Will man größere Datenmengen loggen, reicht das EEPROM des Attiny nicht aus. Hier kann die Platine STI 100 von ELV (Artikel-Nr.: 68-07 59 50) helfen: Mit ihrer Hilfe können nämlich Daten vom Mikrocontroller auf einem handelsüblichen USB-Stick gespeichert werden.

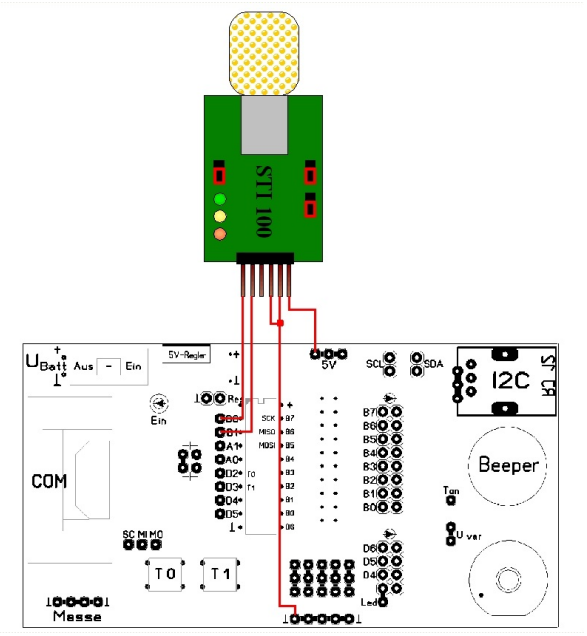

Abbildung 1

Die Handhabung ist relativ einfach: Die STI100-Platine wird mitsamt dem USB-Stick wie in der Abbildung gezeigt an die Attiny-Platine angeschlossen; die drei Jumper werden in Richtung Steckerleiste gesetzt. So wird die STI100-Platine für die Kommunikation über die UART-Schnittstelle (ohne Handshake) konfiguriert. Standardmäßig ist die serielle Schnittstelle des STI100 eingestellt auf 9600 Baud, 1 Startbit, 1 Stoppbit, kein Paritätsbit. Diese Einstellung hat auch unser Attiny, wenn in BASCOM der Befehl $baud = 9600 eingegeben wird.

Die STI100-Platine arbeitet zwar mit 3,3 V - Signalen, kommt aber auch ohne Pegelwandler mit den 5 V - Signalen des Attiny zurecht.

Die Befehle zum Umgang mit dem USB-Stick ähneln den DOS-Kommandos. Hier eine kleine Auswahl:

ASCII-Befehl	Hexadezimaler Befehl	Beschreibung	Rückgabe
<cr>	$0D	Ist ein USB-Stick angeschlossen?	Nein: “No Disk”<cr> Ja: “D:\>”<cr>
“DIR”<cr>	$01, $0D	Verzeichnisstruktur anzeigen	Ordner- und Dateinamen
“CD” <sp><Name><cr>	$02, $20,<Name>, $0D	Verzeichniswechsel	“D:\>”<cr>
“WRF”<sp><#Bytes><cr> <Datenbytes...>	$08, $20, <#Bytes>, $0D	Schreibt die Datenbytes an das Ende der geöffneten Datei	“D:\>”<cr>
“OPW”<sp><Name><cr>	$09, $20, <Name>, $0D	Öffnet/Erstellt eine Datei zum Schreiben	“D:\>”<cr>
“CLF”<sp><Name><cr>	$0a, $20, <Name>, $0D	Schließt die Datei	“D:\>”<cr>

Die Befehle können auf zwei verschiedene Arten an die STI100-Platine gegeben werden: Einmal über ASCII-Befehle und einmal über hexadezimale Befehle. Standardmäßig ist der STI 100 auf ASCII-Befehle eingestellt. Eine Umstellung erfolgt mit dem SCS-Befehl. Die hexadezimalen Befehle sparen etwas Speicherplatz und sind im Umgang mit dem Mikrocontroller etwas einfacher (So vermeiden sie Probleme mit dem Literalzeichen "\".); aber sie sind weniger gut einprägsam. Deswegen soll in unserem einführenden Beispiel der ASCII-Befehlssatz benutzt werden.

Unser Mikrocontroller soll 10 Temperaturwerte über einen LM75-Baustein messen und auf dem USB-Stick in einer Datei namens tempera5.txt im Stammverzeichnis abspeichern. Zunächst laden wir (bei abgetrenntem STI100; dazu +5V-Verbindung kappen) das Programm **`temperatur2.bas`** (s. u.) auf den Attiny. Dann schalten wir die Attiny-Platine aus, stellen die +5V-Verbindung zur STI 100-Platine wieder her und schalten die Attiny-Platine wieder an. Die STI100-Platine sendet im Rahmen des Initialisierungsvorgangs mehrere Zeichenketten. Dies dauert manchmal einige Sekunden. Wenn alles in Ordnung ist, sendet sie als letzte Zeichenkette das Prompt-Zeichen "D:\>". (Einer meiner USB-Sticks bereitete Probleme: In diesem Fall war die letzte Rückmeldung "No Disk". Solche Meldungen ergaben sich auch bei anderen USB-Sticks, als die Batteriespannung zu niedrig war.)

Unser Attiny-Programm überprüft bei der Initialisierung ankommenden Zeichenketten: Erst wenn das letzte Zeichen der Zeichenkette ein ">" ist, dann wird die Datei "tempera5.txt" geöffnet und das eigentliche Loggen kann beginnen. Die Einzelheiten entnimmt man dem beigefügten Sourcecode.

