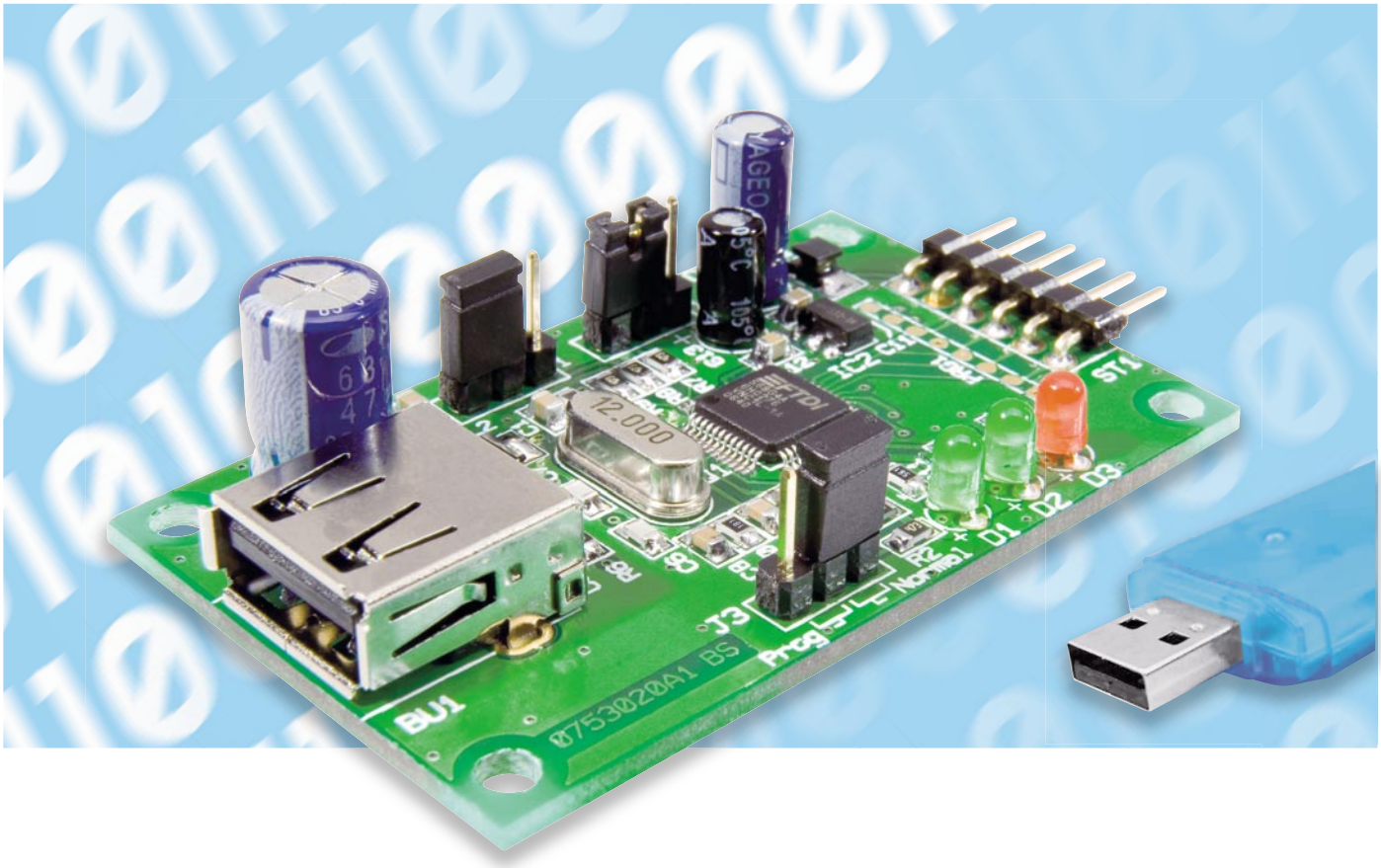


USB-Stick-Interface STI 100

So günstig und mit reichlich Speicherkapazität sie erhältlich sind, im Gegensatz zu anderen Speichermedien ist die Integration von USB-Sticks in eigene Schaltungsapplikationen ein kaum zu überwindendes Hindernis. Das Interface STI 100 bietet die Möglichkeit, einen USB-Stick einfach per UART- oder SPI-Schnittstelle anzusprechen. Der zugehörige Befehlssatz unterstützt vielfältige Dateioperationen, so dass die Verarbeitung von gesammelten Messwerten genauso möglich ist wie die Nutzung von auf einem PC erzeugten Daten (z. B. Konfigurationen) in der eigenen Schaltung.

