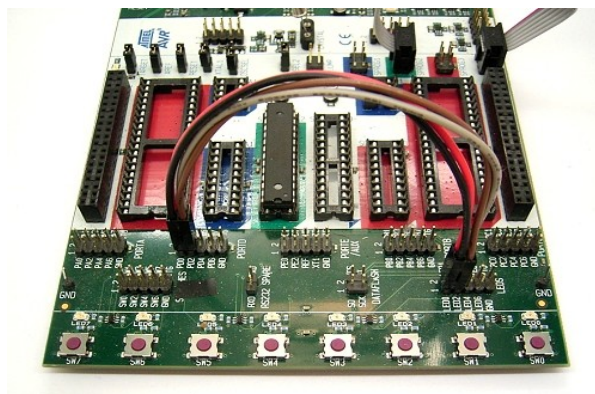
So far there has only been an idea and a rough list of the most important commands. I had developed something similar some time ago:

The *environmental spy* with several sensors and a simple interpreter language (spy basic).

Later then came the Kosmos Microcontroller with its Kosmos Basic. Both were 8-bit systems, that had to be programmed via the PC. This time I want to keep it even simpler, so only four bits and direct program input.

The whole development process should be presented here in all phases.

This means that everyone can then add their own commands and adapt the system to their own needs.



What is the best way to start?

Of course, you could first take a controller, place it on a breadboard and connect it to the three necessary buttons and four LEDs.

But at this moment it's not even clear which controller should be used in the end. That's why I decided to develop the first steps on the STK500.

Coincidentally, an ATmega168 was just in the socket (see picture) , which is now being used for the time being.

And the buttons and LEDs on the STK500 should be used. Two cables connect port D to the LEDs. D0 to D4 are used. As the LEDs in this system are switched to VCC, the outputs must be inverted during this development phase.

The firmware will be developed in Bascom.

During the first step, the interpreter should be started with only three commands (set port output, set waiting time, jump command).

The programming mode does not yet exist, but the first mini-program is written directly into the EEPROM with just five program steps.

The first three commands are:

1: **Direct Port Output** of the 4 bits: 0 ... 15

2: **Waiting Time** defined via 4 bits: 0 ... 15, 1 ms to 1 minute

(1ms, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 30000, 60000 ms)

3: **Jump Back** 0 ... 15 addresses

Each of these commands has 4 bit data entered directly.

Together with the data, one byte is occupied per instruction.

In hexadecimal notation, the upper nibble therefore represents the command and the lower nibble represents the data.

The first test program generates a simple alternating flasher:

Set 4 LEDs to 0001, Delay a sec, LEDs to 1000, Delay a sec, Jump Back to start

<b>&amp;H 1 1</b>	'Port output	<b>1</b>	<b>0001 0001</b>
<b>&amp;H 2 9</b>	'Waiting time	<b>1 sec</b>	<b>0010 1001</b>
<b>&amp;H 1 8</b>	'Port output	<b>8</b>	<b>0001 1000</b>
<b>&amp;H 2 9</b>	'Waiting time	<b>1 sec</b>	<b>0010 1001</b>
<b>&amp;H 3 4</b>	'Jump back	<b>4</b>	<b>0011 0100</b>

Download: [TPS1](#)

```

'-----
' Push Button Programmable Controller TPS
' Test 1: Interpreter, the first 3 instructions
'-----

$regfile    = "m168def.dat"      '
$crystal    = 11059200           '
$hwstack    = 32                 '
$swstack    = 64                 '
$framesize  = 64                 '

Dim Addr    As Byte              '
Dim Eebyte  As Byte              '
Dim Dat     As Byte              '
Dim Kom     As Byte              '

Ddrd  = &HFF                      'D0...D1 Outputs
Portd = &H0F                      'STK500 inverted
Portc = &H0F                      'C0..C3 Inputs with Pullup

S1 Alias Pinc.3                  'Data Input Button
S2 Alias Pinc.0                  'Programming Button

Waitms 200                        '

Dat = &H11 : Writeeprom Dat , 0      'Dout=1
Dat = &H29 : Writeeprom Dat , 1      '1000 ms
Dat = &H18 : Writeeprom Dat , 2      'Dout=8
Dat = &H29 : Writeeprom Dat , 3      '1000 ms
Dat = &H34 : Writeeprom Dat , 4      'Addr = Addr - 4

Waitms 200

If S2 = 0 Then
    Goto Programmieren             ' go to Programming
Else
    Ausfuehren:                    ' Execute
        Addr = 0
        Do
            Readeeprom Eebyte , Addr
            Addr = Addr + 1
            Dat = Eebyte And 15
            Kom = Eebyte / 16
            If Kom = 1 Then          '1: Direct Port Output
                Portd = 255 - Dat    'inverted Port Output as of STK500
            End If
            If Kom = 2 Then          '2: Waiting time, 1ms to 1 min

```

```

    If Dat = 0 Then Waitms 1      '
    If Dat = 1 Then Waitms 2      '
    If Dat = 2 Then Waitms 5      '
    If Dat = 3 Then Waitms 10     '
    If Dat = 4 Then Waitms 20     '
    If Dat = 5 Then Waitms 50     '
    If Dat = 6 Then Waitms 100    '
    If Dat = 7 Then Waitms 200    '
    If Dat = 8 Then Waitms 500    '
    If Dat = 9 Then Waitms 1000   '
    If Dat = 10 Then Waitms 2000  '
    If Dat = 11 Then Waitms 5000  '
    If Dat = 12 Then Waitms 10000 '
    If Dat = 13 Then Waitms 20000 '
    If Dat = 14 Then Waitms 30000 '
    If Dat = 15 Then Waitms 60000 '
End If
If Kom = 3 Then                '3: Jump - relative backwards
    Addr = Addr - 1            '
    Addr = Addr - Dat          '
End If
Loop
End If

Programmieren:                ' Programming
    Do
    Loop

End

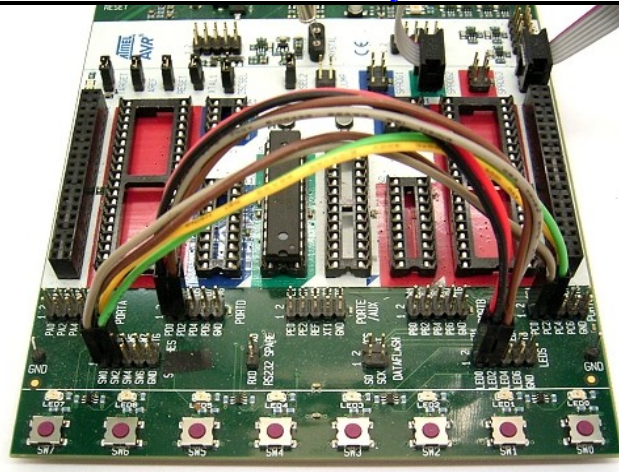
```

Indeed, the LEDs are flashing. Our mini-interpreter is ready with its just three commands. With this system, you can already write many different programs, from running lights to stepper motor control.

[Continue](#)

## TPS 2: The Programming Mode

[Elektronik-Labor Projekte TPS](#)



In order to be able to program the controller via the buttons, two more cables are placed on the STK. The keys **S1 (C3)** and **S2 (C0)** are used for programming. The position of the keys then corresponds to the original design:

**Data Entry** on the left,

**Programming** on the right.

The Programming Mode can be started by resetting while holding down the Programming Button S2. You can now view the entire program by just using the S2 button.

Each address requires two key presses of S2. In this way you change from the display of the command to the display of the data.

In addition, the current address is displayed for a short time.

- **First press of button S2:**
  - Displays the address (the lower four bits), for 300 ms
  - Display is off, for 300 ms
  - Show the command
- **Second press of button S2**
  - View data
- **Third press of button S2**
  - Show next address, 300 ms
  - etc.

For example, if you only want to view an existing program with five steps, but not change it, you can get to the end with a total of ten pushes of S2.

Orientation is easy, because the current address is briefly displayed.

You always know if the display is showing a command or data.

The **S1 button** is only used, if you want to change a command or your data. Basically only numerical values between zero and 15 can be entered.

The first time you press S1, a zero is set. Each subsequent keystroke increases the number by one. The current status is displayed in binary form via the four LEDs.

For example, if you want to enter a four, you press S1 for a total of five times: 0, 1, 2, 3, 4

If either the command or the data or both have been re-entered in this way, pressing S2 a second time causes this byte to be programmed into the EEPROM. To make this clear that programming is in progress, the LED display is switched off for 600 ms before the next address and then the next command is displayed.

This small pause should be understood intuitively as the programming process. In the back of your mind you can build the idea, that the system saves the energy for the display and uses it for programming the EEPROM.

You already know something like this from cars: When the starter is operated, the lights and the radio go out for a short moment.

This helps a lot if you only want to change an existing program in one place.

With S2 you scroll to the desired position and change the command or the data with S1, in order to then save them with S2.

If you want a program with e.g. the two bytes (& H17, & H30) to address 0 and 1 completely re-enter, then you have to type like this:

1. **Reset + S2** to get into the Programming Mode
2. **2 x S1**, the first one to get to 0, the second one to get to 1
3. **1 x S2** to program the first nibble of the first instruction into the EEPROM
4. **8 x S1**, the first one to get to 0, and then another 7 to get to 7
5. **1 x S2** to program the second nibble of the first instruction

The program counter is automatically incremented to address 1

6. **4 x S1**, the first one to get to 0, and then another 3 to get to 3
7. **1 x S2** to program the first nibble of the second instruction
8. **1 x S1** the first one to get to 0, no more presses needed
9. **1 x S2** to program the second nibble of the second instruction

If you've pushed too often, you have to go all over again. You know this from digital watches. However, one time around is not 60 as there, but in our case only 16.

By the way, a program with only two bytes, what could that be? If you look closely at the commands and data, it does this: switch on three of the 4 LEDs, and then end in the form of an endless loop with a jump 0 to the same address.

Download: [TPS2](#)

<b>Programmieren:</b>	<b>' Programming</b>
Addr = 0	'
Prog = 0	'
Do	
Adrlo = Addr And 15	'Display Address

```

Portd = 255 - Addr      '
Waitms 300              '
Portd = 255 - 0          '
Waitms 200              '
Readeeprom Eebyte , Addr '
Dat = Eebyte And 15      '
Kom = Eebyte / 16        '
Portd = 255 - Kom        'Display Instruction
Do
Loop Until S2 = 1        '
Waitms 50                '

Prog = 1                  'Phase 1: Display Instruction
Do
  If S1 = 0 Then          '

    If Prog = 1 Then      '
      Prog = 2            '
      Kom = 15            '
    End If

    If Prog = 2 Then      'Phase 2: Instruction changed
      Kom = Kom + 1      '
      Kom = Kom And 15   '
      Portd = 255 - Kom  '
    End If

    If Prog = 3 Then :    'Phase 3: Instruction unchanged,
                           '
                           '      change Data

      Prog = 5            '
      Dat = 15            '
    End If

                           'Phase 4: Instruction and
                           '      Data changed

      Prog = 5            '
      Dat = 15            '
    End If

    If Prog = 5 Then      'Phase 5: Data changed
      Dat = Dat + 1      '
      Dat = Dat And 15   '
      Portd = 255 - Dat  '
    End If
    Waitms 50            '
  Do

```

```

    Loop Until S1 = 1      '
    Waitms 50              '
End If

If S2 = 0 Then
    If Prog = 3 Then Prog = 7 'only displayed, not changed
    If Prog = 1 Then      '
        Portd = 255 - Dat '
        Prog = 3          '
    End If

    If Prog = 4 Then      '
        Portd = 255 - Dat '
        Prog = 6          '
    End If

    If Prog = 2 Then      '
        Portd = 255 - Dat '
        Prog = 4          '
    End If

    If Prog = 6 Then      ' Only Instruction been changed
        Dat = Dat And 15  '
        Eebyte = Kom * 16 '
        Eebyte = Eebyte + Dat '
        Writeeprom Eebyte , Addr '
        Portd = 255 - 0   '
        Waitms 600        '
        Addr = Addr + 1   '
        Prog = 0          '
    End If

    If Prog = 5 Then      'Data has been changed
        Dat = Dat And 15  '
        Eebyte = Kom * 16 '
        Eebyte = Eebyte + Dat '
        Writeeprom Eebyte , Addr '
        Portd = 255 - 0   '
        Waitms 600        '
        Addr = Addr + 1   '
        Prog = 0          '
    End If

    If Prog = 7 Then      '
        Addr = Addr + 1   '
        Prog = 0          '
    End If
    Waitms 50             '

```



```

        Do                                '
        Loop Until S2 = 1                 '
        Waitms 50                         '
    End If                                '
    Loop Until Prog = 0                   '
Loop

```

End

This Bascom program works like a State Machine in Programming Mode. When entering, it runs through several phases Prog = 0 to Prog = 7, depending on whether the memory content is only to be displayed or also to be changed. The keys are debounced with a waiting time of 50 ms.

And here are some programs you can enter and run at this stage:

### Running light 1:

& H11, & H28, & H12, & H28, & H14, & H28, & H18, & H28, & H38

### Running light 2:

& H11, & H28, & H12, & H28, & H14, & H28, & H18, & H28, & H14, & H28,  
& H12, & H28, & H3C

### Timer, one minute:

& H1F, & H2F, & H10, & H30

In the beginning you might write down the programs with comments, but after a while you learn to think and feel digitally.

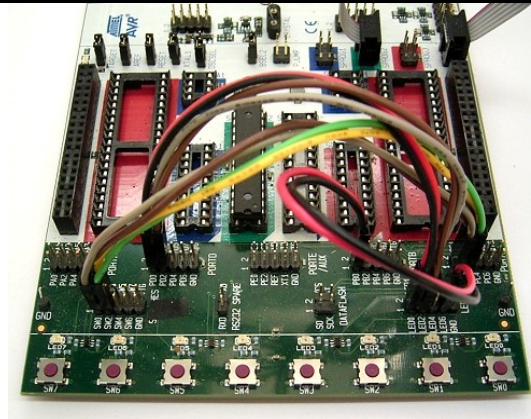
Man and machine become one unit, just like riding a motorcycle.

[back](#)

[next](#)

## TPS 3: Calculating using Variables

[Elektronik-Labor Projekte TPS](#)



The switch programmable controller will now be expanded with the four variables A, B, C and D. There is now also an analog input and a PWM output. Both are limited to 4 bits and can only be accessed via variable A ( $A \leq \text{ADC}$ ,  $\text{PWM} \leq A$ ).

A can also be loaded directly with a number.

To fill B, C or D, you must first load A and then assign the content to the other variable.

A and B can be used to perform some calculation steps.

C and D can serve as intermediate storage, and are used later as counters for counting loops.

**4:  $A \leq 0 \dots 15$**

**5: Target Function  $1 \dots 9 \leq A$**

(Register B, C, D,  
Dout (4 LEDs), or single Bits Dout.0, Dout.1, Dout.2, Dout.3,  
PWM as quasi-analog Output

**6:  $A \leq \text{Source } 1 \dots 9$**

(Register B, C, D,  
Din ( 4 Inputs), or single input bits Din.0, Din.1, Din.2, Din.3,  
ADC – analog input)

**7:  $A \leq \text{Expression } 1 \dots 10$**

( $A + 1$ ,  $A = A - 1$ ,  $A + B$ ,  $A = A - B$ ,  $A * B$ ,  $A / B$ , ( arithmetic function)  
 $A \text{ And } B$ ,  $A \text{ Or } B$ ,  $A \text{ Xor } B$ ,  $\text{Not } A$ ) ( logic function )

The following examples are commented out and at the beginning of the source code. You can enter the programs using the keys anyway,

or (as long as the controller is still on the STK) remove the comment characters and transfer the respective program to the EEPROM via Bascom. This time-saving method has proven very helpful and time saving during firmware development.

