Ein USB-Speicherstick für den Tiger

Gunther Zielosko

1. Grundlagen

Eine häufig auftretende Aufgabe für Tiger-Nutzer ist es, mit dem Tiger Daten zu sammeln, zu erzeugen oder zu verarbeiten und diese dann unkompliziert auf einem Massenspeicher abzulegen bzw. dort abzurufen. Es gibt mittlerweile komfortable Lösungen mit der SecureDigital- oder kurz SD-Card (früher Smart-Media-Card), bei diesen findet die gesamte Verwaltung des Dateisystems im BASIC-Tiger® statt. In diesem Applikationsbericht stellen wir eine Lösung für die Datenspeicherung mit einem ganz normalen USB-Speicherstick vor, der praktisch von jedem Computersystem gelesen und beschrieben werden kann und die zudem die Tiger-CPU entlastet.
Verwendet wird dabei ein USB-Interface mit einem speziell dafür geschaffenen Chip, das alle Aufgaben eines USB-Host übernimmt und damit jeden normalen USB-Speicherstick bedienen kann. Der angeschlossene Controller, in unserem Fall der BASIC-Tiger®, steuert seinerseits das Interface über die vorhandene serielle Schnittstelle mit vergleichsweise einfachen Befehlen. Wird beispielsweise über die serielle Schnittstelle der simple Befehl „DIR“ (in normaler ASCII-Schreibweise) gesendet, schickt das Interface eine Liste der auf dem Speicherstick vorhandenen Dateien zurück (natürlich wieder in normaler ASCII-Schreibweise). Toll – nicht wahr? Das System, das wir kennenlernen wollen, erinnert in vielen Teilen an das gute alte DOS...

2. Das USB-Stick-Interface STI 100 von ELV

Das Interface ist 64x15x35mm groß, funktioniert mit 5V (stabilisiert) und braucht etwa 20 bzw. ca. 150 mA Strom (Standby bzw. mit aktivem Speicherstick). Zwar sind die Ein- bzw. Ausgangs-Pegel auf 3,3V festgelegt, der Tiger kann aber diesen Pegel noch als High interpretieren und die Eingänge des STI 100 tolerieren ihrerseits 5V-Pegel.

Bild 1 STI 100 – Bauelementeseite

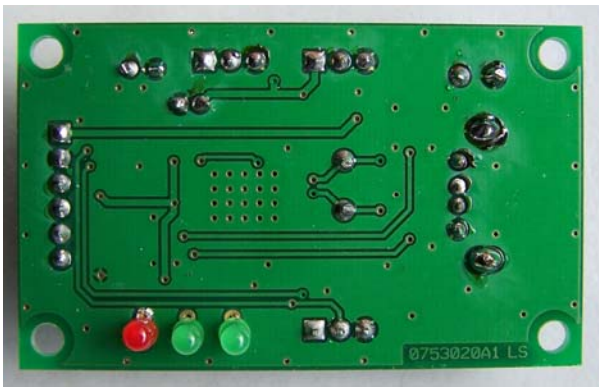

Bild 2 STI 100 – „Leiterseite“

Der komplette Bausatz (alle SMD-Bauteile bereits eingelötet) ist als „Komplettbausatz STI 100“ für 37,95 € unter der ELV-Bestell-Nr. 76-759-50 zu beziehen, die Internet-Adresse ist:
www.elv.de

Wie immer bei ELV liegt auch hier eine ausführliche Beschreibung bei, diesmal eine Kopie der Artikel aus dem ELV-Journal (Hefte 5 und 6/2007).

Das Herz des Interfaces ist ein spezieller USB-Host-Controller von FTDI, der VNC1L „Vinculum“. FTDI ist der Spezialist für USB-Chips, wir erinnern uns an die Applikationsberichte 052 und 074. In beiden Fällen wurden FTDI-Controller benutzt, hier allerdings lediglich in der Slave-Funktion. Die Host-Funktion ist neu für den BASIC-Tiger®. Eine ausführliche Beschreibung der Firmware findet man unter:
www.vinculum.com/documents.html#vfwspecs

3. Nachbau und Inbetriebnahme

Hier werden wir nur wenig Worte verlieren, alles ist in der Dokumentation ausführlich erklärt. Allerdings hat der Autor abweichend von der Beschreibung das Einbaukonzept etwas verändert. Der vorgesehene 6-polige Winkelstecker wurde gegen ein gerades Exemplar ausgetauscht und die LEDs auf der Unterseite angeordnet. So kann die Interface-Leiterplatte mit der „Leiterseite“ nach oben in Buchsen auf der Hauptleiterplatte mit dem BASIC-Tiger® gesteckt werden, wodurch auf letzterer Platz für weitere Komponenten wird. Weiterhin wird weniger Fläche für das Interface gebraucht und der Befestigungsaufwand wird geringer. Die Bilder 1 und 2 zeigen diese Aufbauvariante.