USB-Stick für Eigenes nutzen

Speicherkarten für Digitalkameras, MP3-Player oder USB-Sticks für den PC sind mittlerweile Massenartikel und dementsprechend sehr günstig erhältlich. Auch nimmt die Speicherkapazität immer weiter zu, so dass riesige Datenmengen einfach und handlich transportiert werden können. Dem begeisterten Schaltungsentwickler bleiben diese Speichertechnologien jedoch meist verwehrt, da der Aufwand der Integration immens ist. Findet man für SD- oder MMC-Karten noch zahlreiche Hilfen im Internet, wird es bei der Verwendung von USB-Sticks schon schwierig. Denn hier muss nicht nur das Dateisystem (FAT, FAT32) verwaltet, sondern auch noch ein USB-Host samt „Mass Storage Device Class" zur Verfügung gestellt werden. Eine reine Software-Implementierung mit Hilfe von Standard-Mikrocontrollern ist daher fast unmöglich und für den normalen Anwender nicht realisierbar. Mit dem USB-Stick-Interface STI 100 jedoch wird die Nutzung eines USB-Sticks für den Schaltungsentwickler zum Kinderspiel. Zur Einbindung in eigene Hardwareprojekte muss nur eine serielle Schnittstelle wie UART (bevorzugt) oder SPI

Technische Daten: STI 100

Spannungsversorgung:	5 VDC, stabilisiert
Stromaufnahme:	Stand-by: 20 mA, Normalbetrieb: 150 mA
Eingangs-/Ausgangspegel:	TTL (3,3 V)
Anschlüsse:	USB-Buchse, Typ A, Stiftleiste
Abmessungen (B x H x T):	64 x 15 x 35 mm