'Binary Counter:

```
'Dat = &H71 : Writeeprom Dat , 0      'A = A + 1
'Dat = &H54 : Writeeprom Dat , 1      'Dout = A
'Dat = &H29 : Writeeprom Dat , 2      '1000 ms
'Dat = &H33 : Writeeprom Dat , 3      'Addr = Addr - 3
```

'Analog to Digital Converter and Output

```
'Dat = &H69 : Writeeprom Dat , 0      'A = ADC
'Dat = &H54 : Writeeprom Dat , 1      'Dout = A
'Dat = &H29 : Writeeprom Dat , 2      '1000 ms
'Dat = &H33 : Writeeprom Dat , 3      'Addr = Addr - 3
```

'Analog to Digital Converter and PWM Output

```
'Dat = &H69 : Writeeprom Dat , 0      'A = ADC
'Dat = &H59 : Writeeprom Dat , 1      'PWM =A
'Dat = &H54 : Writeeprom Dat , 2      'Dout = A
'Dat = &H29 : Writeeprom Dat , 3      '1000 ms
'Dat = &H34 : Writeeprom Dat , 4      'Addr = Addr - 3
```

Download: [TPS3](#)

```
'-----
' Pushbutton Controlled System TPS
' Test 3: Variables and Calculation Instructions
'-----
```

```
Dim A As Byte      '
Dim B As Byte      '
Dim C As Byte      '
Dim D As Byte      '
```

```
Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
```

```
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare A Pwm =
Clear Down , Compare B Pwm = Clear Down
Start Timer1
```

...

```
Ausfuehren:      ' Execution
Addr = 0         '
```

```

Do
    Readeeprom Eebyte , Addr '
    Addr = Addr + 1 '
    Dat = Eebyte And 15 '
    Kom = Eebyte / 16 '
    If Kom = 1 Then '1: Direct Port Output
        Portd = 255 - Dat
        'inverted Port Output as of STK500
    End If
    If Kom = 2 Then '2: Waiting Time
        If Dat = 0 Then Waitms 1 '
        If Dat = 1 Then Waitms 2 '
        If Dat = 2 Then Waitms 5 '
        If Dat = 3 Then Waitms 10 '
        If Dat = 4 Then Waitms 20 '
        If Dat = 5 Then Waitms 50 '
        If Dat = 6 Then Waitms 100 '
        If Dat = 7 Then Waitms 200 '
        If Dat = 8 Then Waitms 500 '
        If Dat = 9 Then Waitms 1000 '
        If Dat = 10 Then Waitms 2000 '
        If Dat = 11 Then Waitms 5000 '
        If Dat = 12 Then Waitms 10000 '
        If Dat = 13 Then Waitms 20000 '
        If Dat = 14 Then Waitms 30000 '
        If Dat = 15 Then Waitms 60000 '
    End If
    If Kom = 3 Then '3: Jump Back Relative
        Addr = Addr - 1 '
        Addr = Addr - Dat '
    End If
    If Kom = 4 Then '
        A = Dat '
    End If
    If Kom = 5 Then '
        If Dat = 1 Then B = A 'Variables
        If Dat = 2 Then C = A '
        If Dat = 3 Then D = A '
        If Dat = 4 Then Portd = 255 - A 'Port
        If Dat = 5 Then
            If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1
            'Portbits
        End If
        If Dat = 6 Then
            If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1

```

```

End If
If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If
If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If
If Dat = 9 Then
    Dd = A * 17                                ' PWM
    Pwm1a = Dd                                ' PWM
End If
End If

If Kom = 6 Then
    If Dat = 1 Then A = B                      'Variables
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinb                    'Port
    If Dat = 5 Then A = Portb.0                'Portbits
    If Dat = 6 Then A = Portb.1
    If Dat = 7 Then A = Portb.2
    If Dat = 8 Then A = Portb.3
    If Dat = 9 Then
        Dd = Getadc(4)                        'ADC
        Dd = Dd / 64
        A = Dd
    End If
End If

If Kom = 7 Then
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B
    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B
    If Dat = 9 Then A = A Xor B
    If Dat = 10 Then A = Not A
    A = A And 15
End If

    Loop
End If

```

[back](#)  
[next](#)

# TPS 4: Jumps and Branches

[Elektronik-Labor Projekte TPS](#)

```

BASCOS-AVR IDE - [C:\Arbeit\Elektronik-Labor-Material\tps\TPS4\TPS4.bas]
File Edit Program Tools Options Window Help
Sub Label
If Kom = 8 Then
  Adrhi = Dat
End If
If Kom = 9 Then
  Adr = Adrhi * 16
  Adr = Adr + Dat
End If
If Kom = 10 Then
  C = C - 1
  C = C And 15
  If C > 0 Then
    Adr = Adrhi * 16
    Adr = Adr + Dat
  End If
End If
If Kom = 11 Then
  D = D + 1
End If

```

253: 34 Insert

Until now there has only been a simple **Jump Back relative** (command 3), which only covered a maximum of 15 addresses.

Now there will be an **Absolute Jump**.

Since the jump destination can only be specified with 4 bits, there is an additional command that defines the **4 Bit PAGE**, i.e. the high nibble of the address. This gives us an address space of 0 ... 255.

This should be sufficient for most of our applications.

Two **Counting Loops** with the variables **C** and **D** can also execute absolute jumps if not 0 after execution, whereby the high nibble of the address can be loaded before.

The **Conditional Jumps** are based on the skip commands as in AVR assembler. If the respective condition is true, the following address is skipped. At this location there could be a jump command or a subroutine call, for example. Comparisons between A and B, as well as direct bit queries of the input port bits are available as conditions.

There is also a **Subroutine Call** and the associated **Return Command**.

Several subprograms are allowed, but no further subprogram may be called from a subprogram ( no nesting ), because the interpreter only remembers the one return address of the calling main program.

- 8: Set Adr-high** = 0...15 set the PAGE register  
**9: Direct Jump** to Adr-high, Adr-low (0..15)  
**10: Counting Loop C-times** Adr-high, Adr-low (0..15)  
**11: Counting Loop D-times** Adr-high, Adr-low (0..15)  
**12: Conditional Jumps (Skips):** If (Bedingung 1...11) then Adr = Adr + 1  
     (A > B,      A > B,      A = B,  
     Din.0 = 1, Din.1 = 1, Din.2 = 1, Din.3 = 1,  
     Din.0 = 0, Din.1 = 0, Din.2 = 0, Din.3 = 0)  
**13: Subroutine Call** Adr-high, Adr-low (0..15)  
**14: Return from Subroutine**

With these additions, our programming language is now complete. However, extensions are possible, because two possible commands (0 and 15) have not yet been defined. And some commands such as command 12 do not yet exhaust the possible number of their parameters.

For example, the **Conditional Jump** command (12) could have a few more conditions.

The same applies to commands 5, 6 and 7.

You could define further calculation steps, more AD channels and a second PWM output.

### Examples:

#### 'Test: Counter Loop

```

'Dat = &H45 : Writeeprom Dat , 0           'A=5
'Dat = &H52 : Writeeprom Dat , 1           'C=A
'Dat = &H11 : Writeeprom Dat , 2           'Port=1
'Dat = &H27 : Writeeprom Dat , 3           '200 ms
'Dat = &H10 : Writeeprom Dat , 4           'Port=0
'Dat = &H27 : Writeeprom Dat , 5           '200 ms
'Dat = &HA2 : Writeeprom Dat , 6           'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 7           'Ende

```

#### 'Test: Subroutine

```

'Dat = &H4F : Writeeprom Dat , 0           'A=15
'Dat = &H52 : Writeeprom Dat , 1           'C=A
'Dat = &H82 : Writeeprom Dat , 2           'Adr 32...
'Dat = &HD0 : Writeeprom Dat , 3           'Call 32+0
'Dat = &H80 : Writeeprom Dat , 4           'Adr 00...
'Dat = &HA2 : Writeeprom Dat , 5           'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 6           'Ende

```

```
'Subroutine from address 32
'Dat = &H11 : Writeeprom Dat , 32      'Port=1
'Dat = &H27 : Writeeprom Dat , 33      '200 ms
'Dat = &H10 : Writeeprom Dat , 34      'Port=0
'Dat = &H27 : Writeeprom Dat , 35      '200 ms
'Dat = &H27 : Writeeprom Dat , 35      '200 ms
'Dat = &HE0 : Writeeprom Dat , 36      'Return
```

```
'Test: Branch, flash if ADC < 7
Dat = &H47 : Writeeprom Dat , 0      'A=7
Dat = &H51 : Writeeprom Dat , 1      'B=A
Dat = &H69 : Writeeprom Dat , 2      'A=ADC
Dat = &H82 : Writeeprom Dat , 3      'Adr 32...
Dat = &HC2 : Writeeprom Dat , 4      'if A<B then Adr=Adr+1
Dat = &HD0 : Writeeprom Dat , 5      'Call 32+0
Dat = &H80 : Writeeprom Dat , 6      'Adr 00...
Dat = &H90 : Writeeprom Dat , 7      'Jump 00
```

```
'Subroutine from address 32
Dat = &H11 : Writeeprom Dat , 32      'Port=1
Dat = &H27 : Writeeprom Dat , 33      '200 ms
Dat = &H10 : Writeeprom Dat , 34      'Port=0
Dat = &H27 : Writeeprom Dat , 35      '200 ms
Dat = &H27 : Writeeprom Dat , 35      '200 ms
Dat = &HE0 : Writeeprom Dat , 36      'Return
```

Download: [TPS4](#)

```
'-----
' Pushbutton Programmable Controller TPS
' Test 4: Jumps and Branches
'-----
```

```
$regfile = "m168def.dat"      '
$crystal = 11059200          '
$hwstack = 32                '
$swstack = 64                '
$framesize = 64              '
```

```
Dim Addr    As Byte          '
Dim Eebyte  As Byte          '
Dim Dat     As Byte          '
Dim Kom     As Byte          '
Dim Adrhi   As Byte          '
Dim Adrlo   As Byte          '
Dim Adrret  As Byte          '
```



```

Dim Prog    As Byte
Dim Dd      As Word

Waitms 200

If S2 = 0 Then
    Goto Programmieren
Else
    Ausfuehren:
        Addr = 0
        Do
            Readeeprom Eebyte , Addr
            Addr = Addr + 1
            Dat = Eebyte And 15
            Kom = Eebyte / 16
        ...
        If Kom = 8 Then
            Adrhi = Dat
            End If
            If Kom = 9 Then
                Addr = Adrhi * 16
                Addr = Addr + Dat
            End If
            If Kom = 10 Then
                C = C - 1
                C = C And 15
                If C > 0 Then
                    Addr = Adrhi * 16
                    Addr = Addr + Dat
                End If
            End If
            If Kom = 11 Then
                D = D - 1
                D = D And 15
                If D > 0 Then
                    Addr = Adrhi * 16
                    Addr = Addr + Dat
                End If
            End If
            If Kom = 12 Then
                If Dat = 1 Then
                    If A > B Then Addr = Addr + 1
                End If
                If Dat = 2 Then

```

'Oberes Nibble der Adresse

'Springe absolut 0...255

'C-mal

'Springe absolut 0...255

'D-mal

'Springe absolut 0...255

```

    If A < B Then Addr = Addr + 1
End If
If Dat = 3 Then
    If A = B Then Addr = Addr + 1
End If
If Dat = 4 Then
    If Pinc.0 = 1 Then Addr = Addr + 1
End If
If Dat = 5 Then
    If Pinc.1 = 1 Then Addr = Addr + 1
End If
If Dat = 6 Then
    If Pinc.2 = 1 Then Addr = Addr + 1
End If
If Dat = 7 Then
    If Pinc.3 = 1 Then Addr = Addr + 1
End If
If Dat = 8 Then
    If Pinc.0 = 0 Then Addr = Addr + 1
End If
If Dat = 9 Then
    If Pinc.1 = 0 Then Addr = Addr + 1
End If
If Dat = 10 Then
    If Pinc.2 = 0 Then Addr = Addr + 1
End If
If Dat = 11 Then
    If Pinc.3 = 0 Then Addr = Addr + 1
End If
End If

If Kom = 13 Then
    Addrret = Addr
    Addr = Adrhi * 16 'Call Unterprogramm absolut 0...255
    Addr = Addr + Dat
End If

If Kom = 14 Then
    Addr = Addrret
End If
Loop
End If

```

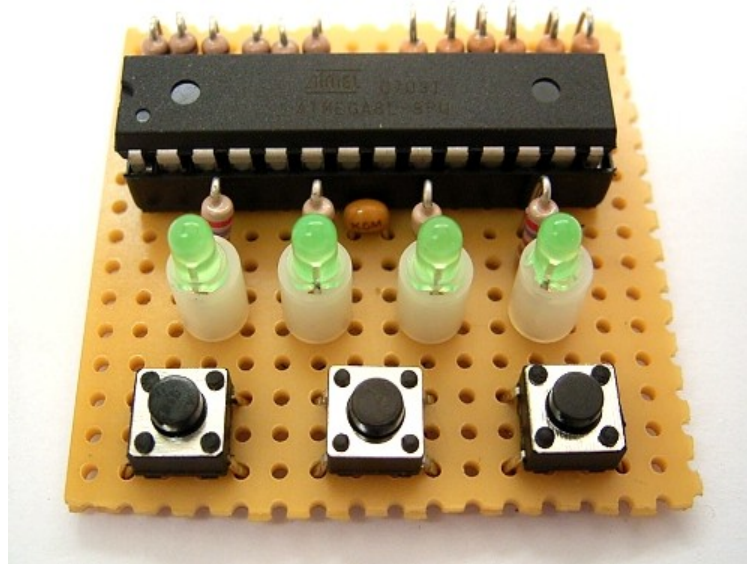
'Return

---

# Mini-TPS mit ATmega8

[Elektronik-Labor](#) [Projekte](#) [TPS](#)

---



Dies ist eine Miniversion mit dem ATmega8. Alle Ein- und Ausgänge liegen am oberen Rand und wurden mit Serienwiderständen gegen Überlastung geschützt, die Eingänge 2,2 k, die Ausgänge mit 1 k. Die Ausgänge liegen wie bisher auf D0 bis D3, die Eingänge wie bisher auf C0 bis C3.

Abweichend vom ersten Entwurf mit dem Mega168 gibt es diesmal zwei PWM-Ausgänge (PWM1, PWM2) und zwei AD-Eingänge (AD1= ADC4 und AD2 = ADC5). Die Taste S1 liegt jetzt an D6 und S2 an D7. Anders als beim STK500 werden die Portausgänge nicht mehr invertiert. Aber man muss darauf achten, dass die beiden Tasten am gleichen Port liegen und die Pullups eingeschaltet bleiben.

Da die Tasten jetzt nicht mehr am Eingangsport liegen, bietet sich eine direkte Tastenabfrage über die bedingten Sprungbefehle an. Deshalb gibt es jetzt vier neue Optionen für den Befehl 12 (CC bis CF).

## Die erweiterte Befehlsliste

**10 ... 1F: Direkte Portausgabe** 0...15

**20 ... 2F: Wartezeit** 0...15

(1, 2, 5, 10, 20, 50, 100, 200, 500,  
1000, 2000, 5000, 10000, 20000, 30000, 60000 ms)

**30 ... 3F: Sprung zurück** 0...15

**40 ... 4F: A = 0...15**

**51...5A: Ziel 1...10 = A**

51: B = A

52: C = A

53: D = A

54: Dout = A

55: Dout.0 = A.0

56: Dout.1 = A.0

57: Dout.2 = A.0

58: Dout.3 = A.0

59: PWM1 = A

5A: PWM2 = A

**61 ..6A: A = Quelle 1...10**

61: A = B

62: A = C

63: A = D

64: A = Din

65: A = Din.0

66: A = Din.1

67: A = Din.2

68: A = Din.3

69: A = ADC1

6A: A = ADC2

**71 ...7A: A = Ausdruck 1...10**

71: A = A + 1

72: A = A - 1

73: A = A + B

74: A = A - B

75: A = A \* B

76: A = A / B

77: A = A And B

78: A = A Or B

79: A = A Xor B

7A: A = Not A)

**80 ... 8F: Adr-high = 0...15**

**90 ... 0F: Direkter Sprung** auf Adr-high, Adr-low (0..15)

**A0 ... AF: Zählschleife C-mal** Adr-high, Adr-low (0..15)

**B0 ... BF: Zählschleife D-mal** Adr-high, Adr-low (0..15)

**C1 ... CF: Bedingter Sprung:** If (Bedingung 1...15) then Adr = Adr + 1

C1: if A > B then Adr = Adr + 1

C2: if A < B then Adr = Adr + 1

C3: if A = B then Adr = Adr + 1

C4: if Din.0 = 1 then Adr = Adr + 1

C5: if Din.1 = 1 then Adr = Adr + 1

C6: if Din.2 = 1 then Adr = Adr + 1

C7: if Din.3 = 1 then Adr = Adr + 1

C8: if Din.0 = 0 then Adr = Adr + 1

C9: if Din.1 = 0 then Adr = Adr + 1

CA: if Din.2 = 0 then Adr = Adr + 1

CB: if Din.3 = 0 then Adr = Adr + 1

CC: if S1 = 0 then Adr = Adr + 1

CD: if S2 = 0 then Adr = Adr + 1

CE: if S1 = 1 then Adr = Adr + 1

CF: if S2 = 1 then Adr = Adr + 1

**D0 ... DF Unterprogrammaufruf** Adr-high, Adr-low (0..15)

**E0 ... EF: Return**

**Ein Beispiel zur Tastenabfrage S1:** Solange man die Taste drückt, läuft ein schneller Zähler, die LEDs leuchten scheinbar mit halber Helligkeit. Lässt man die Taste los, friert der Zählerstand ein. Damit hat man praktisch ein Zufallsprogramm bzw. einen Würfel von Null bis 15.