4. Starthilfe mit PC und einer Pegelwandlerschaltung

Das STI 100 ist für TTL-Pegel ausgelegt und somit nicht direkt an einen PC anschließbar. Trotzdem ist zu empfehlen, eine kleine Schaltung aufzubauen, mit der erste Experimente mit dem STI 100 am Computer durchgeführt werden können. Mit dieser kleinen Schaltung kann man die Funktionen des STI 100 sofort ausprobieren, ohne vorher ein Programm für den BASIC-Tiger schreiben zu müssen:

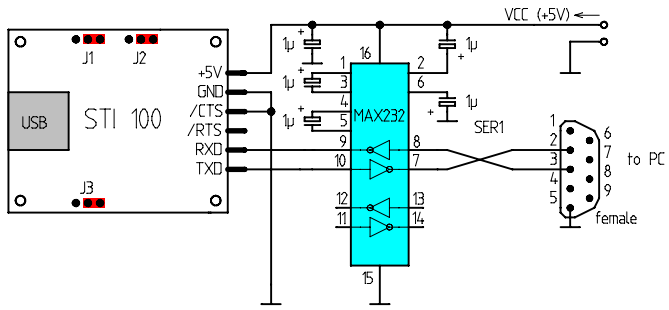

Bild 3 Pegelwandler für den Betrieb am PC

5. Experimente mit Hyperterminal von Windows

Das Programm Hyperterminal führt unter Windows XP eher ein Schattendasein und manche werden es wohl noch nie gesehen oder benutzt haben. Und wenn, es ist alles andere als intuitiv bedienbar und sehr auf Modems und ähnliche System zugeschnitten. Trotzdem ist es für unsere Zwecke jetzt sehr nützlich. Wir werden uns also die Mühe machen und HyperTerminal für uns nutzen.

Rufen Sie also
Start → Programme → Zubehör → Kommunikation → HyperTerminal → Hyperterminal (mit dem gelben Telefon!) auf.

Bild 4 das Fenster zum Einrichten einer neuen Verbindung erscheint

Geben Sie hier einen Namen ein, wie z.B. „Tiger_Stick“ und klicken Sie auf OK.

Bild 5 hier beginnt schon das Verwirrspiel, wir wollen doch nicht telefonieren...

Wenn Sie aber auf das Auswahlfeld „Verbindung herstellen über:“ und dort auf den Pfeil rechts kommen, werden weitere Optionen angezeigt:

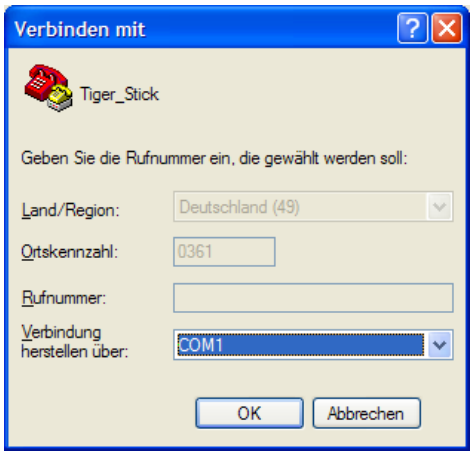

Bild 6 jetzt tauchen die für uns wichtigen COM-Verbindungen auf, hier z.B. COM1

Nun erscheint ein neues Fenster „Eigenschaften von COM1“.

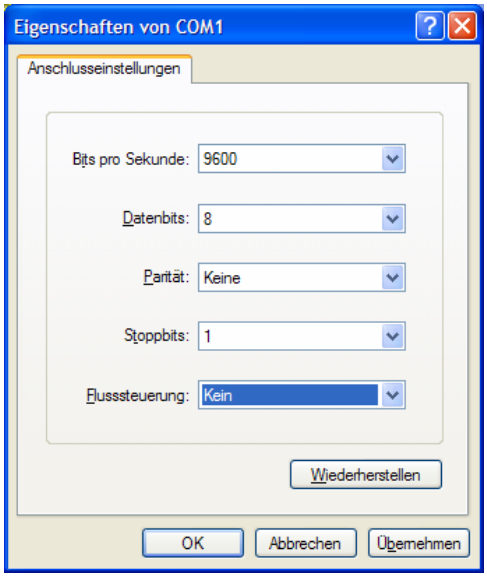

Bild 7 die gewählte COM-Schnittstelle wird konfiguriert.

Für’s erste wählen wir hier für die Flußsteuerung „Kein“. Klicken Sie auf „Übernehmen“ und „OK“.

Gehen Sie nun auf „Datei“ → „Eigenschaften“ → „Einstellungen“ und aktivieren Sie folgende Felder.

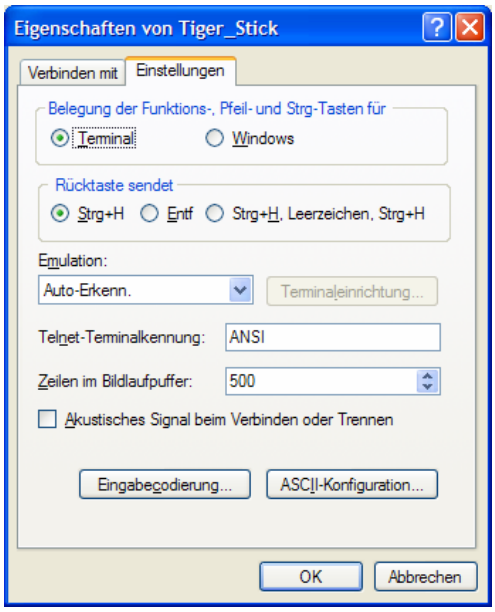

Bild 8 Das Fenster „Einstellungen“

Wählen Sie nun unter „ASCII-Konfiguration“ die folgenden Einstellungen:

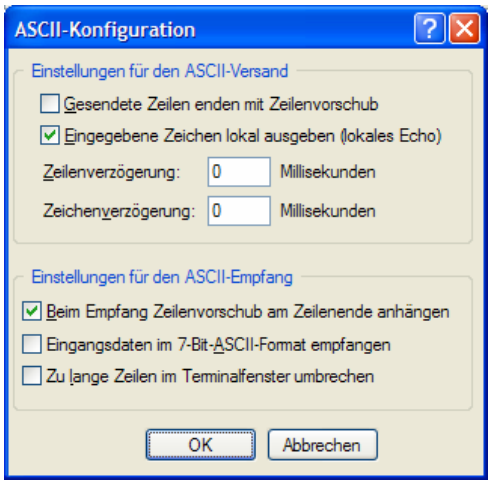

Bild 9 die „ASCII-Konfiguration“ als letzter Schritt

Sie sind nun fertig und können das Ergebnis (eine sogenannte „Verbindung“, die jetzt „Tiger_Stick.ht“ heißt) mit „Datei“ → „Speichern unter“ irgendwo abspeichern, am besten auf dem Desktop, da man es hier am leichtesten finden kann.

Achtung! Stecken Sie vorerst noch keinen Speicherstick in die USB-Buchse, insbesondere keinen mit wichtigen Daten. Der Stick bzw. Ihre Daten könnten bei fehlerhafter Funktion oder irrtümlich eingegebenen Befehlen beschädigt oder zerstört werden. Optimal ist für die nächsten Experimente ein billiger, neuer Speicherstick, den man überall für weniger als 10 € kaufen kann.

Wenn Sie sich nun einen RS232-Adapter gemäß Bild 3 aufgebaut haben, steht einem Funktionstest nichts mehr im Weg. Das Interface STI 100 ist fertig aufgebaut und optisch auf Fehler kontrolliert. Nach dem Anlegen einer stabilisierten Spannung von +5V sollte zunächst die rote LED D3 aufleuchten und danach im Wechsel die LED's D1 und D2 ein paarmal grün blinken. Danach trennen Sie wieder die Betriebsspannung ab. Schließen Sie dann das zusätzliche RS232-Interface an und verbinden es mit dem Computer an der COM-Schnittstelle, die Sie mit HyperTerminal ausgewählt haben.

Starten Sie nun HyperTerminal, d.h. Ihre neue Verbindung „Tiger_Stick.ht".

Wenn Sie nun Betriebsspannung an die beiden verbundenen Interface-Schaltungen legen, sollte Ihr HyperTerminalfenster nach etwa 2 Sekunden so aussehen:

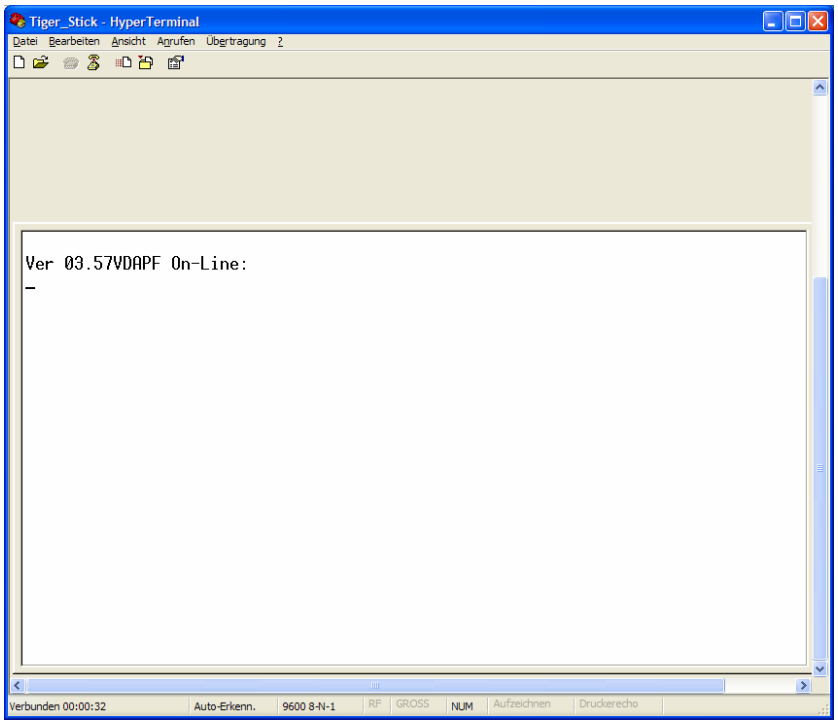

Bild 10 STI 100, das RS232-Interface und HyperTerminal arbeiten korrekt...