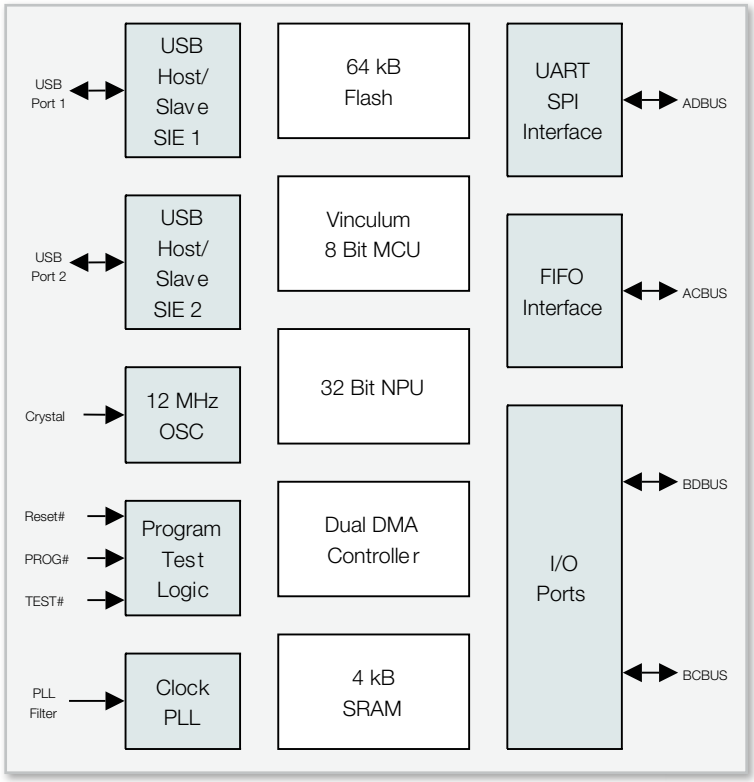


Bild 1: Interner Aufbau des USB-Controllers Vinculum VNC1L

vorhanden sein und eine 5-V-Versorgungsspannung zur Verfügung gestellt werden. Der Anschluss erfolgt über eine simple 6-polige Stiftleiste und ist damit schnell und einfach zu realisieren. Es sind keine Kenntnisse über USB-Spezifikationen und Protokolle oder über die Dateisysteme FAT, FAT32 notwendig. Dies wird vollständig von der intelligenten Hardware auf dem STI 100 übernommen. Ein umfangreicher, aber einfach beherrschbarer Befehlssatz genügt, um mit dem USB-Stick zu kommunizieren und Dateioperationen auszuführen. Damit steht dem ambitionierten Elektroniker ein universell einsetzbares Interface für eigene Applikationen wie etwa Datenlogger, Sound-Recorder und ähnliche Anwendungen zur Verfügung.
Betrachten wir kurz die Grundlagen des USB sowie des FAT-Dateisystems, bevor wir zur Beschreibung des Kernstücks der Schaltung, des USB-Controllers, kommen.

Universal Serial Bus

Der USB ist ein serielles Bussystem, dass allerdings physikalisch nur Punkt-zu-Punkt-Verbindungen realisiert. Startpunkt dieser Verbindungen ist der zentrale Host-Controller, an den bis zu 127 USB-Geräte angeschlossen werden können. Stellt der Host nicht genügend Anschlüsse zur Verfügung, können Hubs zwischengeschaltet werden, um weitere Geräte anzuschließen. Nur der Host-Controller kann Daten eines Gerätes lesen oder Daten schreiben, des Weiteren darf ein Gerät nur auf Anfrage des Hosts Daten senden. Die maximale Datenübertragungsrate beträgt 12 MBit/s (USB 1.1) bzw. 480 MBit/s (USB 2.0). Es sind verschiedene Geräteklassen spezifiziert, die es dem Host ermöglichen, Geräte ohne gerätespezifischen Treiber anzusprechen wie z. B. Tastatur, Maus, Kartenleser, USB-Stick.

FAT-Dateisystem

Das Dateisystem ist im Betriebssystem integriert, es ordnet die Daten, die auf einem Datenträger gespeichert sind, und ermöglicht den Zugriff auf diese Daten. Unter dem Microsoft-Betriebssystem „Windows" wird das FAT-Dateisystem verwendet. Der gesamte Speicherbereich des Datenträgers wird in eine feste Anzahl von Clustern aufgeteilt. Die Größe der Cluster hängt damit von der Gesamtkapazität ab. Da-

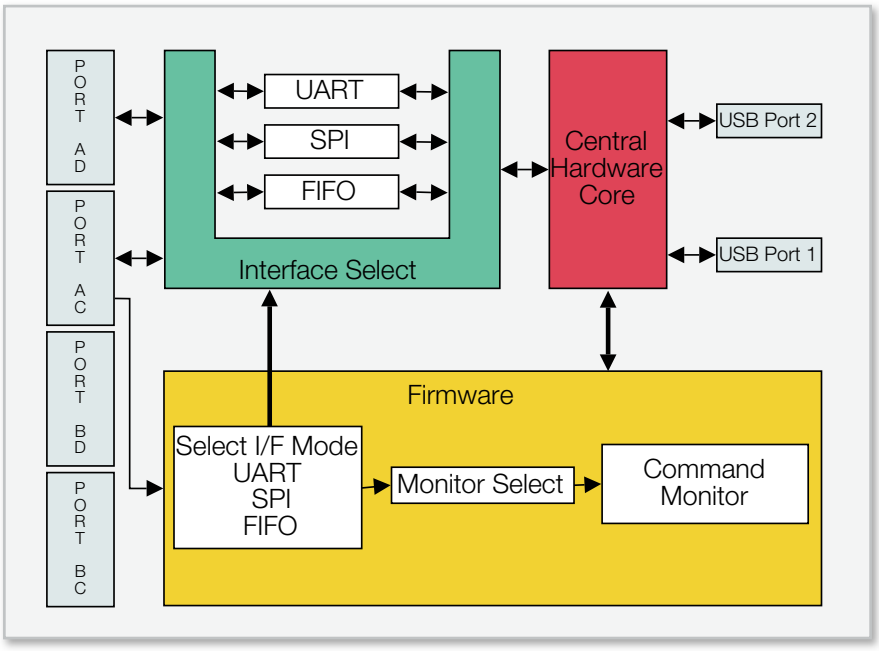


Bild 2: Schematische Darstellung der VNC1L-Firmware vom USB-Controller

Tabelle 1:

Erweiterter ASCII-Befehlssatz	Hexadezimaler Befehlssatz für Mikrocontroller	Beschreibung	Rückgabe
Konfigurationsbefehle			
„SCS"<cr>	$10, $0D	In den Mikrocontroller-Befehlssatz umschalten	Gibt „>"<cr> zurück, um anzuzeigen, dass der Baustein nun den Mikrocontroller-Befehlssatz akzeptiert
„ECS"<cr>	$11, $0D	In den erweiterten ASCII-Befehlssatz umschalten	Gibt „D:\>"<cr> zurück, um anzuzeigen, dass der Baustein nun den erweiterten ASCII-Befehlssatz akzeptiert
„IPA"<cr>	$90, $0D	Zahlendarstellung in ASCII-Zeichen	ECS-Mode und IPA-Mode aktiviert: „WRF"<sp>"10"<cr> 0102030405060708090a <prompt><cr>
„IPH"<cr>	$91, $0D	Zahlendarstellung als hexadezimaler Wert	SCS-Mode and IPH-Mode aktiviert: 08 20 00 00 00 0A 0d 01 02 03 04 05 06 07 08 09 0a <prompt><cr>
„E"<cr>	$45, $0D	Echo-Befehl	Gibt „E"<cr> zurück
„e"<cr>	$65, $0D	Echo-Befehl	Gibt „E"<cr> zurück
Prüfen, ob ein USB-Gerät angeschlossen ist			
<cr>	$0D	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>	$01, $0D	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>	$01, $20, <Name>,$0D	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>	$05, $20,<Name>,$0D	Verzeichnis löschen	Gibt <Prompt><cr> zurück
„MKD"<sp> <Name><cr>	$07, $20, <Name>,$0D	Verzeichnis erstellen	Gibt <Prompt><cr> zurück
„MKD"<sp> <Name><sp><Datum><cr>	$07, $20, <Name>,$20,<Datum>,$0D	Verzeichnis inkl. Erstellungsdatum erstellen	Gibt <Prompt><cr> zurück
„CD"<sp> <Name><cr>	$02, $20,<Name>,$0D	Verzeichniswechsel	Gibt <Prompt><cr> zurück
„CD"<sp>„.."<cr>	$02, $20,$2E,$2E,$0D	Eine Verzeichnisebene nach oben	Gibt <Prompt><cr> zurück.
Datei-Befehle			
„RD"<sp> <Name><cr>	$04, $20,<Name>,$0D	Datei lesen	Gibt die gesamte Datei byte-weise 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>	$0B, $20,<size in hex(4 Bytes)>, $0D	Liest angegebene Anzahl an Bytes der momentan geöffneten Datei	Gibt die angegebene Anzahl an Bytes zurück <Daten><prompt><cr>
„DLF"<sp> <Name><cr>	$07, $20,<Name>, $0D	Datei löschen	Gibt <Prompt><cr> zurück
„WRF"<sp> <size in hex(4 Bytes) ><cr> <data bytes of size><cr>	$08, $20,<size in hex(4 Bytes)>, $0D $data, $0D	Schreibt die angegebenen Daten ans Ende der geöffneten Datei	Gibt <Prompt><cr> zurück
„OPW"<sp> <Name><cr>	$09, $20, <Name>,$0D	Öffnet eine Datei zum Schreiben mit „WRF"	Gibt <Prompt><cr> zurück
„OPW"<sp> <Name><sp><Datum><cr>	$09, $20, <Name>,$20,<Datum>,$0D	Öffnet/erstellt eine Datei zum Schreiben mit „WRF". Datum gibt das Erstellungs- oder Änderungsdatum an.	Gibt <Prompt><cr> zurück