'Dat = &HCC : Writeeprom Dat , 0

'Dat = &H71 : Writeeprom Dat , 1

'Dat = &H54 : Writeeprom Dat , 2

'Dat = &H33 : Writeeprom Dat , 3

'if S1 = 1 then Adr = Adr + 1

'A = A + 1

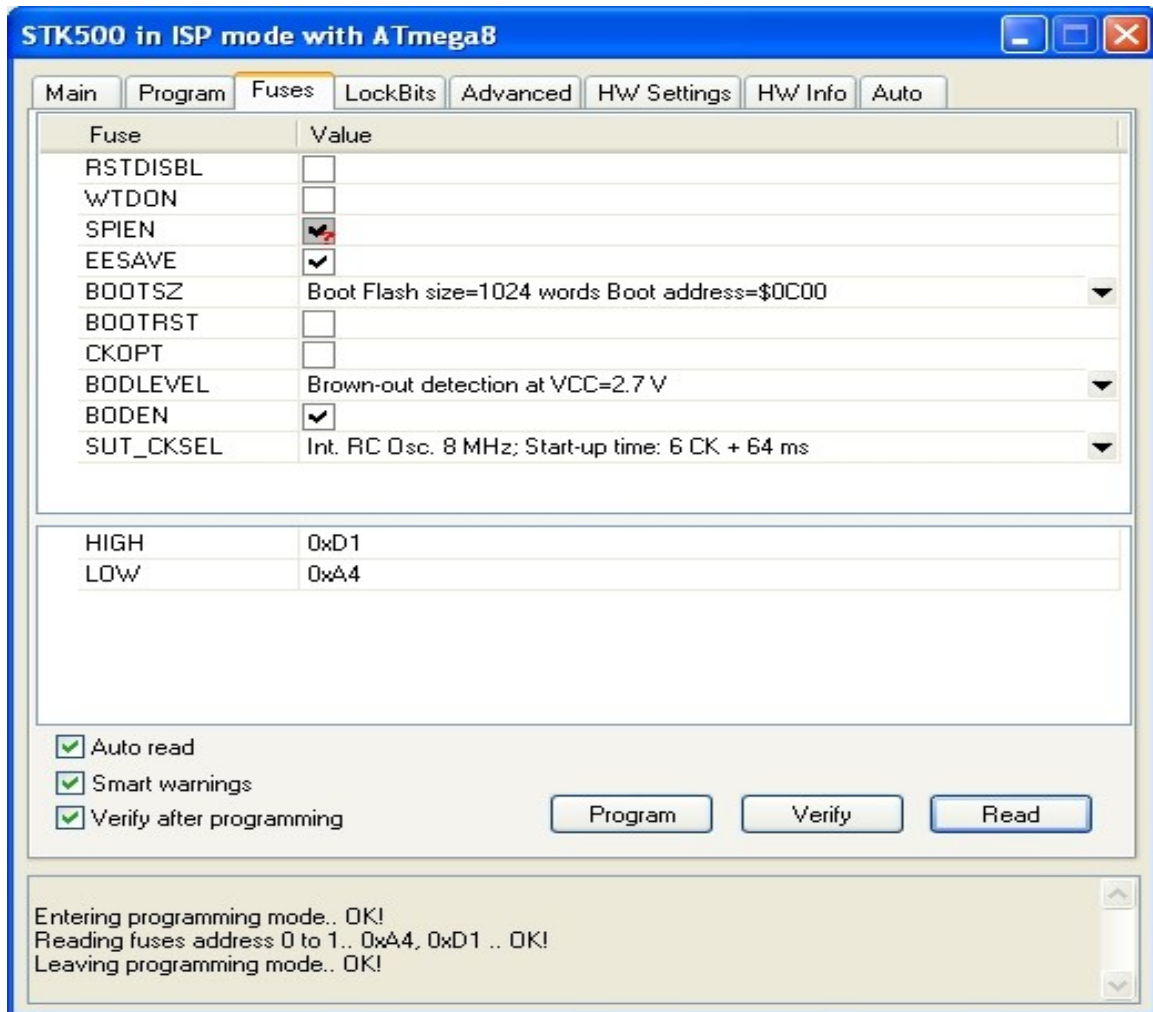
'Dout = A

'Adr = Adr - 3

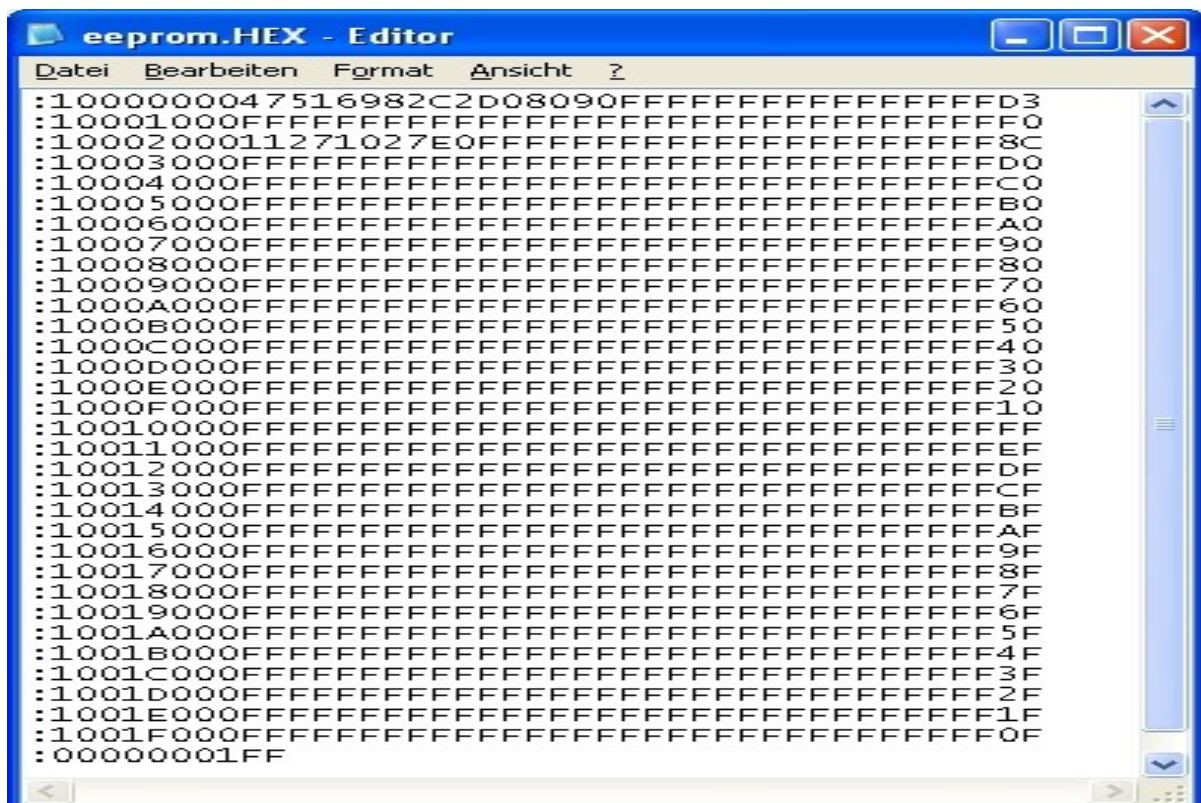
## Die Befehle in einer Programmierkarte

	1	2	3	4	5	6	7	8	9	A	B	C	D	E
<b>0</b>	Dout	1 ms	jmp -	A=				AdrHi	jmp	C*	D*	if ...	call	ret
<b>1</b>		2 ms			B=A	A= B	A = A + 1					A > B		
<b>2</b>		5 ms			C=A	A = C	A = A - 1					A < B		
<b>3</b>		10			D=A	A = D	A = A + B					A = B		
<b>4</b>		20			Dout = A	A = Din	A = A - B					Din.0 = 1		
<b>5</b>		50			Dout.0 = A.0	A = Din.0	A = A * B					Din.1 = 1		
<b>6</b>		100			Dout.1 = A.0	A = Din.1	A = A / B					Din.2 = 1		
<b>7</b>		200			Dout.2 = A.0	A = Din.2	A = A And B					Din.3 = 1		
<b>8</b>		500			Dout.3 = A.0	A = Din.3	A = A Or B					Din.0 = 0		
<b>9</b>		1 s			PWM1 = A	A = ADC1	A = A Xor B					Din.1 = 0		
<b>A</b>		2 s			PWM2 = A	A = ADC2	A = Not A					Din.2 = 0		
<b>B</b>		5 s										Din.3 = 0		
<b>C</b>		10										S1 = 0		
<b>D</b>		20										S2 = 0		
<b>E</b>		30										S1 = 1		
<b>F</b>		60										S2 = 1		

Der Controller läuft jetzt mit 8 MHz intern. EESAVE ist aktiviert. Das bedeutet, man kann beim ersten Brennen ein Programm im EEPROM mitgeben, es dann im Quelltext auskommentieren oder löschen und noch einmal brennen.



Insbesondere nützliche Unterprogramme lassen sich auf diese Weise für die spätere Verwendung bereithalten. Das Bild zeigt ein Programm ab 00 in ein Unterprogramm ab &H20, ausgelesen mit dem STK500.



Download: [TPSm8.zip](#)

Und hier das komplette Listing mit allen Anpassungen für den Mega8.

```

'-----
' Tasten-programmierbare Steuerung TPS
' ATmega8, intern 8 MHz, 2 ADC, 2 PWM
'-----

$regfile      = "m8def.dat"      '
$crystal      = 8000000          '
$hwstack      = 32               '
$swstack      = 64               '
$framesize    = 64              '


Dim ADR      As Byte            '
Dim Eebyte   As Byte            '
Dim Dat      As Byte            '
Dim Kom      As Byte            '
Dim ADRhi    As Byte            '
Dim ADRlo    As Byte            '
Dim ADRret   As Byte            '
Dim Prog     As Byte            '
Dim Dd       As Word            '


Dim A        As Byte            '
Dim B        As Byte            '
Dim C        As Byte            '
Dim D        As Byte            '


Ddrd  = &H0F                      'D0...D1 Outputs
Portd = &HF0                      'Pullup D4...D7
Portc = &H0F                      'C0..C3 Inputs mit Pullup


S1 Alias Pind.6                  'Dateneingabe
S2 Alias Pind.7                  'Programmieren


Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare A Pwm =
Clear Down , Compare B Pwm = Clear Down
Start Timer1


If S2 = 0 Then

```



Goto Programmieren

Else

Ausfuehren:

Adr = 0

Do

Readeeprom Eebyte , Adr

Adr = Adr + 1

Dat = Eebyte And 15

Kom = Eebyte / 16

If Kom = 1 Then

1: Direkte Portausgabe

Portd = Dat Or &HF0

End If

If Kom = 2 Then

'2: Wartezeit

If Dat = 0 Then Waitms 1

If Dat = 1 Then Waitms 2

If Dat = 2 Then Waitms 5

If Dat = 3 Then Waitms 10

If Dat = 4 Then Waitms 20

If Dat = 5 Then Waitms 50

If Dat = 6 Then Waitms 100

If Dat = 7 Then Waitms 200

If Dat = 8 Then Waitms 500

If Dat = 9 Then Waitms 1000

If Dat = 10 Then Waitms 2000

If Dat = 11 Then Waitms 5000

If Dat = 12 Then Waitms 10000

If Dat = 13 Then Waitms 20000

If Dat = 14 Then Waitms 30000

If Dat = 15 Then Waitms 60000

End If

If Kom = 3 Then

'3: Sprung - relativ

Adr = Adr - 1

Adr = Adr - Dat

End If

If Kom = 4 Then

A = Dat

End If

If Kom = 5 Then

If Dat = 1 Then B = A

'Variablen

If Dat = 2 Then C = A

If Dat = 3 Then D = A

If Dat = 4 Then Portd = A Or &HF0

'Port

If Dat = 5 Then

```

    If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1
                                                    'Portbits
End If
If Dat = 6 Then
    If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1
End If
If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If
If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If
If Dat = 9 Then
    Dd = A * 17
                                                    'PWM
    Pwmla = Dd
                                                    'PWM
End If
End If

If Kom = 6 Then
    If Dat = 1 Then A = B
                                                    'Variablen
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinb
                                                    'Port
    If Dat = 5 Then A = Portb.0
                                                    'Portbits
    If Dat = 6 Then A = Portb.1
    If Dat = 7 Then A = Portb.2
    If Dat = 8 Then A = Portb.3
    If Dat = 9 Then
        Dd = Getadc(4)
                                                    'ADC
        Dd = Dd / 64
        A = Dd
    End If
    If Dat = 10 Then
        Dd = Getadc(5)
                                                    'ADC
        Dd = Dd / 64
        A = Dd
    End If
End If

If Kom = 7 Then
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B
    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B

```

```

    If Dat = 9 Then A = A Xor B
    If Dat = 10 Then A = Not A
    A = A And 15
End If

If Kom = 8 Then
    Adrhi = Dat                                'Oberes Nibble der Adresse
End If

If Kom = 9 Then
    Adr = Adrhi * 16                            'Springe absolut 0...255
    Adr = Adr + Dat
End If

If Kom = 10 Then
    C = C - 1
    C = C And 15
    If C > 0 Then                                'C-mal
        Adr = Adrhi * 16                        'Springe absolut 0...255
        Adr = Adr + Dat
    End If
End If

If Kom = 11 Then
    D = D - 1
    D = D And 15
    If D > 0 Then                                'D-mal
        Adr = Adrhi * 16                        'Springe absolut 0...255
        Adr = Adr + Dat
    End If
End If

If Kom = 12 Then
    If Dat = 1 Then
        If A > B Then Adr = Adr + 1
    End If
    If Dat = 2 Then
        If A < B Then Adr = Adr + 1
    End If
    If Dat = 3 Then
        If A = B Then Adr = Adr + 1
    End If
    If Dat = 4 Then
        If Pinc.0 = 1 Then Adr = Adr + 1
    End If
    If Dat = 5 Then

```

```

    If Pinc.1 = 1 Then Adr = Adr + 1
End If
If Dat = 6 Then
    If Pinc.2 = 1 Then Adr = Adr + 1
End If
If Dat = 7 Then
    If Pinc.3 = 1 Then Adr = Adr + 1
End If
If Dat = 8 Then
    If Pinc.0 = 0 Then Adr = Adr + 1
End If
If Dat = 9 Then
    If Pinc.1 = 0 Then Adr = Adr + 1
End If
If Dat = 10 Then
    If Pinc.2 = 0 Then Adr = Adr + 1
End If
If Dat = 11 Then
    If Pinc.3 = 0 Then Adr = Adr + 1
End If
If Dat = 12 Then
    If Pind.6 = 1 Then Adr = Adr + 1
End If
If Dat = 13 Then
    If Pind.7 = 1 Then Adr = Adr + 1
End If
If Dat = 14 Then
    If Pind.6 = 0 Then Adr = Adr + 1
End If
If Dat = 15 Then
    If Pind.7 = 0 Then Adr = Adr + 1
End If

```

```
End If
```

```

If Kom = 13 Then
    Adrret = Adr
    Adr = Adrhi * 16    'Call Unterprogramm absolut  0...255
    Adr = Adr + Dat
End If

```

```

If Kom = 14 Then
    Adr = Adrret
End If

```

```

Loop
End If

```

# Programmieren:

```

Adr = 0
Prog = 0
Do
    Adrlo = Adr And 15                                'Adresse anzeigen
    Portd = Adr Or &HF0
    Waitms 300
    Portd = 0 Or &HF0
    Waitms 200
    Readeeprom Eebyte , Adr
    Dat = Eebyte And 15
    Kom = Eebyte / 16
    Portd = Kom Or &HF0                                'Befehl anzeigen
    Do
    Loop Until S2 = 1
    Waitms 50

    Prog = 1                                            'Phase 1: Befehl anzeigen
    Do
        If S1 = 0 Then

            If Prog = 1 Then
                Prog = 2
                Kom = 15
            End If
            If Prog = 2 Then                            'Phase 2: Befehl verändert
                Kom = Kom + 1
                Kom = Kom And 15
                Portd = Kom Or &HF0
            End If
            If Prog = 3 Then :
                'Phase 3: Befehl unverändert, Daten ändern
                Prog = 5
                Dat = 15
            End If
            If Prog = 4 Then                            'Phase 4: Befehl und Daten geändert
                Prog = 5
                Dat = 15
            End If
            If Prog = 5 Then                            'Phase 5: Daten verändert
                Dat = Dat + 1
                Dat = Dat And 15
                Portd = Dat Or &HF0
            End If
            Waitms 50
        Do
        Loop Until S1 = 1

```

```

    Waitms 50
End If

If S2 = 0 Then
    If Prog = 3 Then Prog = 7
                                'nur angezeigt, nicht verändert

    If Prog = 1 Then
        Portd = Dat Or &HF0
        Prog = 3
    End If
    If Prog = 4 Then
        Portd = 255 - Dat
        Prog = 6
    End If
    If Prog = 2 Then
        Portd = Dat Or &HF0
        Prog = 4
    End If
    If Prog = 6 Then
                                'nur Kommando wurde verändert
        Dat = Dat And 15
        Eebyte = Kom * 16
        Eebyte = Eebyte + Dat
        Writeeprom Eebyte , Adr
        Portd = 255 - 0
        Waitms 600
        Adr = Adr + 1
        Prog = 0
    End If
    If Prog = 5 Then
                                'Daten wurden verändert
        Dat = Dat And 15
        Eebyte = Kom * 16
        Eebyte = Eebyte + Dat
        Writeeprom Eebyte , Adr
        Portd = 0 Or &HF0
        Waitms 600
        Adr = Adr + 1
        Prog = 0
    End If
    If Prog = 7 Then
        Adr = Adr + 1
        Prog = 0
    End If
    Waitms 50
    Do
        Loop Until S2 = 1
    Waitms 50
End If
Loop Until Prog = 0

```

Loop

End

---

Korrektur der Firmware, von Michael Gaus

Müsste das statt Pinb nicht Pinc und statt Portb.0...Portb.3 nicht Pinc.0...Pinc.3 heißen, es sollen ja die Eingänge abgefragt werden?

```

    If Kom = 6 Then
        If Dat = 1 Then A = B
'Variablen
        If Dat = 2 Then A = C
        If Dat = 3 Then A = D
        If Dat = 4 Then A = Pinc
'Port
        If Dat = 5 Then A = Portc.0
'Portbits
        If Dat = 6 Then A = Portc.1
        If Dat = 7 Then A = Portc.2
        If Dat = 8 Then A = Portc.3

```

Antwort: Das stimmt wohl, die Eingaben müssen auf Port C geändert werden. Diese Version für den Mega8 wurde nicht so sorgfältig geprüft wie die Holtek-Version für die Lenpakete. Es würde mich nicht wundern, wenn noch mehr Fehler auftauchen.