Drücken Sie nun die „ENTER"-Taste, das Resultat ist „No Disk". Klar, es ist kein Speicherstick eingesteckt. Das tun wir jetzt und die Antwort ist „Device Detected P2". Als nächstes tippen Sie „IDD" ein und drücken die „Enter"-Taste, die Antwort könnte wie folgt aussehen:

USB VID = $090C
USB PID = $1000
Vendor Id = SMI
Product Id = USB DISK
Revision Level = 1100
I/F = SCSI
FAT16

Bytes/Sector = $0200
Bytes/Cluster = $004000
Capacity = $3BFB8000 Bytes
Free Space = $3BFB4000 Bytes

Der Befehl „IDD“ gibt Informationen zum Speicherstick aus, die natürlich je nach Gerät unterschiedlich sein können:

USB VID = $0C76 USB PID = $0007 Vendor Id = TrekStor Product Id = USB-MiniStick Revision Level = 1.00 I/F = SCSI FAT16 Bytes/Sector = $0200 Bytes/Cluster = $000800 Capacity = $07C9A000 Bytes Free Space = $02	USB VID = $090C USB PID = $1000 Vendor Id = SMI Product Id = USB DISK Revision Level = 1100 I/F = SCSI FAT16 Bytes/Sector = $0200 Bytes/Cluster = $004000 Capacity = $3BFB8000 Bytes Free Space = $3BFB4000 Bytes	USB VID = $0204 USB PID = $1976 Vendor Id = USB 2.0 Product Id = Flash Disk Revision Level = 5.00 I/F = SCSI FAT32 Bytes/Sector = $0200 Bytes/Cluster = $001000 Capacity = $F9824000 Bytes Free Space = $F47E0000 Bytes
130.654.208 (128MB)	1.006.338.048 (1GB)	4.186.062.848 (4GB)

Tabelle 1 einige getestete Speichersticks

Falls Sie schon einige Dateien bzw. Ordner auf Ihren Stick haben, können Sie den Befehl „DIR“ ausprobieren (immer mit „Enter“ abschließen!).

Je nach Füllzustand Ihres Speichersticks kann das Auslesen der Verzeichnisse und Dateien schon einige Zeit in Anspruch nehmen, unser Interface arbeitet schließlich nur mit einer normalen RS232-Schnittstelle (derzeit mit 9600 Baud!) und nicht mit USB2.0-Geschwindigkeit. Was wir auch noch feststellen ist, daß längere Verzeichnisnamen nicht mehr vollständig gelesen werden. Das Interface verhält sich also ähnlich wie ein DOS-Rechner oder sehr alte Windows-Anwendungen. Für eigene Anwendungen mit dem Tiger müssen wir diese Verhalten also berücksichtigen und sollten z.B. keine langen Dateinamen verwenden, die sich nur im hinteren Teil des Namens unterscheiden (jenseits der 8 Zeichen).

Mit Hilfe der Befehls-Tabelle in der von ELV mitgelieferten Beschreibung bzw. der angepassten Tabelle im Anhang dieses Berichtes können Sie nun nach Herzenslust experimentieren.

6. Das Zusammenspiel mit dem Tiger

Für die nächsten Schritte wollen wir folgendes vereinbaren:

- Wir arbeiten immer ohne Hardware-Handshake und lassen demzufolge das /CTS-Pin des STI 100 auf GND-Potential. Dies gilt für SER1 sowieso und zunächst auch für SER0.
- Wir verwenden für die Kommunikation mit dem STI 100 grundsätzlich den so genannten „Erweiterten ASCII-Befehlssatz“ (siehe Tabelle am Ende des Berichtes).
- Wir ändern an der Default-Einstellung (z.B. UART-Betriebsart, Baudrate 9600 Bd usw.) vorerst nichts.

Die Zusammenschaltung des STI 100 mit einem BASIC-Tiger® ist simpel. STI 100 ermöglicht zwei grundsätzliche Steuermöglichkeiten, nämlich UART- und SPI-Interface. Wir werden uns hier auf die UART-Variante (serielle Schnittstelle) beschränken. Die 6-polige Stiftleiste ST1 beinhaltet dafür alle notwendigen Verbindungen. Je nach Verwendung der seriellen Schnittstellen am BASIC-Tiger® ergeben sich dann folgende Möglichkeiten:

Pin STI 100	Fkt. STI 100	BASIC-Tiger®-Pin SER0	SER1
1	+5V	VCC	VCC
2	GND	GND	GND
3	CTS →GND	CTS0→GND	
4	RTS		
5	RXD	TxD0	TxD1
6	TXD	RxD0x	RxD1

Tabelle 2 Verbindungen zwischen STI 100 und BASIC-Tiger®

Im Normalfall verlangt das Interface STI 100 einen Hardware-Handshake-Betrieb mit CTS und RTS. Das bietet beim BASIC-Tiger® nur die Schnittstelle SER0. Will man SER1 benutzen, ist der CTS-Pin des STI 100 mit GND zu verbinden (ein Widerstand von etwa 2 k gegen Masse reicht aus).