Hier nur drei Bemerkungen:

1. Die STI100-Befehle werden mit einem <cr> (Ascii-Code 13) abgeschlossen. Der BASCOM-Befehl print schließt gewöhnlich die zu sendende Zeichenkette mit einem <cr><lf> ab. Das zusätzliche LineFeed-Zeichen würde die Kommunikation stören. Deswegen werden alle Print-Befehle mit einem Semikolon beendet; dadurch werden die <cr><lf>-Steuerzeichen unterdrückt. Die <cr>-Zeichen werden dann über printbin cr gesendet.

2. Beim WRF-Befehl muss die Anzahl der Bytes unserer Temperaturangabe stehen. Bei uns sind das 3 Bytes: 2 Bytes für den Temperaturwert selbst, der mit dem Print-Befehl als Zeichenkette gesendet wird, und 1 Byte für das Leerzeichen, welches hier als Abgrenzungszeichen (Delimiter) eingesetzt wird. Die STI 100 - Platine erwartet für diese Angabe eine Zahl aus 4 Bytes; die 3 führenden Bytes sind dann 0 und das letzte ist 3.

3. Wenn man der Datei die Extension "csv" gibt, dann kann sie auch mit Tabellenkalkulationsprogrammen (z. B. Open Office.org Calc) weiter bearbeitet werden. Für diesen Fall empfiehlt sich als Delimiter ein Semikolon (->Zeile> oder ein <cr>-Zeichen (->Spalte).

Quellcode von **`temperatur2.bas`**:

```
' Attiny-Platine von E. Eube, G. Heinrichs und U. Ihlefeldt
' plus I2C-Temperatursensor LM75 plus STI100
' LEDs an PortB.0 (rot), PortB.1 (gelb)
' 10 Temperaturen werden über STI100 auf USB-Stick geschrieben
' STI100 über TTL-COM/USART (verträgt 5 V)
' Speichern der Daten im ASCII-Format durch Leerzeichen getrennt
'
'-------------------------------------------------------------------------

$regfile = "attiny2313.dat"                 'Attiny2313
$crystal = 4000000                          '4 MHz
$baud = 9600

'*********************************************************
'****************** Deklarationen ************************

Dim Temperatur As Byte
Dim Adresse As Byte
Dim Dateiname As String * 20
Dim Antwort As String * 20
Dim I As Byte
Const Prompt = ">"                          'Der vollständige Prompt D:\> ist wegen des
                                            \-Zeichens problematisch
Const Cr = 13                               'CR-Steuerzeichen
Declare Sub Messung
Declare Sub Usb_init
Declare Sub Wert_speichern
Declare Sub Schluss

'****************** Initialisierung **********************

Ddrb = &B11111111                           'Port B als Ausgangsport
Ddrd = &B01110000                           'D4, D5, D6 als Ausgang; Rest als Eingang
Portd = &B10001111                          'Eingänge auf high legen

Config Scl = Portb.7                        'Konfigurieren von I2C
Config Sda = Portb.5

Adresse = 153                               'Adresse des LM75 (kann auch anders sein!)
Dateiname = "tempera5.txt"                  'keine überlangen Dateinamen
```

```
'*********************************************************
'******************** Hauptprogramm ************************

Call Usb_init
If Antwort = Prompt Then
  Portb.0 = 1                          'rote LED an: jetzt USB-Stick nicht entfernen
  For I = 1 To 10
    Portb.1 = 1                        'gelbe Led blinkt während Messung
    Call Messung
    Call Wert_speichern
    Portb.1 = 0
    Wait 2
  Next I
  Call Schluss
  Portb.0 = 0                          'rote LED aus; USB-Stick kann entfernt werden
End If
End

'*********************************************************
'******************** Unterprogramme ************************

Sub Messung
  I2cstart
  I2cwbyte Adresse
  I2crbyte Temperatur , Nack           'Kein Acknowledge
  I2cstop
  Waitms 300
End Sub

Sub Usb_init
  Do
    Input Antwort Noecho
    Antwort = Right(antwort , 1)       'wegen Problemen mit \ nur auf > testen
  Loop Until Antwort = Prompt          'vierte Antwort sollte Prompt sein

  Portb.2 = 1                          'USB-Stick eingeloggt
  Waitms 100
  Portb.2 = 0
  Wait 1

  Printbin Cr                          'Puffer bei sti100 leeren; denn: ggf. sendet Attiny
                                       am Anfang ein sinnloses Zeichen. Kann entfernt
                                       werden, wenn STI100 nach dem Attiny einge-
```

```
                                    schaltet wird.
  Input Antwort Noecho

  Print "OPW " ; Dateiname;         'semikolon verhindert CR LF
  Printbin Cr                       'CarriageReturn ohne LineFeed
  Input Antwort Noecho
  Antwort = Right(antwort , 1)
  If Antwort = Prompt Then          'Datei geöffnet
    Portb.1 = 1
    Waitms 100
    Portb.1 = 0
  End If
End Sub

Sub Wert_speichern
  Print "WRF " ;
  Printbin 0
  Printbin 0
  Printbin 0
  Printbin 3                        '2 Ziffern und 1 Leerzeichen-> 3 Bytes
  Printbin Cr
  Print Temperatur ; " ";
  Input Antwort Noecho
End Sub

Sub Schluss
  Print "CLF " ; Dateiname;
  Printbin Cr
  Input Antwort Noecho
End Sub
```