---

Weitere Korrekturen, von Michael Gaus

Ich habe entsprechende Änderungen vorgenommen und mit der neuesten Bascom-AVR Version 2.0.7.5 Demo kompiliert. Dabei wurde angemeckert, dass "Adr" ein "reserved Word" sei, ich habe dann alle "Adr" ersetzt durch "Addr".

Änderungen:

- 1) PINB ersetzt durch PINC, PORTB.x ersetzt durch PINC.x , s.o.
- 2) Skip if S1/S2=0/1 waren alle verdreht abgefragt => korrigiert
- 3) Beim Label "Programmieren" muss es in der 2. Zeile nach dem "Do" Befehl heißen: PORTD = Adrlo OR &HF0 statt Addr
- 4) Bei PWM Config: clear\_down geändert in clear\_up
5. C- und D-Zählschleife am die Holtek-Version angeglichen

Down load: [Tpsm8\\_korrigiert\\_2.zip](#)

```
'-----
' Tasten-programmierbare Steuerung TPS
' ATmega8, intern 8 MHz, 2 ADC, 2 PWM
'-----

$regfile = "m8def.dat"
$crystal = 8000000
$hwstack = 32
$swstack = 64
$framesize = 64

Dim Addr As Byte
Dim Eebyte As Byte
Dim Dat As Byte
Dim Kom As Byte
Dim Adrhi As Byte
Dim Adrlo As Byte
Dim Adrret As Byte
Dim Prog As Byte
Dim Dd As Word

Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim D As Byte

Ddrd = &H0F                                'D0...D1 Outputs
Portd = &HF0                                'Pullup D4...D7
Portc = &H0F                                'C0..C3 Inputs mit Pullup

S1 Alias Pind.6                             'Dateneingabe
S2 Alias Pind.7                             'Programmieren

Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare_a_pwm =
Clear_up , Compare_b_pwm = Clear_up
Start Timer1

Waitms 200
```



## 'Test: Zählschleife

```

'Dat = &H45 : Writeeprom Dat , 0      'A=5
'Dat = &H52 : Writeeprom Dat , 1      'C=A
'Dat = &H11 : Writeeprom Dat , 2      'Por=1
'Dat = &H27 : Writeeprom Dat , 3      '200 ms
'Dat = &H10 : Writeeprom Dat , 4      'Port=0
'Dat = &H27 : Writeeprom Dat , 5      '200 ms
'Dat = &HA2 : Writeeprom Dat , 6      'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 7      'Ende

```

## 'Test: Unterprogramm

```

'Dat = &H4F : Writeeprom Dat , 0      'A=15
'Dat = &H52 : Writeeprom Dat , 1      'C=A
'Dat = &H82 : Writeeprom Dat , 2      'Adr 32...
'Dat = &HD0 : Writeeprom Dat , 3      'Call 32+0
'Dat = &H80 : Writeeprom Dat , 4      'Adr 00...
'Dat = &HA2 : Writeeprom Dat , 5      'C-mal jmp 2
'Dat = &H30 : Writeeprom Dat , 6      'Ende
'
'                               Unterprogramm ab Adresse 32
'Dat = &H11 : Writeeprom Dat , 32     'Port=1
'Dat = &H27 : Writeeprom Dat , 33     '200 ms
'Dat = &H10 : Writeeprom Dat , 34     'Port=0
'Dat = &H27 : Writeeprom Dat , 35     '200 ms
'Dat = &H27 : Writeeprom Dat , 35     '200 ms
'Dat = &HE0 : Writeeprom Dat , 36     'Return

```

## 'Test: Verzweigung, Blinken wenn ADC &lt; 7

```

'Dat = &H47 : Writeeprom Dat , 0      'A=7
'Dat = &H51 : Writeeprom Dat , 1      'B=A
'Dat = &H69 : Writeeprom Dat , 2      'A=ADC
'Dat = &H82 : Writeeprom Dat , 3      'Adr 32...
'Dat = &HC2 : Writeeprom Dat , 4      'if A<B then Adr=Adr+1
'Dat = &HD0 : Writeeprom Dat , 5      'Call 32+0
'Dat = &H80 : Writeeprom Dat , 6      'Adr 00...
'Dat = &H90 : Writeeprom Dat , 7      'Jump 00
'
'                               Unterprogramm ab Adresse 32
'Dat = &H11 : Writeeprom Dat , 32     'Port=1
'Dat = &H27 : Writeeprom Dat , 33     '200 ms
'Dat = &H10 : Writeeprom Dat , 34     'Port=0
'Dat = &H27 : Writeeprom Dat , 35     '200 ms
'Dat = &H27 : Writeeprom Dat , 35     '200 ms
'Dat = &HE0 : Writeeprom Dat , 36     'Return

```

## 'Test : Tastenabfrage S1

```

'Dat = &HCC : Writeeprom Dat , 0      'if S1 = 1 then Adr = Adr+1
'Dat = &H71 : Writeeprom Dat , 1      'A = A + 1
'Dat = &H54 : Writeeprom Dat , 2      'Dout =A

```

'Dat = &H33 : Writeeprom Dat , 3

'Adr = Adr - 3

Waitms 200

If S2 = 0 Then

Goto Programmieren

Else

Ausfuehren:

Addr = 0

Do

Readeeprom Eebyte , Addr

Addr = Addr + 1

Dat = Eebyte And 15

Kom = Eebyte / 16

If Kom = 1 Then

'1: Direkte Portausgabe

Portd = Dat Or &HF0

End If

If Kom = 2 Then

'2: Wartezeit

If Dat = 0 Then Waitms 1

If Dat = 1 Then Waitms 2

If Dat = 2 Then Waitms 5

If Dat = 3 Then Waitms 10

If Dat = 4 Then Waitms 20

If Dat = 5 Then Waitms 50

If Dat = 6 Then Waitms 100

If Dat = 7 Then Waitms 200

If Dat = 8 Then Waitms 500

If Dat = 9 Then Waitms 1000

If Dat = 10 Then Waitms 2000

If Dat = 11 Then Waitms 5000

If Dat = 12 Then Waitms 10000

If Dat = 13 Then Waitms 20000

If Dat = 14 Then Waitms 30000

If Dat = 15 Then Waitms 60000

End If

If Kom = 3 Then

'3: Sprung - relativ

Addr = Addr - 1

Addr = Addr - Dat

End If

If Kom = 4 Then

A = Dat

End If

If Kom = 5 Then

'Variablen

If Dat = 1 Then B = A

If Dat = 2 Then C = A

If Dat = 3 Then D = A

If Dat = 4 Then Portd = A Or &HF0

'Port

```

If Dat = 5 Then
    If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1
                                                    'Portbits
End If
If Dat = 6 Then
    If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1
End If
If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If
If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If
If Dat = 9 Then
    Dd = A * 17
                                                    ' PWM
    Pwm1a = Dd
                                                    ' PWM
End If
End If
If Kom = 6 Then
    If Dat = 1 Then A = B
                                                    'Variablen
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinc
                                                    'Port
    If Dat = 5 Then A = Pinc.0
                                                    'Portbits
    If Dat = 6 Then A = Pinc.1
    If Dat = 7 Then A = Pinc.2
    If Dat = 8 Then A = Pinc.3
    If Dat = 9 Then
        Dd = Getadc(4)
                                                    ' ADC
        Dd = Dd / 64
        A = Dd
    End If
    If Dat = 10 Then
        Dd = Getadc(5)
                                                    ' ADC
        Dd = Dd / 64
        A = Dd
    End If
End If
If Kom = 7 Then
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B
    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B
    If Dat = 9 Then A = A Xor B

```

```

    If Dat = 10 Then A = Not A
    A = A And 15
End If
If Kom = 8 Then
    Adrhi = Dat                                'Oberes Nibble der Adresse
End If
If Kom = 9 Then
    Addr = Adrhi * 16                          'Springe absolut 0...255
    Addr = Addr + Dat
End If
If Kom = 10 Then
    If C > 0 Then                                'C-mal
        C = C - 1
        C = C And 15
        Addr = Adrhi * 16                      'Springe absolut 0...255
        Addr = Addr + Dat
    End If
End If
If Kom = 11 Then
    If D > 0 Then                                ' D-mal
        D = D - 1
        D = D And 15
        Addr = Adrhi * 16                      'Springe absolut 0...255
        Addr = Addr + Dat
    End If
End If
If Kom = 12 Then
    If Dat = 1 Then
        If A > B Then Addr = Addr + 1
    End If
    If Dat = 2 Then
        If A < B Then Addr = Addr + 1
    End If
    If Dat = 3 Then
        If A = B Then Addr = Addr + 1
    End If
    If Dat = 4 Then
        If Pinc.0 = 1 Then Addr = Addr + 1
    End If
    If Dat = 5 Then
        If Pinc.1 = 1 Then Addr = Addr + 1
    End If
    If Dat = 6 Then
        If Pinc.2 = 1 Then Addr = Addr + 1
    End If
    If Dat = 7 Then

```

```

    If Pinc.3 = 1 Then Addr = Addr + 1
End If
If Dat = 8 Then
    If Pinc.0 = 0 Then Addr = Addr + 1
End If
If Dat = 9 Then
    If Pinc.1 = 0 Then Addr = Addr + 1
End If
If Dat = 10 Then
    If Pinc.2 = 0 Then Addr = Addr + 1
End If
If Dat = 11 Then
    If Pinc.3 = 0 Then Addr = Addr + 1
End If
If Dat = 12 Then
    If Pind.6 = 0 Then Addr = Addr + 1
End If
If Dat = 13 Then
    If Pind.7 = 0 Then Addr = Addr + 1
End If
If Dat = 14 Then
    If Pind.6 = 1 Then Addr = Addr + 1
End If
If Dat = 15 Then
    If Pind.7 = 1 Then Addr = Addr + 1
End If

End If
If Kom = 13 Then
    Addrret = Addr
    Addr = Addrhi * 16
                                'Call Unterprogramm absolut 0...255
    Addr = Addr + Dat
End If
If Kom = 14 Then
    Addr = Addrret
                                'Return
End If
Loop
End If

Programmieren:
Addr = 0
Prog = 0
Do
    Adrlo = Addr And 15
                                'Adresse anzeigen
    Portd = Adrlo Or &HF0
    Waitms 300
    Portd = 0 Or &HF0

```

```

Waitms 200
Readeeprom Eebyte , Addr
Dat = Eebyte And 15
Kom = Eebyte / 16
Portd = Kom Or &HF0                                'Befehl anzeigen
Do
Loop Until S2 = 1
Waitms 50

Prog = 1                                            'Phase 1: Befehl anzeigen
Do
  If S1 = 0 Then

    If Prog = 1 Then
      Prog = 2
      Kom = 15
    End If
    If Prog = 2 Then                                'Phase 2: Befehl verändert
      Kom = Kom + 1
      Kom = Kom And 15
      Portd = Kom Or &HF0
    End If
    If Prog = 3 Then :
      'Phase 3: Befehl unverändert, Daten ändern
      Prog = 5
      Dat = 15
    End If
    If Prog = 4 Then                                'Phase 4: Befehl und Daten geändert
      Prog = 5
      Dat = 15
    End If
    If Prog = 5 Then                                'Phase 5: Daten verändert
      Dat = Dat + 1
      Dat = Dat And 15
      Portd = Dat Or &HF0
    End If
    Waitms 50
  Do
  Loop Until S1 = 1
  Waitms 50
End If

If S2 = 0 Then
  If Prog = 3 Then Prog = 7
                                'nur angezeigt, nicht verändert
  If Prog = 1 Then
    Portd = Dat Or &HF0
    Prog = 3

```

```

End If
If Prog = 4 Then
    Portd = 255 - Dat
    Prog = 6
End If
If Prog = 2 Then
    Portd = Dat Or &HF0
    Prog = 4
End If
If Prog = 6 Then
    'nur Kommando wurde verändert

    Dat = Dat And 15
    Eebyte = Kom * 16
    Eebyte = Eebyte + Dat
    Writeeprom Eebyte , Addr
    Portd = 255 - 0
    Waitms 600
    Addr = Addr + 1
    Prog = 0
End If
If Prog = 5 Then
    'Daten wurden verändert

    Dat = Dat And 15
    Eebyte = Kom * 16
    Eebyte = Eebyte + Dat
    Writeeprom Eebyte , Addr
    Portd = 0 Or &HF0
    Waitms 600
    Addr = Addr + 1
    Prog = 0
End If
If Prog = 7 Then
    Addr = Addr + 1
    Prog = 0
End If
Waitms 50
Do
    Loop Until S2 = 1
    Waitms 50
End If
Loop Until Prog = 0
Loop

```

End

---

Korrektur zum Port-Lesen, von Manfred Tischer

Im Programm muss man bei Verarbeitung des Befehls A=Din beachten, dass ja nur das untere Nibble des Bytes genutzt wird. Die beiden AD-Eingänge sind ja auch auf PortC (Teile des

oberen Nibble). Von daher müssen beim Verarbeiten dieses Befehls die oberen Bits ausgeblendet werden. Ansonsten funktionieren anschließende Vergleiche ggf. nicht korrekt.

z.B.:

A=Din

B=A

A=1110

A=B? (Ergebnis des Vergleichs hängt vom Eingangssignal an den Analogeingängen ab)

Mit folgender Änderung für das Kommando 6 ist das Problem aus meiner Sicht behoben.

```

If Kom = 6 Then
  If Dat = 1 Then A = B
  If Dat = 2 Then A = C
  If Dat = 3 Then A = D
  If Dat = 4 Then A = &H0F And Pinc
  If Dat = 5 Then A = Pinc.0
  If Dat = 6 Then A = Pinc.1
  If Dat = 7 Then A = Pinc.2
  If Dat = 8 Then A = Pinc.3
  If Dat = 9 Then
    Dd = Getadc(4)
    Dd = Dd / 64
    A = Dd
  End If
  If Dat = 10 Then
    Dd = Getadc(5)
    Dd = Dd / 64
    A = Dd
  End If
End If

```

Download der korrigierten Version, kompiliert mit Bascom

1.11.9.8: [Tpsm8\\_n3.zip](#)

(Hinweis von B.K: Daraufhin habe ich noch mal in den Quelltext der Holtek-Version geschaut, dort war es schon genauso wie jetzt hier koorigiert:

In Holtek-C: if (Dat == 4) A = \_pa & 15;)



## TPS CODE Part 1

```

'
' -----
' Tasten-programmierbare Steuerung TPS
' Pushbutton Programmable Controller
' Test 1: Interpreter, the first 3 Instructions
' -----

'
'
$regfile    = "m168def.dat"
$crystal    = 11059200
$hwstack    = 32
$swstack    = 64
$framesize  = 64

'
'
Dim Addr    As Byte
Dim Eebyte  As Byte
Dim Dat     As Byte
Dim Kom     As Byte

'
'
Ddrd  = &HFF           'D0...D1 Outputs
Portd = &H0F           'STK500 inverted
Portc = &H0F           'C0..C3 Inputs with Pullup

'
'
S1 Alias Pinc.3        'Data Input
S2 Alias Pinc.0        'Programming

'
Waitms 200

'
Dat = &H11 : Writeeprom Dat , 0      'Dout=1
Dat = &H29 : Writeeprom Dat , 1      '1000 ms
Dat = &H18 : Writeeprom Dat , 2      'Dout=8
Dat = &H29 : Writeeprom Dat , 3      '1000 ms
Dat = &H34 : Writeeprom Dat , 4      'Adr = Adr - 4

'
Waitms 200

'
If S2 = 0 Then
    Goto Programmieren              ' Programming
Else
    Ausfuehren:                     ' Exacute
    Addr = 0

```

Do

```

Readeeprom Eebyte , Addr '
Addr = Addr + 1          '
Dat = Eebyte And 15      '
Kom = Eebyte / 16        '

```

```

'
'
'
'

```

```

If Kom = 1 Then                                '1: Direct Port Output 0 to F
    Portd = 255 - Dat    'inverted Port Output as of STK500
End If

```

```

'
'
'
'

```

```

If Kom = 2 Then                                '2: Waiting Time 1ms to 1 min
    If Dat = 0 Then    Waitms 1
    If Dat = 1 Then    Waitms 2
    If Dat = 2 Then    Waitms 5
    If Dat = 3 Then    Waitms 10
    If Dat = 4 Then    Waitms 20
    If Dat = 5 Then    Waitms 50
    If Dat = 6 Then    Waitms 100
    If Dat = 7 Then    Waitms 200
    If Dat = 8 Then    Waitms 500
    If Dat = 9 Then    Waitms 1000
    If Dat = 10 Then   Waitms 2000
    If Dat = 11 Then   Waitms 5000
    If Dat = 12 Then   Waitms 10000
    If Dat = 13 Then   Waitms 20000
    If Dat = 14 Then   Waitms 30000
    If Dat = 15 Then   Waitms 60000
End If

```

```

'
'
'
'

```

```

If Kom = 3 Then                                '3: Jump back - relative
    Addr = Addr - 1
    Addr = Addr - Dat
End If

```

```
    Loop  
End If
```

```
Programmieren:
```

```
' Programming
```

```
    Do  
    Loop
```

```
End
```