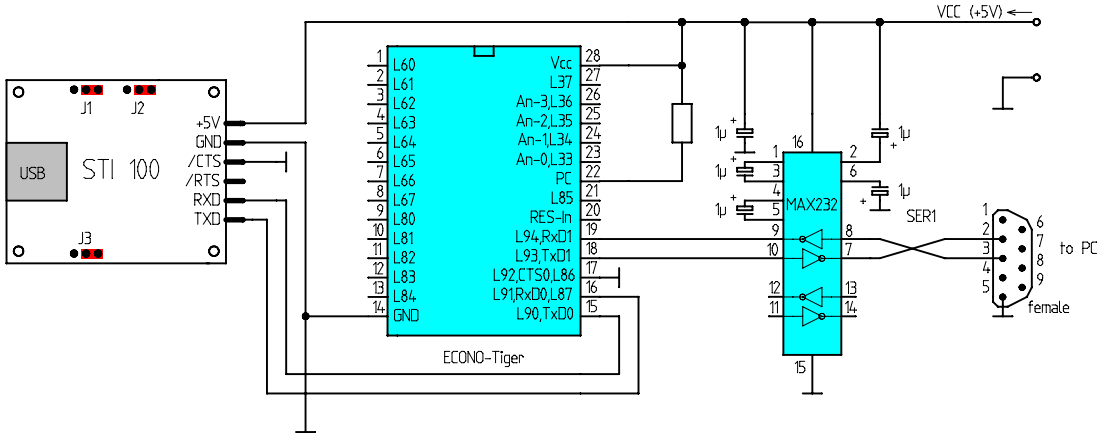

Bild 11 Die Verdrahtung des STI 100 mit einem ECONO-Tiger®.

Die in Bild 11 angeschlossene Pegelwandlerschaltung mit MAX232 ist optional und dient zur Kopplung mit einem PC. Damit wird eine Kontrolle der vom BASIC-Tiger® kommenden Befehle und der vom STI 100 kommenden Antworten via SER1 am PC möglich. Im Demo-Programm „STICK01.TIG“ wird dies ausgenutzt und man kann mit HyperTerminal ganz einfach (übrigens dieselbe Konfiguration wie oben beschrieben) den Datenverkehr beobachten.

Der mit HyperTerminal auf SER1 belauschte Dialog zwischen BASIC-Tiger® und STI 100 beim Durchlaufen des Demo-Programms „STICK01.TIG“ (blau = Befehle vom BASIC-Tiger®, rot = Antwort des STI 100):

```
Ver 03.57VDAPF On-Line:                 Antwort nach dem Einschalten der Betriebsspannung
Device Detected P2                      Speicherstick festgestellt
No Upgrade                              keine Upgrade-Datei darauf entdeckt
D:\>                                    Prompt
IDD                                     Befehl „IDD“ gibt Daten des Speichersticks aus

USB VID = $090C                         USB VID
USB PID = $1000                         USB PID
Vendor Id = SMI                         Vendor ID
Product Id = USB DISK                   Produkt ID
Revision Level = 1100                   Revision Level
I/F = SCSI                              I/F
FAT16                                   FAT16
Bytes/Sector = $0200                    Bytes pro Sektor
Bytes/Cluster = $004000                 Bytes pro Cluster
Capacity = $3BFB8000 Bytes              Gesamtspeicher-Größe
Free Space = $3BFB4000 Bytes            Freier Speicher

D:\>                                    Promt
DIR                                     Befehl „DIR“ gibt die Verzeichnisstruktur zurück

TEST00.TXT                              Es gibt zur Zeit nur eine Datei „TEST00.TXT“
D:\>                                    Prompt
OPR test00.txt                          Befehl „OPR“ (Öffnen der Datei „TEST00.TXT“ zum Lesen)
D:\>                                    Prompt
RD test00.txt                           Befehl „RD“ (Lesen der Datei „TEST00.TXT“)
ABCDEFD:\>                              Der Inhalt der Text-Datei ist z.Zt. „ABCDEF“ -> Prompt
CLF test00.txt                          Befehl „CLF“ (Schließen der Datei „TEST00.TXT“)
D:\>                                    Prompt
OPW test00.txt                          Befehl „OPR“ (Öffnen „TEST00.TXT“ zum Schreiben)
D:\>                                    Prompt
WRF                                     Befehl „WRF“ Anhängen von 8 Bytes an „TEST00.TXT“
  123456D:\>
CLF test00.txt                          Befehl „CLF“ schießt Datei „TEST00.TXT“
D:\>                                    Prompt
OPR test00.txt                          mit „OPR“ Datei „TEST00.TXT“ wieder öffnen
D:\>                                    Prompt
RD test00.txt                           und mit „RD“ Datei „TEST00.TXT“ lesen
ABCDEF  123456D:\>                      an den Text „ABCDEF“ wurde „  123456“ angehängt
CLF test00.txt                          Datei wieder schließen
D:\>                                    Prompt
```