## TPS CODE Part 2

```
'
'
'
'
```

## Programmieren:

```
Addr = 0
```

```
Prog = 0
```

```
Do
```

```
  Adrlo = Addr And 15
```

```
'Adresse anzeigen
```

```
  Portd = 255 - Adr
```

```
  Waitms 300
```

```
  Portd = 255 - 0
```

```
  Waitms 200
```

```
  Readeeprom Eebyte , Addr
```

```
  Dat = Eebyte And 15
```

```
  Kom = Eebyte / 16
```

```
  Portd = 255 - Kom
```

```
'Befehl anzeigen
```

```
Do
```

```
Loop Until S2 = 1
```

```
Waitms 50
```

```
Prog = 1
```

```
'Phase 1: Befehl anzeigen
```

```
Do
```

```
  If S1 = 0 Then
```

```
    If Prog = 1 Then
```

```
      Prog = 2
```

```
      Kom = 15
```

```
    End If
```

```
    If Prog = 2 Then
```

```
'Phase 2: Befehl verändert
```

```
      Kom = Kom + 1
```

```
      Kom = Kom And 15
```

```
      Portd = 255 - Kom
```

```
    End If
```

```

If Prog = 3 Then :
    'Phase 3: Befehl unverändert, Daten ändern
    Prog = 5
    Dat = 15
End If

If Prog = 4 Then
    'Phase 4: Befehl und Daten geändert
    Prog = 5
    Dat = 15
End If

If Prog = 5 Then
    'Phase 5: Daten verändert
    Dat = Dat + 1
    Dat = Dat And 15
    Portd = 255 - Dat
End If

Waitms 50

Do
    Loop Until S1 = 1
    Waitms 50
End If

If S2 = 0 Then
    If Prog = 3 Then Prog = 7
        'nur angezeigt, nicht verändert

    If Prog = 1 Then
        Portd = 255 - Dat
        Prog = 3
    End If

    If Prog = 4 Then
        Portd = 255 - Dat
        Prog = 6
    End If

    If Prog = 2 Then
        Portd = 255 - Dat
        Prog = 4
    End If

    If Prog = 6 Then
        'nur Kommando wurde verändert
        Dat = Dat And 15
        Eebyte = Kom * 16
        Eebyte = Eebyte + Dat
        Writeeprom Eebyte , Addr
        Portd = 255 - 0
        Waitms 600
        Adr = Adr + 1
    End If
End If

```

```

    Prog = 0
End If
If Prog = 5 Then          'Daten wurden verändert
    Dat = Dat And 15
    Eebyte = Kom * 16
    Eebyte = Eebyte + Dat
    Writeeprom Eebyte , Addr
    Portd = 255 - 0
    Waitms 600
    Addr = Addr + 1
    Prog = 0
End If
If Prog = 7 Then
    Addr = Addr + 1
    Prog = 0
End If
Waitms 50
Do
    Loop Until S2 = 1
    Waitms 50
End If
    Loop Until Prog = 0
Loop

End

```

## TPS CODE Part 3

```
'
'
'
'
```

```
'-----
' Tasten-programmierbare Steuerung TPS
' Test 3: Variablen und Rechenbefehle
'-----
```

```
Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim D As Byte
```

```
Config Adc = Single , Prescaler = Auto , Reference = Off
Start Adc
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare A Pwm =
Clear Down , Compare B Pwm = Clear Down
Start Timer1
```

```
...
```

```
Ausfuehren:
```

```
Addr = 0
```

```
Do
```

```
Readeeprom Eebyte , Addr
```

```
Addr = Addr + 1
```

```
Dat = Eebyte And 15
```

```
Kom = Eebyte / 16
```

```
If Kom = 1 Then
```

```
'1: Direkte Portausgabe
```

```
Portd = 255 - Dat
```

```
'invertierte Portausgabe wegen STK500
```

```
End If
```

```
If Kom = 2 Then
```

```
'2: Wartezeit
```

```
If Dat = 0 Then Waitms 1
```

```
If Dat = 1 Then Waitms 2
```

```

If Dat = 2 Then Waitms 5
If Dat = 3 Then Waitms 10
If Dat = 4 Then Waitms 20
If Dat = 5 Then Waitms 50
If Dat = 6 Then Waitms 100
If Dat = 7 Then Waitms 200
If Dat = 8 Then Waitms 500
If Dat = 9 Then Waitms 1000
If Dat = 10 Then Waitms 2000
If Dat = 11 Then Waitms 5000
If Dat = 12 Then Waitms 10000
If Dat = 13 Then Waitms 20000
If Dat = 14 Then Waitms 30000
If Dat = 15 Then Waitms 60000
End If

If Kom = 3 Then                                '3: Sprung - relativ
    Addr = Addr - 1
    Addr = Addr - Dat
End If

If Kom = 4 Then
    A = Dat
End If

If Kom = 5 Then
    If Dat = 1 Then B = A                        'Variablen
    If Dat = 2 Then C = A
    If Dat = 3 Then D = A
    If Dat = 4 Then Portd = 255 - A              'Port
    If Dat = 5 Then
        If A.0 = 0 Then Portd.0 = 0 Else Portd.0 = 1
                                                'Portbits
    End If
End If

If Dat = 6 Then
    If A.0 = 0 Then Portd.1 = 0 Else Portd.1 = 1
End If

If Dat = 7 Then
    If A.0 = 0 Then Portd.2 = 0 Else Portd.2 = 1
End If

If Dat = 8 Then
    If A.0 = 0 Then Portd.3 = 0 Else Portd.3 = 1
End If

```



```

    If Dat = 9 Then
        Dd = A * 17
        Pwmla = Dd
    End If
End If

If Kom = 6 Then
    If Dat = 1 Then A = B
    If Dat = 2 Then A = C
    If Dat = 3 Then A = D
    If Dat = 4 Then A = Pinb
    If Dat = 5 Then A = Portb.0
    If Dat = 6 Then A = Portb.1
    If Dat = 7 Then A = Portb.2
    If Dat = 8 Then A = Portb.3
    If Dat = 9 Then
        Dd = Getadc(4)
        Dd = Dd / 64
        A = Dd
    End If
End If

If Kom = 7 Then
    If Dat = 1 Then A = A + 1
    If Dat = 2 Then A = A - 1
    If Dat = 3 Then A = A + B
    If Dat = 4 Then A = A - B
    If Dat = 5 Then A = A * B
    If Dat = 6 Then A = A / B
    If Dat = 7 Then A = A And B
    If Dat = 8 Then A = A Or B
    If Dat = 9 Then A = A Xor B
    If Dat = 10 Then A = Not A
    A = A And 15
End If

Loop
End If

```

' PWM  
' PWM  
'Variablen  
'Port  
'Portbits  
'ADC

## TPS CODE Part 4

```

'
'
'
'

'-----
' Tasten-programmierbare Steuerung TPS
' Test 4: Sprünge und Verzweigungen
'-----

$regfile    = "m168def.dat"      '
$crystal    = 11059200          '
$hwstack    = 32                 '
$swstack    = 64                 '
$framesize  = 64                 '


Dim Addr    As Byte             '
Dim Eebyte  As Byte             '
Dim Dat     As Byte             '
Dim Kom     As Byte             '
Dim Adrhi   As Byte             '
Dim Adrlo   As Byte             '
Dim Adrret  As Byte             '
Dim Prog    As Byte             '
Dim Dd      As Word             '


Waitms 200                       '

If S2 = 0 Then                    '
    Goto Programmieren           '

Else
Ausfuehren:                      '
    Addr = 0                     '

    Do
        Readeeprom Eebyte , Addr
        Addr = Addr + 1
        Dat = Eebyte And 15
        Kom = Eebyte / 16
    ...
        If Kom = 8 Then
            Adrhi = Dat
            'Oberes Nibble der Adresse

```

End If

```
If Kom = 9 Then
  Addr = Adrhi * 16          'Springe absolut 0...255
  Addr = Addr + Dat
End If
```

```
If Kom = 10 Then
  C = C - 1
  C = C And 15
  If C > 0 Then              'C-mal
    Addr = Adrhi * 16        'Springe absolut 0...255
    Addr = Addr + Dat
  End If
```

End If

```
If Kom = 11 Then
  D = D - 1
  D = D And 15
  If D > 0 Then              'D-mal
    Addr = Adrhi * 16        'Springe absolut 0...255
    Addr = Addr + Dat
  End If
```

End If

```
If Kom = 12 Then
  If Dat = 1 Then
    If A > B Then Addr = Addr + 1
  End If
```

```
  If Dat = 2 Then
    If A < B Then Addr = Addr + 1
  End If
```

```
  If Dat = 3 Then
    If A = B Then Addr = Addr + 1
  End If
```

```
  If Dat = 4 Then
    If Pinc.0 = 1 Then Addr = Addr + 1
  End If
```

```
  If Dat = 5 Then
    If Pinc.1 = 1 Then Addr = Addr + 1
  End If
```

```
If Dat = 6 Then
    If Pinc.2 = 1 Then Addr = Addr + 1
End If
```

```
If Dat = 7 Then
    If Pinc.3 = 1 Then Addr = Addr + 1
End If
```

```
If Dat = 8 Then
    If Pinc.0 = 0 Then Addr = Addr + 1
End If
```

```
If Dat = 9 Then
    If Pinc.1 = 0 Then Addr = Addr + 1
End If
```

```
If Dat = 10 Then
    If Pinc.2 = 0 Then Addr = Addr + 1
End If
```

```
If Dat = 11 Then
    If Pinc.3 = 0 Then Addr = Addr + 1
End If
End If
```

```
If Kom = 13 Then
    Addrret = Addr
    Addr = Adrhi * 16 'Call Unterprogramm absolut 0...255
    Addr = Adr + Dat
End If
```

```
If Kom = 14 Then
    Addr = Addrret 'Return
End If
```

```
Loop
End If
```

## BASCOM

[https://www.mcselec.com/index.php?Itemid=41&id=14&option=com\\_content&task=view](https://www.mcselec.com/index.php?Itemid=41&id=14&option=com_content&task=view)

### BASCOM-AVR



**BASCOM-AVR®** is the original **Windows BASIC COMPILER** for the **AVR** family. It is designed to run on **XP/VISTA/WIN7** and **WIN8**

**BASCOM® AVR®**

**BASCOM-AVR®** is the original **Windows BASIC COMPILER** for the **AVR** family. It is designed to run on **XP/VISTA/WIN7, WIN8 and WIN10**

This product description is updated in 2016. But we do not change it each time we update the software. In the help you can find a list of all statements and functions.

### Key Benefits

- Structured **BASIC** with labels.
- Structured programming with IF-THEN-ELSE-END IF, DO-LOOP, WHILE-WEND, SELECT- CASE.
- Fast machine code instead of interpreted code.
- Variables and labels can be as long as 32 characters.
- Bit, Byte, Integer, Word, Long, Single, Double and String variables.
- Large set of Trig Floating point functions.
- Date & Time calculation functions.
- Compiled programs work with all AVR microprocessors that have internal memory.
- Statements are highly compatible with Microsoft's VB/QB.
- Special commands for **LCD**-displays , **I2C** chips and **1WIRE** chips, PC keyboard, matrix keyboard, **RC5** reception, software UART, SPI , graphical LCD, send IR RC5, **RC6** or Sony code.
- **TCP/IP** with W3100A/W5100/W5200/W5300/W5500 chips.
- Built in AVR-DOS functions like MKDIR, CHDIR, DIR, OPEN,

CLOSE, etc. Just as they work in QB/VB !

- Local variables, user functions, library support.
- Integrated terminal emulator with download option..
- Integrated simulator for testing.
- Integrated ISP programmer (application note AVR910.ASM).
- Integrated STK200 programmer and STK300 programmer.  
Also supported is the low cost Sample Electronics programmer.  
Can be built in 10 minutes! Many other programmers supported via the Universal Interface.
- Many supported programmers like STK500, STK600, MKII, USBASP, JTAG , Arduino
- Editor with statement highlighting.
- Context sensitive help.
- DEMO version compiles **4KB** of binary code.  
Well suited for the ATmega48.
- English and German Books available
- AT mouse simulator, AT keyboard simulator, I2C Slave available as add on.
- This product is developed in 1995 and is updated regularly.

**The following statements are supported  
(actually there are many more - look in the on-line helpfile):**

### **Decision and structures**

IF, THEN, ELSE, ELSEIF, END IF, DO, LOOP,  
WHILE, WEND, UNTIL, EXIT DO, EXIT WHILE, FOR,  
NEXT, TO, STEP, EXIT FOR, ON .. GOTO/GOSUB,  
SELECT, CASE.

### **Input and output**

PRINT, INPUT, INKEY, PRINT, INPUTHEX,  
LCD, UPPERLINE, LOWERLINE, DISPLAY ON/OFF,  
CURSOR ON/OFF/BLINK/NOBLINK, HOME, LOCATE,  
SHIFTLCD LEFT/RIGHT, SHIFTCURSOR LEFT/RIGHT, CLS,  
DEFLCDCHAR, WAITKEY, INPUTBIN, PRINTBIN,  
OPEN, CLOSE, DEBOUNCE, SHIFTIN, SHIFTOUT,  
GETATKBD, SPC, SERIN, SEROUT

### **Numeric functions**

AND, OR, XOR, INC, DEC, MOD, NOT, ABS, BCD,  
LOG, EXP, SQR, SIN, COS, TAN, ATN, ATN2, ASIN,

ACOS, FIX, ROUND, MOD, SGN, POWER, RAD2DEG,  
DEG2RAD, LOG10, TANH, SINH, COSH.

## **I2C**

I2CSTART, I2CSTOP, I2CWBYTE, I2CRBYTE, I2CSEND  
and I2CRECEIVE.

## **1WIRE**

1WWRITE, 1WREAD, 1WRESET, 1WIRECOUNT,  
1WSEARCHFIRST, 1WSEARCHNEXT.

## **SPI**

SPIINIT, SPIIN, SPIOUT, SPIMOVE.

## **CAN**

CONFIG CANBUSMODE, CONFIG CANMOB, CANBAUD,  
CANRESET, CANCEARMOB, CANCEARALLMOBS, CANSEND,

CANRECEIVE , CANID, CANSELPAGE, CANGETINTS

## **TCP/IP**

TCP/IP routines can be used with the  
W3100/IIM7000/IIM7010/W5100/W5200/W5300 modules.

BASE64DEC , BASE64ENC , IP2STR , UDPREAD , UDPWRITE ,  
UDPWRITESTR , TCPWRITE , TCPWRITESTR , TCPREAD ,  
GETDSTIP , GETDSTPORT , SOCKETSTAT ,  
SOCKETCONNECT , SOCKETLISTEN , GETSOCKET ,  
SOCKETCLOSE , SETTCP , GETTCPREGS , SETTCPREGS ,  
SETIPPROTOCOL , TCPCHECKSUM , SOCKETDISCONNECT ,  
SNTP , TCPREADHEADER , UDPREADHEADER

## **Interrupt programming**

ON INT0/INT1/TIMER0/TIMER1/SERIAL, RETURN, ENABLE,  
DISABLE, COUNTERx, CAPTUREx, INTERRUPTS, CONFIG,  
START, LOAD.

## **Bit manipulation**

SET, RESET, ROTATE, SHIFT, BITWAIT, TOGGLE.

## **Variables**

DIM, BIT , BYTE , INTEGER , WORD, LONG, SINGLE, DOUBLE, STRING , DEFBIT, DEFBYTE, DEFINT, DEFWORD.

### **Miscellaneous**

REM, ' , SWAP, END, STOP, CONST, DELAY, WAIT, WAITMS, GOTO, GOSUB, POWERDOWN, IDLE, DECLARE, CALL, SUB, END SUB, MAKEDEC, MAKEBCD, INP,OUT, ALIAS, DIM , ERASE, DATA, READ, RESTORE, INCR, DECR, PEEK, POKE, CPEEK, FUNCTION, READMAGCARD, BIN2GREY, GREY2BIN, CRC8, CRC16, CRC32, CHECKSUM.

### **Compiler directives**

\$INCLUDE, \$BAUD and \$CRYSTAL, \$SERIALINPUT, \$SERIALOUTPUT, \$RAMSIZE, \$RAMSTART, \$DEFAULT XRAM, \$ASM-\$END ASM, \$LCD, \$EXTERNAL, \$LIB.

### **String manipulation**

STRING, SPACE, LEFT, RIGHT, MID, VAL, HEXVAL, LEN, STR, HEX, LTRIM, RTRIM, TRIM, LCASE, UCASE, FORMAT, FUSING, INSTR, CHARPOS.

### **And many other functions, statements and directives**

**To make a program takes just a few steps :**

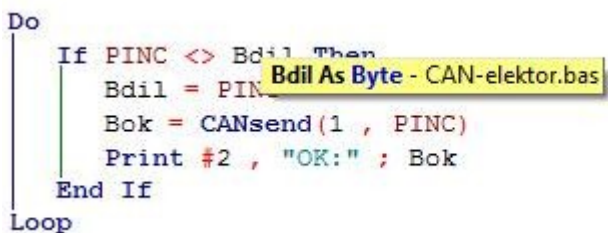
- **Write the program in BASIC**
- **Compile it to fast machine binary code**
- **Test the result with the integrated simulator (with additional hardware you can simulate the hardware too).**
- **Program the chip with one of the integrated programmers. (hardware must be purchased separately)**



This is a screen shot of the editor.  
 You can work in normal mode or project mode.  
 At the left you find the Code Explorer.  
 The Code explorer can show unused data in a different colour.

When the Code Explorer is visible, the editor supports Proper Indent and Indent drawing.  
 Indent lines can be a great visual help.

A tool tip with info can be shown by pressing SHIFT :



```

Do
  If PINC <> Bdil Then
    Bdil = PINC
    Bok = CANsend(1, PINC)
    Print #2, "OK:" ; Bok
  End If
Loop
  
```

Tip: Bdil As Byte - CAN-elektor.bas

Here it is clear that BDIL is a byte variable, dimension-ed in the module CAN-Elektor.bas  
 It will show info for constants, aliases, variables and functions.

BASCOM-AVR supports the tiny, mega and Xmega processors with internal SRAM.

A full list you find on [avrhelp.mcselec.com](http://avrhelp.mcselec.com) under the Chips topic.

For the new Xtiny processors you need an add-on which need to be purchased separately.

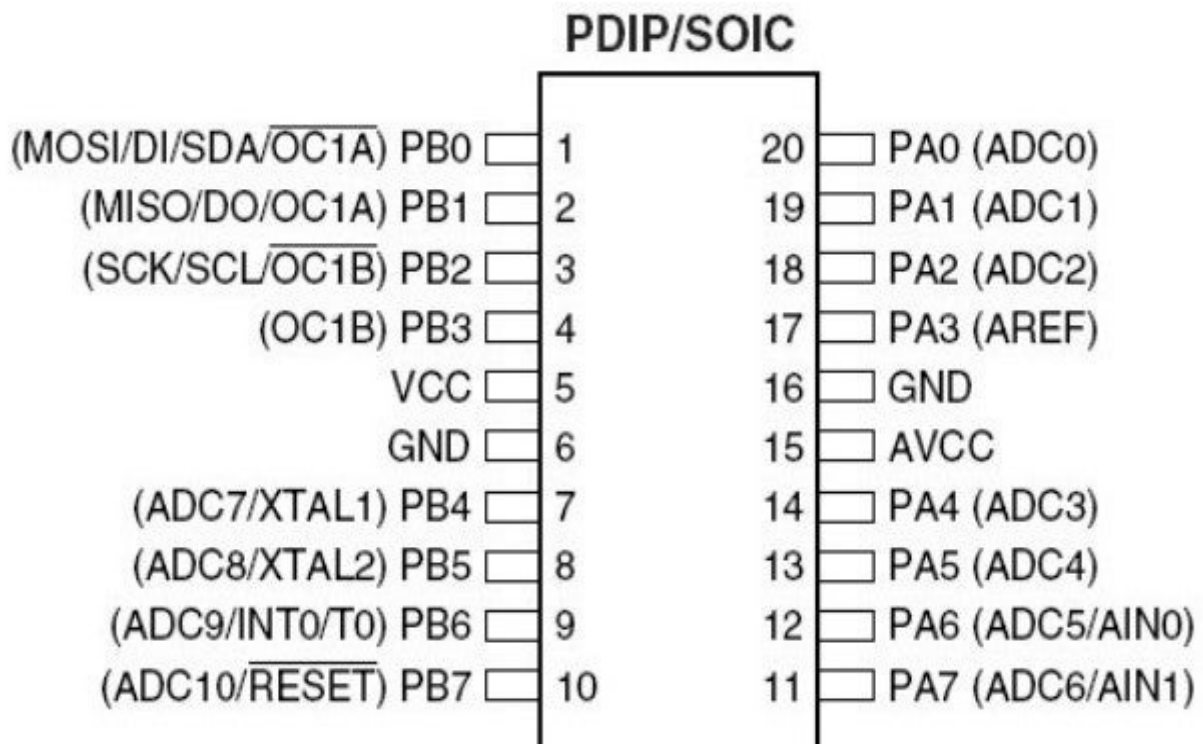
<https://www.instructables.com/Getting-Started-with-Atmel-AVR-and-BASCOM/>

# Getting Started With Atmel AVR and BASCOM

By [askjerry](#) in [CircuitsMicrocontrollers](#)

DownloadFavorite

## Introduction: Getting Started With Atmel AVR and BASCOM



By [askjerry](#) [Askjerry...everyone else does!](#) Follow



More by the author:



About: I like to create YouTube videos and have a channel. I also like to work with CNC, electronics, and I build robots. It's been a while, but another hobby is high powered rockets. [More About askjerry »](#)

I have seen plenty of Instructables showing how to work with microprocessors, but they all assume that you have worked with them before and know what you are doing. I have not seen an Instructable that takes you from nothing and builds on each step.

What we will do here is to start with a bare breadboard and build each connection and each component, until we have everything, we need to program a microcontroller to do something. In this Instructable we will blink some LEDs in sequence... then if you build this circuit... your first project can be to change the code slightly to make it into a traffic light.

### **I picked an older Atmel chip, the Tiny-26 to get started.**

It is a smaller microprocessor, very inexpensive, and easy to understand. Once you understand what we are doing here, you may want to try a more powerful chip like the Mega-328P which has more pins and more memory.

**Note:** The **Tiny-261**, **Tiny-461**, and **Tiny-861** are pin compatible newer versions of the Tiny-26. They have 2K, 4K, and 8K of memory.

**If used, simply change the header** by selecting the appropriate chip and recompile the program to use the newer version. The new chips have more functions that can be assigned to each pin.

*See the datasheets for more details.*

**Tiny 261, 461, 861**      [Datasheet](#) (PDF)

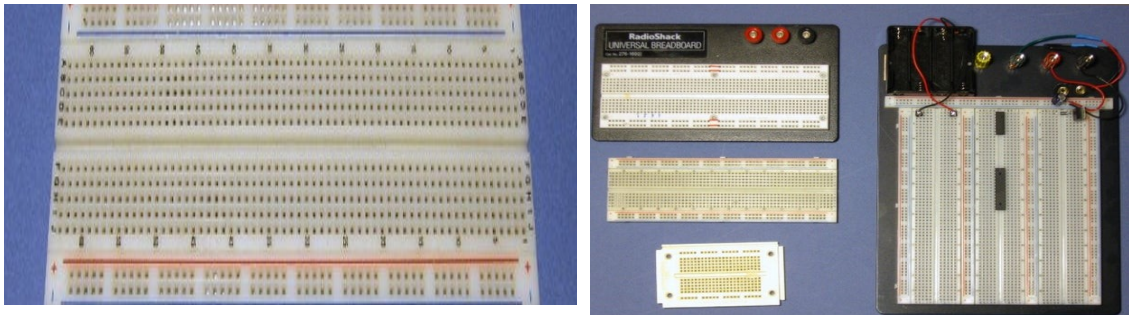
**Tiny 26**                [Datasheet](#) (PDF)

Below is an image with the pins for the chip we will be using... we will be connecting the power and ground... in this case 5v. So where do we get 5 volts? We will build a power supply from a 9v battery.

Let's get started!

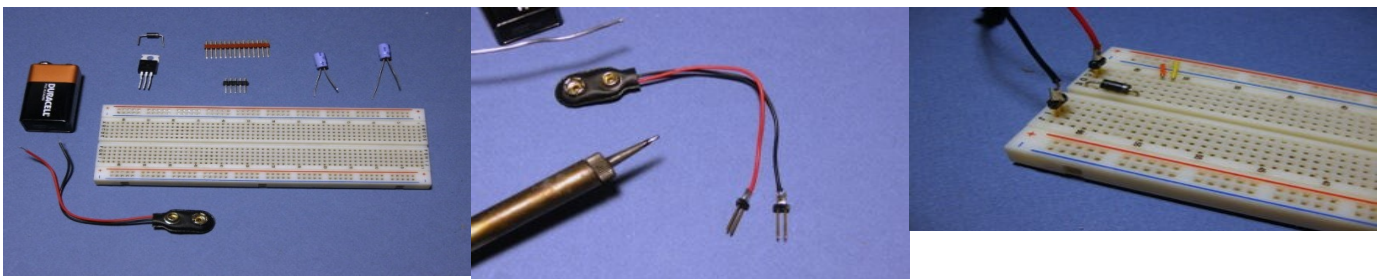
Video posted in a larger size at: <http://www.youtube.com/watch?v=Jxica6Yenh8>

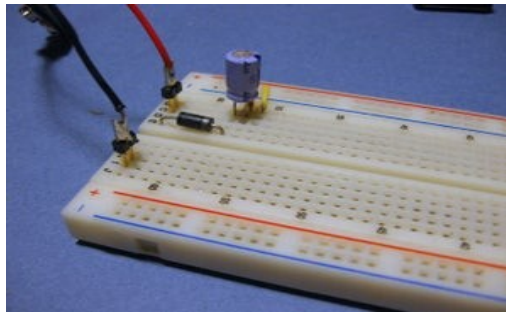
## **Step 1: Breadboards and Building Circuits.**



Below you can see a breadboard... they come in many sizes depending on how complex the circuit you want to build will be. We will use a medium sized breadboard with enough room for our power supply, the microprocessor, and the LEDs we want to control.

## Step 2: Starting With the Power Supply





We will use a voltage regulator called a 7805. It has three pins, the first is the input, next is the ground, and last is the output voltage... in this case 5 volts. The chip needs to have greater than 6 volts to be able to regulate it down to 5 volts... but it cannot have more than 36 volts. The 9 volt battery works well in this application.

We also want to install a diode into the circuit... it will allow current to flow in only one direction. **The reason we install it is so that if we connect the battery backwards it will not let the current flow and will protect our circuit from damage.** The diode has a line printed on it, this should be on the most negative side, in our case... it connects to the input of the voltage regulator.

The 9 volt battery clip uses stranded wires... they do not plug into a breadboard very well so we need to make it so that we can install the connections properly. We will use what is called a SIP header. (Single Inline Pins) The wires are soldered to the pins and then can be inserted into the breadboard.

If you look at image #3, you can see that the wires are soldered to TWO pins. This is done to provide a stronger connection that won't turn or jiggle loose. *(It's a good tip to remember! )*

We insert the pins into the breadboard and install the diode. (Image #4)

In Image #5 you can see that we have added a capacitor... it is a **47uF** rated at least 6 volts... these here are 25 volts... so they are fine. They act to smooth out the voltage when things like LEDs turn on and off. One way to think of them is like the water tank in your toilet... when you flush you need a large flow of water all at once... but the supply line is very small. The tank holds enough water to even out the flow. Likewise, the capacitor holds extra power for when there is a surge from things starting and stopping.

One capacitor goes between the input to the voltage regulator and the ground, the other goes between the ground and the regulated 5 volts. The capacitor has a marking to show which side connects to ground.



In **Image #6** and **Image #7** you can see the completed power supply.

### **Step 3: Wires and Connections**



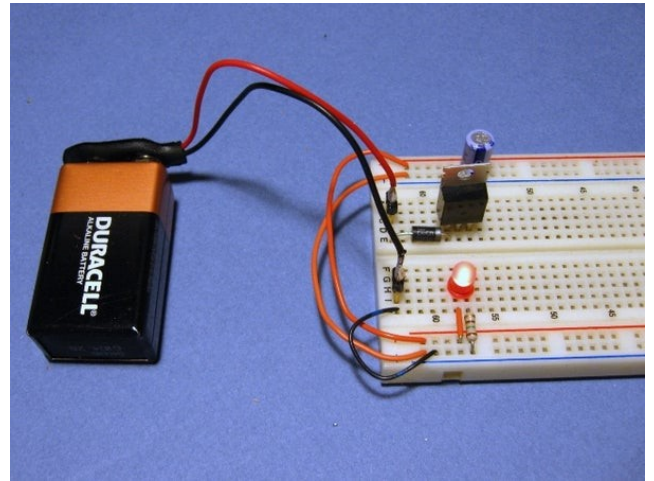
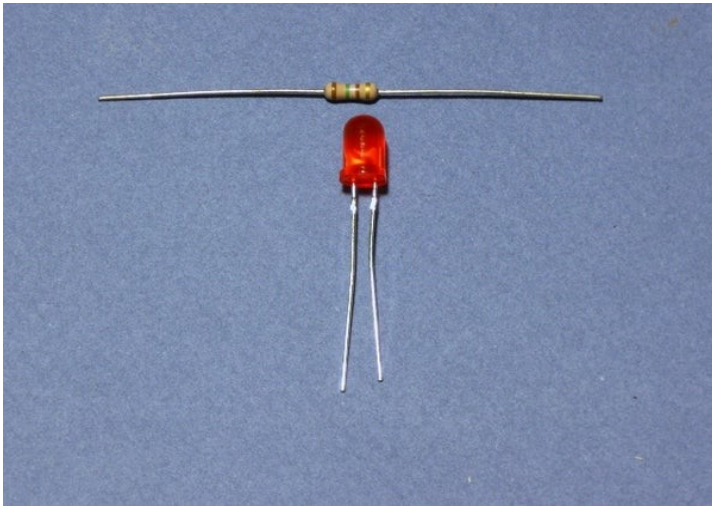
You will notice that my connections to the board are very neat and easy to follow. (At least I hope so!) I am using a wire kit that is available online from several places. The kits are about \$10 to \$20 depending where you look... and refills are available from places like [Digikey](#).

The kits make your wire connections short, clean, and easy to follow when you come back to them in the future. The refills are a bit pricey, but if you get into electronics quite a bit you will find that you don't want to live without them.

A great way to get started is to find some **22 gauge to 24 gauge SOLID wire**. You don't want stranded wires in a breadboard... they just don't work well. Solid wire is useful when you want it stiff and bendable as we are doing here. Stranded wire is used when it must be flexible.

If you see the local telephone repair person ask if they have any scrap 25 pair cable. They often get it in 500 foot rolls... and if it has less than 20 or so feet will often throw it away. The cable has a string that when pulled will cut the jacket and give you a whole bunch of wires as shown in **Image #3**.

### **Step 4: How Do We Know If It's Working???**



So now we have a power supply for the microprocessor and our LEDs. But how do we know if it is working? We can install an LED so that when the power supply is on it will illuminate. This is also a good time to teach you about resistors. We will be working with 5 volts... but an LED is designed to run on only 2 volts. (*Approximate, they vary depending on the type.* )

If we were to connect a 2v LED to the 5v supply, it would get very bright for a very short period of time... then POOF! Here is where you need a little math... don't worry... it's not too bad.

If we look at the specification sheet for the LED, it says that it runs on 2 volts at 20mA. Okay... let's start with the voltage... 5 volts minus 2 volts is 3 volts. So we have 3 volts too much for the LED. (Told you the math wasn't too hard.)

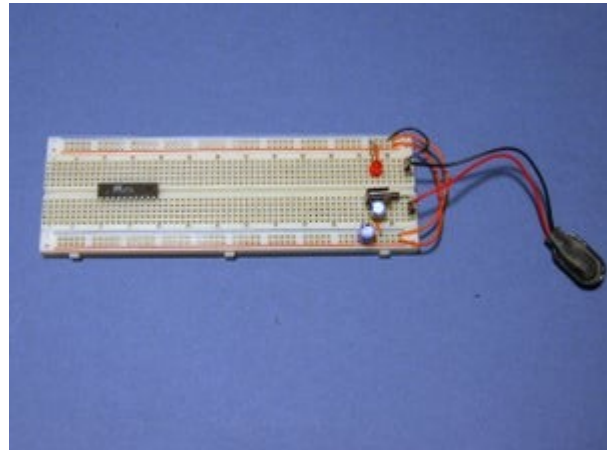
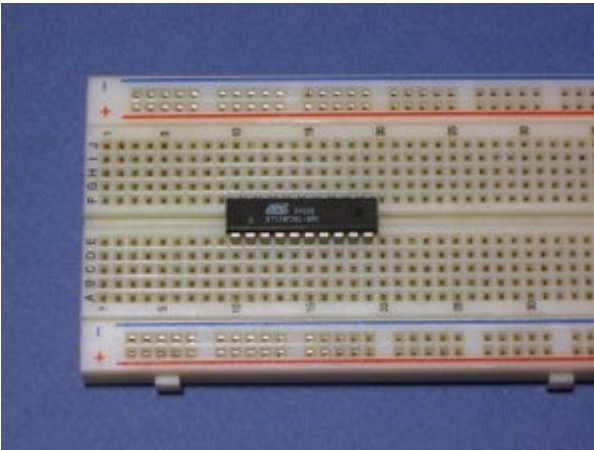
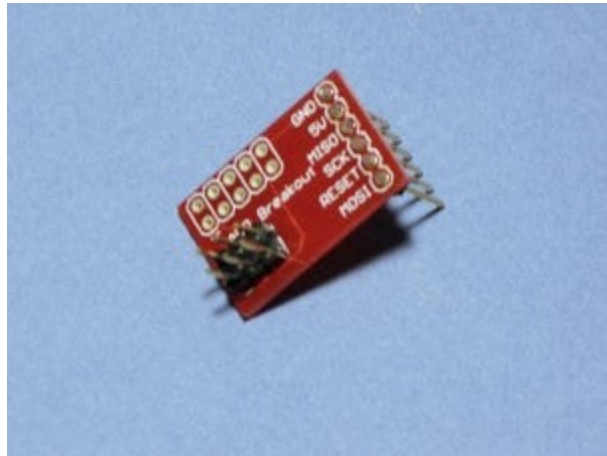
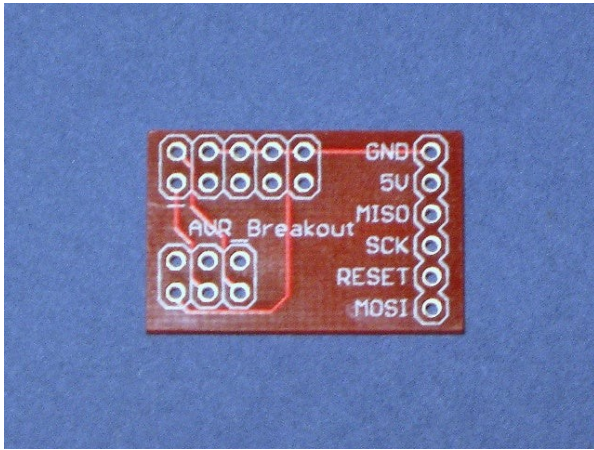
Okay... the LED runs on 20mA... that's the same as 0.020 amps. (mA are 1/1000 of an amp... so 1000mA = 1A) We know that we have 3 volts too much... so divide that by the current that the LED is supposed to run at...  $3 \text{ divided by } 0.020 = 150$ .