Das Demo-Programm „STICK01.TIG“ zeigt ein paar Befehle, die einige Möglichkeiten des STI 100 vorführen. Ausgangspunkt sei ein neuer USB-Speicherstick, hier z.B. mit 1 GB Kapazität. Dieser wird zunächst in eine USB-Buchse des PC eingesteckt, dort erkannt und als neues Laufwerk eingeführt. Mit dem Editor aus Windows erstellen wir nun eine Datei „TEST00.TXT“ in die wir hier nur 6 Buchstaben „ABCDEF“ eingetippt haben. Nach dem Speichern der Datei können wir den Stick abziehen und auf unser zunächst stromloses Interface stecken, das wie beschrieben mit dem Tiger verbunden ist. Nach Einschalten der Betriebsspannung erfolgt automatisch und ohne Mitwirkung des Tigers (Wartezeit) eine Initialisierung mit den bekannten Meldungen. Diese werden über SER1 zusätzlich ausgegeben. Die dann folgenden Befehle und Reaktionen werden oben ausführlich kommentiert. Die vorhandene Text-Datei wird zunächst zum Lesen geöffnet, gelesen und wieder geschlossen. Dann wird sie zum Schreiben wieder geöffnet und am Ende werden einige Zeichen (123456) angehängt, danach wird sie wieder geschlossen. Zur Kontrolle

erfolgen am Schluß wieder Öffnen zum Lesen, Lesen und Schließen. Am Ergebnis kann man erkennen, daß die Datei nun wirklich zusätzliche Zeichen enthält (ABCDEF 123456).

Beachten Sie bitte, daß STICK01.TIG wirklich nur ein Demo-Programm ist. In einer echten Anwendung muß das Programm unbedingt auf die Reaktion des Interfaces und auf Fehlermeldungen reagieren. So dauert z.B. die Reaktion auf den Befehl „DIR“ unterschiedlich lange, je nach Umfang der Dateien und Verzeichnisse. Dies kann vernünftig nicht durch Wartezeiten allein beherrscht werden.

7. Ein Fazit

Für die BASIC-Tiger®-Familie ist das Interface STI 100 eine echte Bereicherung. Es gibt von Wilke Technology bereits ein SmartMedia-Card- sowie zwei SD-Card-Interfaces, die jedes für sich komfortabel spezielle FAT-basierte Speichermedien verwalten können. Ein USB-Speicherstick bietet jedoch den einen oder anderen Vorteil: Zum einen ist ein USB-Speicherstick heute systemübergreifend überall im Einsatz und zum anderen ohne jeden Kartenleser in allen Standardrechnern zu lesen und zu beschreiben. Zusätzlich erleichtert das von ELV bzw. FTDI verwendete Konzept die Programmierung des Mikro-Controllers erheblich, da man sich nicht um das verwendete FAT-System, die Speicher-Struktur und andere Einzelheiten kümmern muß.

Ein großer Vorteil des STI 100 ist weiterhin, daß keinerlei Controller-bezogene Device-Driver benötigt werden, was den Einsatz auch mit anderen Geräten ermöglicht. Vielleicht kann man nun auch USB-Speichersticks mit speziellen bzw. alten Computern oder Betriebssystemen verwenden, die sonst keinen USB-Betrieb erlauben.

Ein Nachteil hingegen ist, daß das Interface nicht mit solchen Sticks zurechtkommt, die ihrerseits z.B. SD-Karten als Speichermedien aufnehmen (Flash-Card-Reader). Dabei spielt keine Rolle, ob dort z.B. 1GB-SD-Karten (FAT16) oder 4GB-SD-Karten eingesteckt wurden.

Interessant auch im Zusammenhang mit dem vorhergehenden Bericht Nr. 089 ist die Erkenntnis, daß die uralte serielle Schnittstelle zwar schon lange tot gesagt wurde, wegen ihrer übersichtlichen Struktur und ihres Verzichtes auf die Host/Slave-Struktur aber immer wieder zu neuem Leben erwacht...

Am Ende finden Sie nun noch ein paar Tabellen mit wichtigen Befehlen. Die Tabelle wurde im Wesentlichen mit freundlicher Genehmigung der ELV-Redaktion aus dem ELV-Journal entnommen und auf die BASIC-Tiger®-Schreibweisen angepaßt. Zyan gekennzeichnet sind die Befehle und einige Antworten des Interfaces im Erweitertem ASCII-Befehlssatz, den wir hier allgemein benutzt haben.

Viel Erfolg mit den neuen Möglichkeiten!

Achtung!
In der ELV-Veröffentlichung finden sich einige Fehler, auf die in der angehängten Tabelle hingewiesen wird.

Erweiterter ASCII-Befehlssatz	**Hexadezimaler Befehlssatz für Mikrocontroller**	**Beschreibung**	**Rückgabe**
Konfigurationsbefehle			
„SCS“<CR>	10h, 0Dh	In den Mikrocontroller-Befehlssatz umschalten	Gibt“>“<CR> zurück, um anzuzeigen, dass der Baustein nun den Mikrocontroller-Befehlssatz akzeptiert
„ECS“<CR>	11h,0Dh	In den erweiterten ASCII-Befehlssatz umschalten	Gibt“D:>“<CR> zurück, um anzuzeigen, dass der Baustein nun den erweiterter ASCII-Befehlssatz akzeptiert
„IPA“<CR>	90h,0Dh	Zahlendarstellung in ASCII-Zeichen	ECS-Mode und IPA-Mode aktiviert: „WRF“<SP>“10“<CR> 01020304050607080 90A <prompt><CR>
„IPH“<CR>	91h,0Dh	Zahlendarstellung als hexadezimaler Wert	SCS-Mode und IPH-Mode aktiviert: 08 20 00 00 00 0A 0D 01 02 03 04 05 06 07 08 09 0A<prompt><CR>
„E“<CR>	45h,0Dh	Echo-Befehl	Gibt „E“<CR> zurück
„e“<CR>	65h,0Dh	Echo-Befehl	Gibt „E“<CR> zurück