So we know that we need to have a resistance of 150 ohms to slow down the flow of electricity to a point where it won't burn out the led.  $\text{Voltage} / \text{Current} = \text{Ohms}$ .

We install our resistor (150 ohms) and our LED onto the board... now we know if we have power or not!

## **Step 5: Adding the Microprocessor**





We now have a working power supply, we understand how a breadboard works, and we have the wires... let's connect up a microprocessor and do something!

On order to make the microprocessor do anything, we need a way to get the software into it. There are four signals that send the data to and from the microprocessor during programming, they need to be connected to a programmer. If you have a PC or Laptop with a parallel printer port, the BASCOM software we will be using later has plans to make a very simple programmer. If you only have USB ports, then you can purchase a USB programmer. There are links on the BASCOM page.

To connect your programmer to the microprocessor, there is usually a 6-pin (or sometimes an older 10-pin) connector. The folks at [Sparkfun](http://www.sparkfun.com) have made a great adapter as shown in images #1 and #2. Once you solder in your pins, simply plug it into the breadboard and connect your wires. You can get yours here: <http://www.sparkfun.com/products/8508>

**Image #3** and **#4** show how we are installing the Tiny-26 microprocessor into the breadboard. Note that we rotated the breadboard 180 degrees so the text would face correctly in the photographs.

Next we connect the power and ground connections, then make the jumpers for the following signals from the [Sparkfun](#) board to the programming pins of the chip.

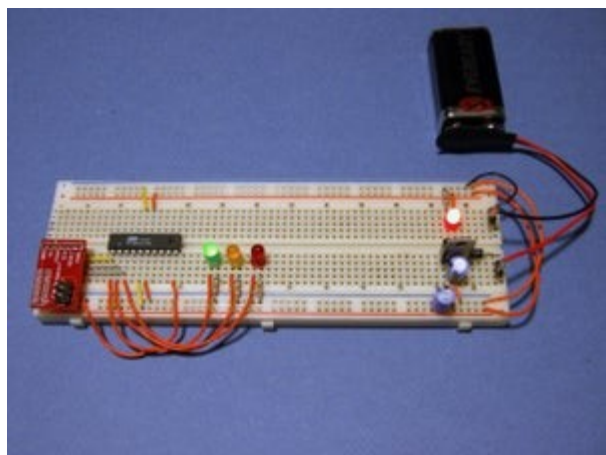
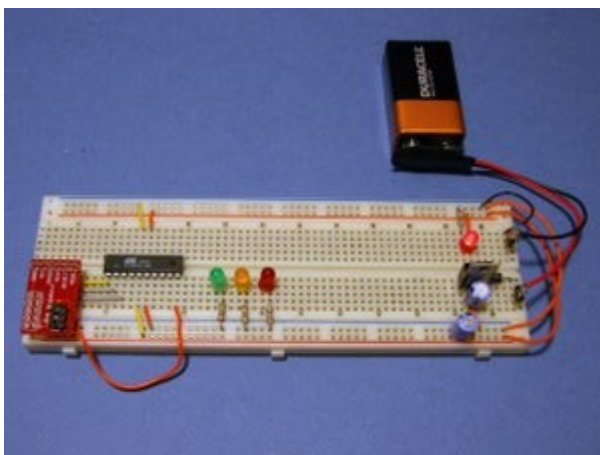
- \* **MISO** - Data into the chip from the PC.
- \* **SCK** - System Clock and timing.
- \* **RESET** - Reset, tells the chip to enter programming mode.
- \* **MOSI** - Data out from the chip to the PC.

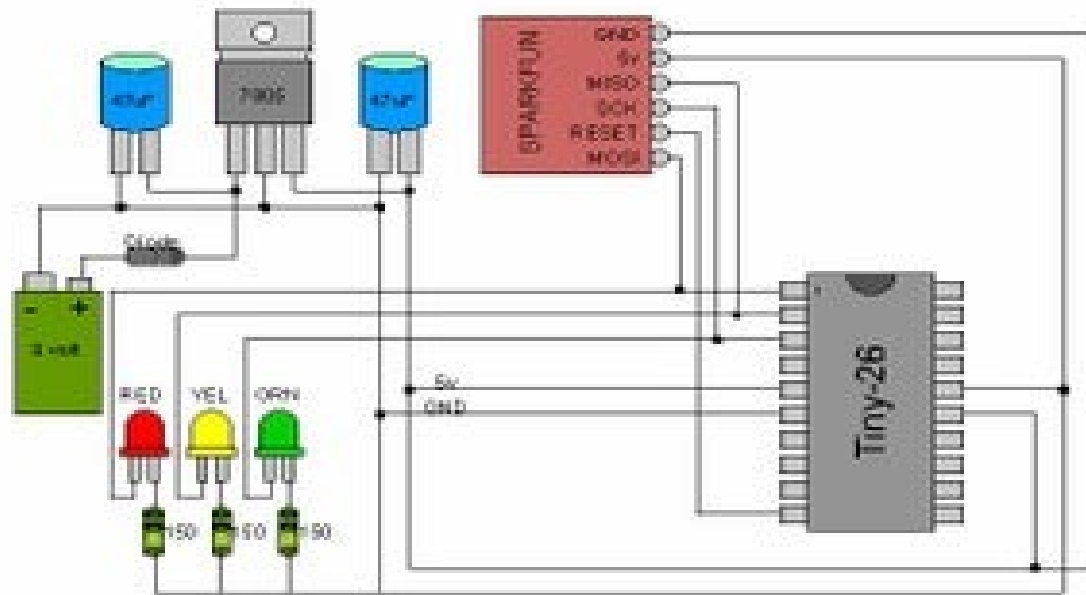
We also have connections for power and ground. Image #7 shows where we want to install the programmer, Image #8 shows the connections, and #9 shows it all assembled.

Image #10 shows the names and connections of the Tiny-26 Pins. You can see that we connected MOSI to MOSI on the chip for example. If you get MOSI and MISO reversed it won't work... not that I ever did that. :-/

## Step 6: Connecting the LEDs

		PDIP/SOIC	
(MOSI/DI/SDA/ $\overline{OC1A}$ ) PB0	1	20	PA0 (ADC0)
(MISO/DO/OC1A) PB1	2	19	PA1 (ADC1)
(SCK/SCL/ $\overline{OC1B}$ ) PB2	3	18	PA2 (ADC2)
(OC1B) PB3	4	17	PA3 (AREF)
VCC	5	16	GND
GND	6	15	AVCC
(ADC7/XTAL1) PB4	7	14	PA4 (ADC3)
(ADC8/XTAL2) PB5	8	13	PA5 (ADC4)
(ADC9/INT0/T0) PB6	9	12	PA6 (ADC5/AIN0)
(ADC10/ $\overline{RESET}$ ) PB7	10	11	PA7 (ADC6/AIN1)





We need something to control with this microprocessor... let's add three LEDs to the circuit.

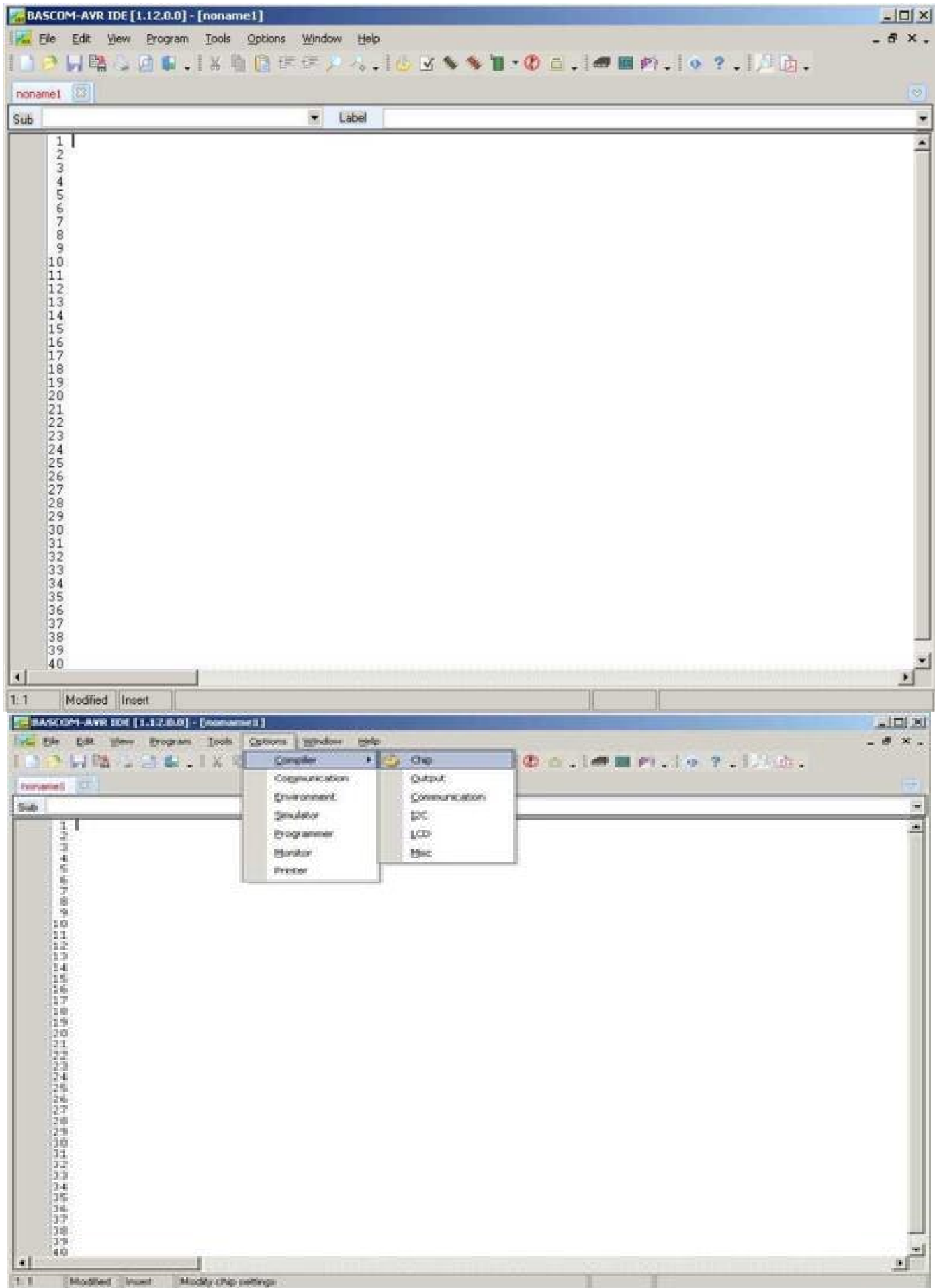
This chip has two ports named **PORTA** and **PORTB** as can be seen in **image #1**. (Look familiar?)

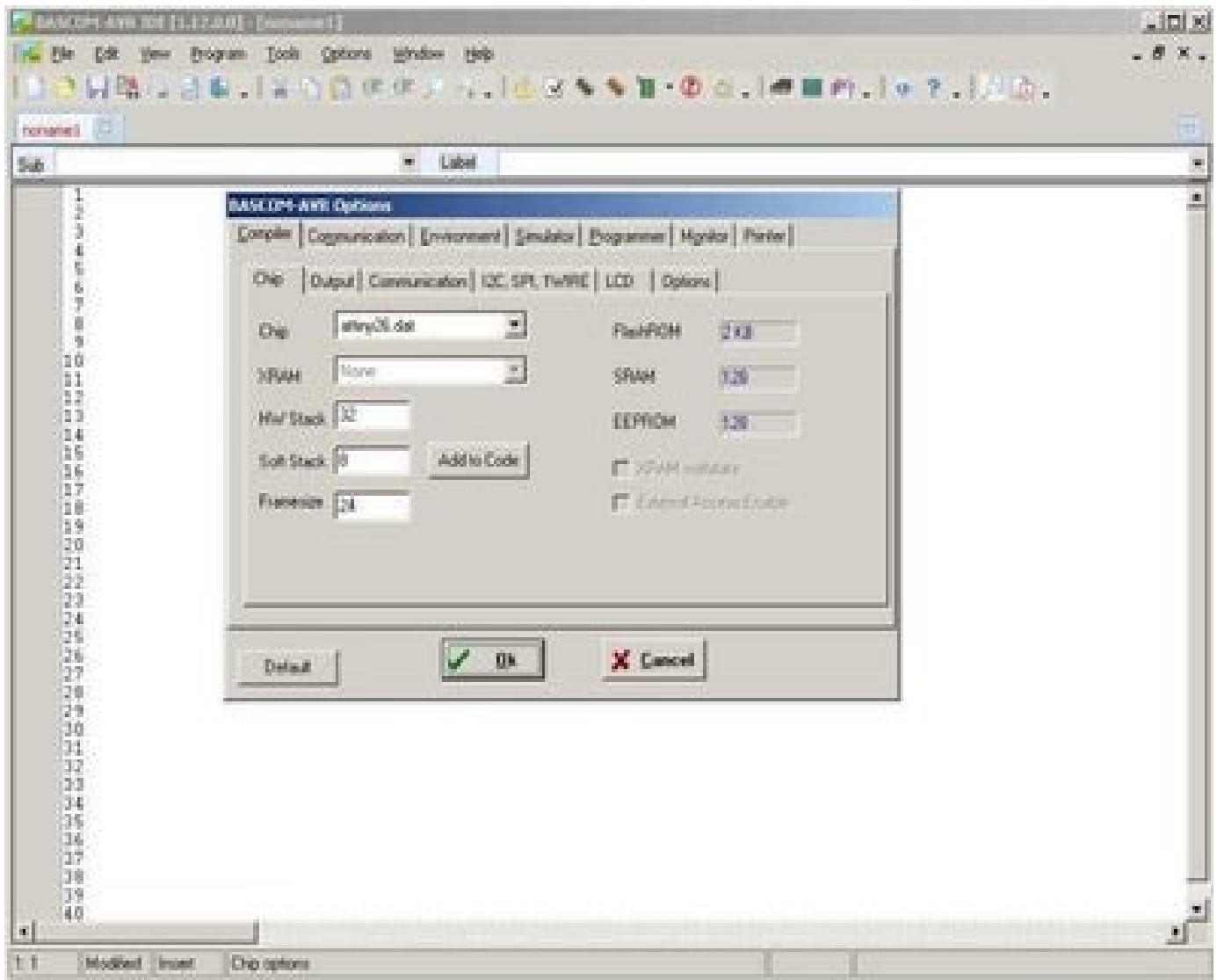
We could have connected the LEDs to **PORTA**, but I wanted to show that you can use the programming port to also control things. Connecting something like a motor controller to the programming port would not be a good idea... the motor would turn on and off uncontrollably during programming. But with the LEDs connected, you will see them blink as we program the chip. I just think that looks cool too.

The chip can support up to 20mA per pin... so a single LED on the pin is fine. If you wanted to connect multiple LEDs to a pin, then you would need to install something to drive the needed power... like a 2N2222 transistor or similar. (But that's for another Instructable!)

If you look at the remaining images you will see that the LEDs are connected and the resistors are installed. Remember that the LED needs to be installed correctly, the FLAT side is connected to ground via the resistor.

## Step 7: Programming the Chip





There are many programming languages to choose from for programming the Atmel series of chips. Some people like to use assembly, others prefer C. I have been programming in BASIC since 1978 so I like to use that language. There is a GREAT version of BASIC for the Atmel that is very powerful and easy to learn, it's called [BASCOSM](http://www.mcselec.com/index.php?option=com_content&task=view&id=14&Itemid=41). You can download it and get more information here: [http://www.mcselec.com/index.php?option=com\\_content&task=view&id=14&Itemid=41](http://www.mcselec.com/index.php?option=com_content&task=view&id=14&Itemid=41)

The demo version will allow you to program up to 4K of memory space... and since this is a 2K microprocessor... that will never be an issue. When your programs get bigger and you migrate to more powerful chips, the program only costs about \$80 which is a real bargain for all it does.

Once you install **BASCOSM**, the screen will look something like **image #1**

**Image #2:** Select options, compiler, then chip. a menu screen will open up.

**Image #3:** Select the TINY26 from the list. then click the **ADD TO CODE** button which will add the commands to the code so that you won't have

to keep selecting the chip type. It defaults with a speed of 4MHZ for the crystal... and needs to be changed to 1MHZ since we will use the internal clock of the chip. The line should read...

**\$CRYSTAL = 1000000**

**Image #4:** Here you can see the code that was generated. It tells the software what kind of chip is selected, what speed we are going to run it at, and it has some other (optional) data to define how the hardware is configured. Once this is in the software, it knows everything needed to program the chip. It wouldn't do anything we would call useful... but it would program okay.

**Image #5:** This is our program... let's go through it.

```
-----
$regfile      = "attiny26.dat"
$crystal      = 1000000
$hstack      = 32
$swstack      = 8
$framesize    = 24
```

```
Config PORTA = Output
Config PORTB = Output
```

```
RED  Alias PORTB.0
YEL  Alias PORTB.1
GRN  Alias PORTB.2
```

Begin:

```
Red = 1 : Yel = 0 : Grn = 0
Wait 1
```

```
Red = 0 : Yel = 1 : Grn = 0
Wait 1
```

```
Red = 0 : Yel = 0 : Grn = 1
Wait 1
```

```
Goto Begin
-----
```

The first section sets up the chip, then we need to configure the two ports. A port can be an INPUT or an OUTPUT. Since we want to run some LEDs, we set the port to be an OUTPUT. May as well define them all at one time... so

we did.