Prüfen, ob ein USB-Gerät angeschlossen ist			
<CR>	0Dh	Ist ein USB-Stick angeschlossen?	Gibt folgende Antwort zurück:
Im erweiterten ASCII-Befehlssatz		Kein Gerät angeschlossen	„No Disk“<CR>
		Gerät angeschlossen	„D:\>“<CR>
Im Mikrocontroller-Befehlssatz		Kein Gerät angeschlossen	„ND“<CR>
		Gerät angeschlossen	„>“<CR>

Verzeichnis-Befehle			
„DIR“<CR>	01h,0Dh	Verzeichnisstruktur ansehen	Gibt eine Liste mit Ordner- und Dateinamen zurück. Jede Zeile wird mit einem <CR> abgeschlossen. Ordner werden mit der Zeichenkette <SP>DIR hinter dem Namen gekennzeichnet.
„DIR“<SP><Name><CR>	01h,20h,<Name>,0Dh	Gibt die Größe der angegebenen Datei zurück. Sollte vor dem Lesen einer Datei verwendet werden, um die Anzahl der Bytes zu kennen, die gelesen werden.	<CR><Name><SP>Größe (4 Byte, LSB first)><CR>
„DLD“<SP>Name<CR>	05h,20h,<Name>,0Dh	Verzeichnis löschen	Gibt <promt><CR> zurück
„MKD“<SP><Name> <CR>	06h,20h,<Name>,0Dh	Verzeichnis erstellen	Gibt <promt><CR> zurück
„MKD“<SP><Name> <SP><Datum><CR>	06h,20h,<Name>,20h, <Datum>,0Dh	Verzeichnis mit Erstellungsdatum erstellen	Gibt <promt><CR> zurück
„CD“<SP><Name><CR	02h,20h,<Name>,0Dh	Verzeichniswechsel	Gibt <promt><CR> zurück
„CD“<SP>“..“<CR>	02h,20h,2Eh,2Eh,0Dh	Eine Verzeichnisebene nach oben	Gibt <promt><CR> zurück

Hinweis!
In den Befehlen „MKD“<SP><Name><CR> und „MKD“<SP><Name><SP><Datum><CR> sind in der ELV-Veröffentlichung Fehler enthalten. Die Befehle beginnen im Hexadezimalen Befehlssatz für Mikrocontroller beide mit 06h statt wie bei ELV mit 07h. Dieser Fehler kann fatale Folgen für eine Datei oder ein Verzeichnis haben, da 07h eigentlich eine Datei löschen würde (Befehl 07h steht für Datei löschen!)

Datei-Befehle			
„RD“<SP><Name><CR>	04h,20h,<Name>,0Dh	Datei lesen	Gibt die gesamte Datei byteweise zurück. <Daten><prompt><CR> Die Länge kann vorher mit dem Befehl „DIR“<SP><Name> <CR> ermittelt werden
„RDF“<SP><size in hex (4 bytes)><CR>	0Bh,20h,<size in hex (4 bytes)>,0Dh	Liest angegebene Anzahl an Bytes der momentan geöffneten Datei	Gibt die angegebene Anzahl an Bytes zurück <Daten><promt><CR>
„DLF“<SP><Name> <CR>	07h,20h,<Name>,0Dh	Datei löschen	Gibt <promt><CR> zurück
„WRF“<SP><size in hex (4 bytes)><CR> <data bytes of size>	08h,20h, <size in hex (4 bytes)>,0Dh,datah	Schreibt die angegebenen Daten ans Ende der geöffneten Datei	Gibt <promt><CR> zurück
„OPW“<SP><Name> <CR>	09h,20h,<Name>,0Dh	Öffnet eine Datei zum Schreiben mit „WRF“	Gibt <promt><CR> zurück
„OPW“<SP><Name> <SP><Datum><CR>	09h,20h,<Name,20h, <Datum>,0Dh	„Öffnet/erstellt eine Datei zum Schreiben mit „WRF“. Datum gibt das Erstellungs – oder Änderungsdatum an.	Gibt <promt><CR> zurück
„OPR“<SP><Name> <CR>	0Eh,20h,<Name>,0Dh	Öffnet eine Datei zum Lesen mit „RDF“	Gibt <promt><CR> zurück
„OPR“<SP><Name> <SP><Datum><CR>	0Eh,20h,<Name>,20h, <Datum>,0Dh	Öffnet eine Datei zum Lesen mit „RDF“. Datum gibt das Zugriffsdatum an.	Gibt <promt><CR> zurück
„CLF“<SP><Name> <CR>	0Ah,20h,<Name>,0Dh	Schließt die angegebene Datei	Gibt <promt><CR> zurück
„REN“<SP><orig Name> <SP><new Name><CR>	0Ch,20h,<orig Name>, 20h,<new Name>,0Dh	Umbenennen einer Datei oder eines Verzeichnisses	Gibt <promt><CR> zurück
„FS“<CR>	12h,0Dh	Gibt den freien Speicherplatz zurück. Sind mehr als 4GB frei, wird FFFFFFFFh zurückgegeben	<Freier Speicher (4Byte, LSB first)><CR>
„FSE“<CR>	93h,0Dh	Gibt den freien Speicherplatz zurück	<Freier Speicher (6Byte, LSB first)><CR>
„SEK“<SP><offset in hex (4 Bytes MSB first)><CR	28h,20h,<offset in hex (4Bytes,MSB first)>, 0Dh	Zu einem Offset innerhalb der geöffneten Datei springen. Beim Schließen der Datei werden alle Daten hinter der aktuellen Position abgeschnitten.	Gibt <promt><CR> zurück