The next section is where we define the pin names. I don't know about you... but I would forget which pin the RED LED was connected to, or the green, or the yellow. I don't feel like typing in **PORTB.0** for the first pin every time... so we told the software that it's name was "RED". Now all we need to do is reference it by it's name.

Once defined, if we make them equal a "1" the LED will turn ON, and if we make it equal to a "0" it would turn OFF. The next series of lines defines how we want the LEDs to be set, then waits 1 second. (The **WAIT** command.)

After we change the state of the LEDs 3 times... we jump back to the beginning and do it all over again... over and over.

**Image #6:** To get the software into the chip we must first **COMPILE** it into something it understands. Clicking on the black chip will run the compiler... this makes a HEX file that can be loaded into the chip. If there are any errors they will be shown at the bottom of the screen and you will need to correct them.

**Image #7 :** When you click the green chip, the programmer opens up. If the chip is connected properly, the programmer screen will display. If not, it will say that it can't find chip FFFFFFFF and you will need to correct the problem.

**Image #8:** Once you get the programming screen to show up, simply click the green chip on that display and the program will be loaded into your chip... once finished, your chip will start running your program. You can disconnect the PC or Laptop and your chip will run your program all by it's-self.

## **Step 8: It's Alive!**

At this point your microprocessor will be running the LEDs in sequence...

Red --> Yellow ---> Green ---> Red...

They will each light for 1 second. If you change this...

**WAIT 1**

to this...



## WAITMS 100

Then it will blink every 100 milliseconds (1/10 of a second.)

Can you change the code to make a traffic light? (*Hint, change the time settings.* )

With the addition of only 2 more parts we were able to have an LCD screen running. You can see some images of that on my [BLOG page](#) .

Stay tuned for more information, downloads, and videos. If you like... visit me at <http://askjerry.info> to see my tutorials, blogs, and more.

Jerry

Just in...

I got with Mark Alberts (author of BASCOM) and asked him which USB programmers he recommended for use with BASCOM.

Here are his favorites.

- 1) [Atmel AVRISP mkII In-System Programmer \(ATAVRISP2\)](#)
- 2) [USBasp - USB programmer for Atmel AVR controllers](#)

So get one and dive in!

[Add Tip](#)[Ask Question](#)[Comment](#)[Download](#)

Participated in the  
[Microcontroller Contest](#)  
[View Contest](#)

**Be the First to Share**



BASCOM und Arduino UNO <https://bascomforum.de/index.php?thread/65-arduino-uno-mit-bascom-%C3%BCber-bootloader-nutzen/>

## • Arduino Uno mit Bascom über Bootloader nutzen

20. Dezember 2016+2

Das hier soll kein "BASCOM ist besser als Arduino" oder umgekehrt werden sondern einfach nur zeigen, wie man einen Arduino UNO mit Bascom nutzen kann. Gerade vielleicht für Leute interessant die Bascom einfach einmal probieren wollen ohne grossartig zu investieren. So ein UNO R3 mit gesockeltem Atmega328P und ATmega16U2 als USB wandler bekommt man ab ca 6 EUR, ein Programmer ist nicht notwendig.

Vorteil ist auch das man sich nicht mit den Fuses beschäftigen muss, die kann man über den Bootloader nicht ändern.

Den Atmega328P auf dem UNO bekommt man mit der Beschränkung auf 4K Programm der Demo nicht voll.

Getestet mit der aktuellen Bascom Demoversion 2.0.7.5 , einem frisch erworbenem UNO und frischem Windows7.

Zuerst Installieren wir Bascom und die Arduino Programmierungsumgebung von hier: [arduino.cc/en/Main/Software](http://arduino.cc/en/Main/Software) .

Ich empfehle die: Windows ZIP file for non admin install, gebraucht wird eigentlich nur der Unterodner drivers.

Zip entpacken reicht bei der non admin Version.

Jetzt erst Arduino UNO am PC anschliessen,  
Windows wird einen Treiber suchen und keinen finden,  
Treiber dann manuell installieren, liegen unter ...\\arduino-1.6.13\\drivers .

Im Gerätemanager nachschauen unter welchem COM port sich der UNO installiert hat, den COM port merken.

Auf dem UNO ist standardmässig ein Blinkprogramm installiert, die LED mit der Bezeichnung L blinkt im Sekundentakt sobald Strom (z.b.per USB) anliegt.

3. Bascom starten, unter **Options\\programmer** braucht der UNO dies hier:

Nun ein kleines testprogramm, kopieren und kompilieren.

### BASCOM-Quellcode

1. `$regfile = "m328pdef.dat"`
2. `$crystal = 16000000`
3. `$hwstack = 40`

```

4.    $swstack = 16
5.    $framesize = 32
6.    baud = 4800
7.
8.    Config Portb.5 = output 'Konfigurieren des Ausgangs der Onboard LED
9.    dim Durchlauf as Byte 'Variable fuer Durchlaufe setzen
10.   Durchlauf = 0 'Durchläufe auf null setzen, default ist 255
11.
12.
13.   Print "Bascom Test auf Arduino UNO R3"
14.   Wait 1
15.
16.   DO 'start der schleife
17.   Print "durchlauf Nr "; durchlauf
18.   print "LED an"
19.   portb.5 = 1 'LED einschalten
20.   print "Warten 1 sekunde"
21.   wait 1
22.   print "LED aus"
23.   portb.5 = 0 'LED ausschalten
24.   wait 2
25.   print "Warten 2 sekunden"
26.   incr durchlauf
27.   loop 'ende der Schleife, zurueck zum anfang der Schleife

```

Alles anzeigen

Nun sollte sich das Programm mit der Brennfunktion von Bascom Brennen lassen.

Erster Unterschied ist das die LED jetzt nun 1 Sekunde an ist und 2 Sekunden aus bleibt.

Zweiter Test ist die serielle Ausgabe der print Befehle.

Dazu in Bascom den Terminal

Emulator starten,

den gemerkten COM Port vom Gerätemanager einstellen und

die serielle Geschwindigkeit passend zum Programm (4800) einstellen:

Jetzt sollte der UNO auch mit uns kommunizieren:

Somit einfach und günstig das erste "Hello World" mit Bascom produziert.

Eigentlich macht sich sowas besser im Wiki, da kann man sowas bestimmt noch genauer erklären. Was nicht ist kann noch werden, hab noch ähnliches für den Mega2560, der ist ein bisschen anders.

Wenn thematisch nicht ins Programmerforum passt bitte verschieben.

Tobias

Dieser Beitrag wurde bereits 1 mal editiert, zuletzt von [stefanhamburg](#) (21. Dezember 2016) aus folgendem Grund: BASCOM und Arduino kann man ruhig ausschreiben. Muss nicht mit B... und A... abgekürzt werden. stefanhamburg und daja gefällt das.

28.

- [stefanhamburg](#)



#### BASCOM-Fan

[Erhaltene Likes](#)

118

[Beiträge](#)

1.139

[Lexikon Einträge](#)

2

[Karteneintrag](#)

ja

1. [2](#)

20. Dezember 2016

Super Beitrag, Tobias!

#### [Schraubbaer schrieb:](#)

Eigentlich macht sich sowas besser im Wiki

Damit das nicht untergeht und weil ich es sehr sehr gut finde, habe ich mir erlaubt, das für Dich ins Wiki zu stellen.

[hier als Wiki/Lexikon-Eintrag](#)

Danke noch einmal. Solche Beiträge brauchen wir!

- [mac5150](#)

#### Tüftler

[Erhaltene Likes](#)

7

[Beiträge](#)

463

[Karteneintrag](#)

ja

1. [3](#)

21. Dezember 2016

Drei Dumme - ein Gedanke...

Die Arduino-Clones sind mittlerweile richtig preiswert zu bekommen.

Und sooo klein!

Wer den Platz des Bootloaders dennoch braucht, kann den via ISP und einem anderen Programmer überschreiben.

Dann ist aber kein Zugriff via Arduino-Programmer mehr möglich!

#### [Heisenberg bei einer Radarkontrolle:](#)

*Polizist: "Wissen Sie, wie schnell Sie waren?"*

*Heisenberg: "Nein. Aber ich weiß genau, wo ich jetzt bin!"*

Dieser Beitrag wurde bereits 1 mal editiert, zuletzt von [stefanhamburg](#) (21. Dezember 2016) aus folgendem Grund: Arduino kann man ruhig ausschreiben. Muss man nicht mit A... abkürzen.

- [Schraubbaer](#)



## Inventar

[Erhaltene Likes](#)

67

[Beiträge](#)

750

[Lexikon Einträge](#)

5

[Karteneintrag](#)

ja

1.

21. Dezember 2016

Na sowas mach ich gerne, ich weiss das grad der Anfang das schwerste ist. Mit frust was anfangen mag keiner. Hab mit der Bootloadergeschichte noch so einiges in der Pipeline, dauert aber noch etwas bis ich zeit finde und die neue Testhardware da ist.

Tobias

- [Mechanic](#)

## Benutzer

[Erhaltene Likes](#)

7

[Beiträge](#)

91

1.

7. Mai 2017

Ich habe gerade einen China-Clone des Uno benutzt. Der hatte auch den Arduino-Bootloader, man musste ihn aber mit 57600 Baud ansprechen.

- [mac5150](#)

## Tüftler

[Erhaltene Likes](#)

7

[Beiträge](#)

463

[Karteneintrag](#)

ja

1.

7. Mai 2017

### Mechanic schrieb:

Ich habe gerade einen China-Clone des Uno benutzt. Der hatte auch den Arduino-Bootloader, man musste ihn aber mit 57600 Baud ansprechen.

Ja, in der Übertragungsgeschwindigkeit differieren die CN-Clones häufig. Kleines Problemchen im Vergleich mit der Ersparnis.

### **Heisenberg bei einer Radarkontrolle:**

*Polizist: "Wissen Sie, wie schnell Sie waren?"*

*Heisenberg: "Nein. Aber ich weiß genau, wo ich jetzt bin!"*

- [mac5150](#)

## Tüftler

[Erhaltene Likes](#)

7

[Beiträge](#)

[Karteneintrag](#)

ja

1. **7**[15. September 2017](#)

Wie geht das mit den Mega 2560 Arduinos?

Ich habe ein "einwandfrei funktionierendes Gerät" auf Com3 - nur Bascom will nicht mal den µC identifizieren.

Alle Baudraten gecheckt.

**Heisenberg bei einer Radarkontrolle:****Polizist: "Wissen Sie, wie schnell Sie waren?"****Heisenberg: "Nein. Aber ich weiß genau, wo ich jetzt bin!"**• [tschoeatsch](#)**Meister Hora**[Erhaltene Likes](#)

468

[Beiträge](#)

5.882

[Lexikon Einträge](#)

3

[Karteneintrag](#)

ja

1. **8**[15. September 2017](#)

Ich hab' die Erfahrung gemacht, wenn das nicht auf Anhieb geht, bascom mal beenden und neu starten, dann ging es mit einer Einstellung, die beim Probieren auch schon mal verwendet wurde. Also immer bascom beenden und nach dem Neustart die Einstellung ändern und testen, oder einstellen - beenden - Neustart und testen. Zuviel verändern bringt was durcheinander.

Raum für Notizen

-----

-----

• [mac5150](#)**Tüftler**[Erhaltene Likes](#)

7

[Beiträge](#)

463

[Karteneintrag](#)

ja

1. **9**[15. September 2017](#)

Hallo Mike,

ich war artig, nicht hektisch - quasi die Ruhe selbst. Ooommmmmmm.....  
Nochmals das Gerät neu installiert:

Sieht gut aus. Warum jetzt der COM-Port von 3 auf 4 gewechselt hat - I don't know.

In Bascom als Programmierer "ARDUINO" ausgewählt und sämtliche Baudraten getestet.  
Mit Bascom-Neustarts.

Ergebnis:

Ich bin ziemlich ratlos.

LG

Mathias

**Heisenberg bei einer Radarkontrolle:***Polizist: "Wissen Sie, wie schnell Sie waren?"**Heisenberg: "Nein. Aber ich weiß genau, wo ich jetzt bin!"*

- [mac5150](#)

**Tüftler**[Erhaltene Likes](#)

7

[Beiträge](#)

463

[Karteneintrag](#)

ja

1. **10**[16. September 2017](#)

Fehler gefunden:

Nicht "ARDUINO" sondern "ARDUINO STK500V2" als Programmer auswählen

Den COM-Port einstellen und Baudrate 115200.

Funktioniert.

**Heisenberg bei einer Radarkontrolle:***Polizist: "Wissen Sie, wie schnell Sie waren?"**Heisenberg: "Nein. Aber ich weiß genau, wo ich jetzt bin!"*

- [klaru](#)

**Neuer Benutzer**[Beiträge](#)

9

1. **11**[21. September 2017](#)

Hallo mac5150

USBasp Programmer geht bei mir ebenfalls.

Nutze den z.B. für die 328p Billig-Arduino pro mini

Grüße, Klaus

- [tschoeatsch](#)

**Meister Hora**

ja

1. **12**[22. September 2017](#)

Jetzt hab' ich mal eine Frage: wenn ich einen Arduino mit bootloader habe, die bauds durch probieren herausgebracht habe, kann ich jetzt an dessen Rxd und Txd ein entsprechen konfiguriertes bluetooth modul anflanschen und mein Programm einfach darüber in den Kontroller kriegen? Die Verbindungen zum Usb-Teil müsste evtl, gekappt werden?

Raum für Notizen

-----

-----

- [DIVX](#)

**Benutzer**[Erhaltene Likes](#)

9

[Beiträge](#)

95

1. **13**

22. September 2017+1  
Meinst Du das zb.

**Arduino flashen über Bluetooth**

[ardumower.de/index.php/de/foru...ueber-bluetooth-anleitung](http://ardumower.de/index.php/de/foru...ueber-bluetooth-anleitung)

tschoeatsch gefällt das.

- [tschoeatsch](#)

**Meister Hora**

[Erhaltene Likes](#)

468

[Beiträge](#)

5.882

[Lexikon Einträge](#)

3

[Karteneintrag](#)

ja

1. **14**

22. September 2017  
Schaut gut aus, danke!  
Raum für Notizen

-----  
-----

- [six1](#)

**Team**

[Erhaltene Likes](#)

22. September 2017

**[tschoeatsch schrieb:](#)**

Jetzt hab' ich mal eine Frage: wenn ich einen Arduino mit bootloader habe, die bauds durch probieren herausgebracht habe, kann ich jetzt an dessen Rxd und Txd ein entsprechen konfiguriertes bluetooth modul anflanschen und mein Programm einfach darüber in den Kontroller kriegen? Die Verbindungen zum Usb-Teil müsste evtl, gekappt werden?

Im Prinzip ja... ABER, wenn auch nur ein Byte bei der Übertragung falsch oder fehlerhaft ist, war es das...

Zu Neuflashen musst du den Bootloader anspringen. das kann man per Software im laufenden Programm auslösen. Wenn das System aber nicht mehr arbeitet, durch vorangegangene fehlerhafte übertragung, ist es vorbei.

Ich habe ein neues System immer erst in einen Speicherbaustein reingeschoben und per Prüfsummen gecheckt. Wenn alles ok war, wurde das dann vom externen Speicher (z.B. External Serial Flash) ge-flasht.

Code first, think later - Natural programmer



- [tschoeatsch](#)

**Meister Hora**

ja

1. **16**

22. September 2017

Ist das nicht bei jedem flash-Verfahren so?

Wäre es denkbar, im Programm eine Routine einzubauen, die den flash zu eine Prüfsumme zusammen führt und mit der übertragenen, vorher berechneten vergleicht? Wenn alles passt, kurzes Ledgeblinke, ansonsten Dauerlicht.

Raum für Notizen

---



---

- [six1](#)

Team

ja

1. **17**

22. September 2017

Bei meinem, bisher angewendeten Verfahren, wurde über das laufende System ein neues System ampfangen und in einen ESF Speicher geschrieben. War alles ok, wurde DANACH der Inhalt des ESF in den Prozessor geflasht. (umgebauter Bootloader, der ESF lesen kann)

Code first, think later - Natural programmer



- [tschoeatsch](#)

Meister Hora

[Erhaltene Likes](#)

468

[Beiträge](#)

5.882

[Lexikon Einträge](#)

3

[Karteneintrag](#)

ja

1. **18**

22. September 2017

So wie es wohl bei den kommerziellen Geräten gemacht wird. Wenn man aber die popeligen bootloader verwenden will, dann muss man wohl mit Misserfolgen rechnen. Aber kaputtmachen, kann man ja nix, muss man halt neu flashen.

Raum für Notizen

---



---

- [Michael](#)

Administrator

[Erhaltene Likes](#)

ja

1. **19**

22. September 2017

[tschoeatsch schrieb:](#)

ber kaputtmachen, kann man ja nix, muss man halt neu flashen.

der Controller könnte sich aber aufhängen, mit dem kaputten Programm. Dann kommst du nicht mehr in den Bootloader.

- [tschoeatsch](#)

Meister Hora

ja

22. September 2017

Wie meinst du das mit 'aufhängen'? Kann der bootloader durch das Programm beschädigt werden? Kann das kaputte Programm den reset außer Kraft setzen?

Raum für Notizen

---



---