Hinweis!
Im Befehl „WRF“ ist in der ELV-Veröffentlichung ein Fehler enthalten. Der Befehl darf nach den Daten kein <CR> mehr enthalten.

Einstellen der Baudrate (nur UART-Schnittstelle) - gelb markiert die BASIC-Tiger Baudraten SER1B			
„SBD“<SP><Divisor (3 Bytes, LSB first)><CR>	14h,20h,<Divisor (3 Bytes, LSB first)>,0Dh	Baudrate einstellen	Gibt <promt><CR> zurück
Baudrate	1. Byte	2.Byte	3. Byte
300	10h	27h	00h
600	88h	13h	00h
1.200	C4h	09h	00h
2.400	E2h	04h	00h
4.800	71h	02h	00h
9.600*	38h	41h	00h
19.200	9Ch	80h	00h
38.400	4Eh	C0h	00h
57.600	34h	C0h	00h
115.200	1Ah	00h	00h
230.400	0Dh	00h	00h
460.800	06h	40h	00h
921.600	03h	80h	00h
1.000.000	03h	00h	00h
1.500.000	02h	00h	00h
2.000.000	01h	00h	00h
3.000.000	00h	00h	00h

Power-Management-Befehle			
„SUD“<CR>	15h,0Dh	Aktiviert den Suspend-Modus. Das angeschlossene USB-Gerät wird deaktiviert, wenn kein Zugriff erfolgt (geringerer Verbrauch)	Gibt <promt><CR> zurück
„WKD“<CR>	16h,0Dh	Deaktiviert den Suspend-Modus. Das USB-Gerät bleibt immer aktiv, auch ohne Zugriff.	Gibt <promt><CR> zurück

Debug-Befehle			
„FWV“<CR>	13h,0Dh	Firmware-Version ausgeben	Gibt dieVersionsnummer der Firmware und des Bootloaders zurück <CR> „MAIN<SP>dd.dd.AAAA“ <CR> „RPRG<SP>d.ddR“<CR>< promt><CR>
„IDD“<CR>	0Fh,0Dh	Geräteinformationen ausgeben	<Daten wie unten dargestellt><CR>
	„USB VID = $“, 2 Bytes in ASCII, 0Dh		
	„USB PID = $“, 2 Bytes in ASCII, 0Dh		
	„Vendor Id = $“, 8 Bytes in ASCII, 0Dh		
	„Product Id = $“, 16 Bytes in ASCII, 0Dh		
	„Revision Level = $“, 4 Bytes in ASCII, 0Dh		
	„I/F“ = „SCSI“ or „ATAPI“ in ASCII, 0Dh		
	„FAT12“ or „FAT16“ or „FAT32“ in ASCII, 0Dh		
	„Bytes/Sector = $“, 2 Bytes in ASCII, 0Dh		
	„Bytes/Cluster = $“, 3 Bytes in ASCII, 0Dh		
	„Capacity = $“, 4 Bytes in ASCII, 0Dh		
	„Free Space = $“, 4 Bytes in ASCII, 0Dh		

Fehlermeldungen		
Unbekannter Befehl	Erweiterter ASCI-Befehlssatz	„Bad Command“<CR>
	Mikrocontroller-Befehlssatz	„BC“,0Dh
Ausführung des Befehls ist fehlgeschlagen	Erweiterter ASCI-Befehlssatz	„Command Failed“<CR>
	Mikrocontroller-Befehlssatz	„CF“,0Dh
Der zu löschende Ordner ist nicht leer	Erweiterter ASCI-Befehlssatz	„Dir Not Empty“<CR>
	Mikrocontroller-Befehlssatz	„NE“,0Dh
Datei ist schreibgeschützt	Erweiterter ASCI-Befehlssatz	„Read Only“<CR>
	Mikrocontroller-Befehlssatz	„RO“,oDh
Datei kann nicht geöffnet werden	Erweiterter ASCI-Befehlssatz	„Invalid“<CR>
	Mikrocontroller-Befehlssatz	„FI“.oDh
Kein freier Speicherplatz mehr	Erweiterter ASCI-Befehlssatz	„Disk Full“<CR>
	Mikrocontroller-Befehlssatz	„DF“,0Dh
Datei ist noch geöffnet	Erweiterter ASCI-Befehlssatz	„File Open“<CR>
	Mikrocontroller-Befehlssatz	„FO“,oDh