„OPR“<sp> <Name><cr>	$0E,$20, <Name>$0D	Öffnet eine Datei zum Lesen mit „RDF“	Gibt <Prompt><cr> zurück
„OPR“<sp> <Name><sp><Datum><cr>	$0E,$20, <Name>,$20,<Datum>,$0D	Öffnet eine Datei zum Lesen mit „RDF“. Datum gibt das Zugriffsdatum an.	Gibt <Prompt><cr> zurück
„CLF“<sp> <Name><cr>	$0A,$20, <Name,>$0D	Schließt die angegebene Datei	Gibt <Prompt><cr> zurück
„REN“<sp> <orig Name> <sp> <new Name><cr>	$0C,$20<orig Name,>$20, <new Name> $0D	Umbenennen einer Datei oder eines Verzeichnisses	Gibt <Prompt><cr> zurück
„FS“<cr>	$12, $0D	Gibt den freien Speicherplatz zurück. Sind mehr als 4 GB frei, wird $FFFFFFFF zurückgegeben.	<Freier Speicher (4 Byte, LSB first)> <cr>
„FSE“<cr>	$93,$0D	Gibt den freien Speicherplatz zurück	<Freier Speicher (6 Byte, LSB first)> <cr>
„SEK“<sp><offset in hex(4 Bytes MSB first) ><cr>	$28,$20,<offset in hex (4 Bytes, MSB first)>,$0D	Zu einem Offset innerhalb der geöffneten Datei springen. Beim Schließen der Datei werden alle Daten hinter der aktuellen Position abgeschnitten.	Gibt <Prompt><cr> zurück
Einstellen der Baudrate (nur UART-Schnittstelle)			
„SBD“<sp><Divisor (3 Bytes) LSB first ><cr>	$14, $20<Divisor (3 Bytes) >, $0D	Baudrate einstellen (siehe Tabelle 3)	Gibt <Prompt><cr> zurück
Power-Management-Befehle			
„SUD“<cr>	$15, $0D	Aktiviert den Suspend-Modus. Das angeschlossene USB-Gerät wird deaktiviert, wenn kein Zugriff erfolgt (geringerer Verbrauch).	Gibt <Prompt><cr> zurück
„WKD“<cr>	$16, $0D	Deaktiviert den Suspend-Modus. Das USB-Gerät bleibt immer aktiviert, auch ohne Zugriff.	Gibt <Prompt><cr> zurück.
Debug-Befehle			
„IDD“<cr>	$0F, $0D	Geräteinformationen ausgeben	<Daten wie in Tabelle 4 dargestellt><cr>
„FWV“<cr>	$13, $0D	Firmware-Version ausgeben	Gibt die Versionsnummer der Firmware und des Bootloaders zurück <cr> „MAIN <sp>dd.ddAAAA“<cr> „RPRG<sp>d.ddR“<cr> <prompt><cr>

Tabelle 1: Der Befehlssatz für die Kommunikation mit dem USB-Stick und Datei-Operationen

Tabelle 2: Fehlermeldungen

Fehler	Befehlssatz	Rückgabe
Unbekannter Befehl	Erweiterter ASCII-Befehlssatz	„Bad Command“<cr>
	Mikrocontroller-Befehlssatz	„BC“,$0D
Ausführung des Befehls ist fehlgeschlagen	Erweiterter ASCII-Befehlssatz	„Command Failed“<cr>
	Mikrocontroller-Befehlssatz	„CF“,$0D
Der zu löschende Ordner ist nicht leer	Erweiterter ASCII-Befehlssatz	„Dir Not Empty“<cr>
	Mikrocontroller-Befehlssatz	„NE“,$0D
Datei ist schreibgeschützt	Erweiterter ASCII-Befehlssatz	„Read Only“<cr>
	Mikrocontroller-Befehlssatz	„RO“,$0D
Datei kann nicht geöffnet werden	Erweiterter ASCII-Befehlssatz	„Invalid“<cr>
	Mikrocontroller-Befehlssatz	„FI“,$0D
Kein freier Speicherplatz mehr	Erweiterter ASCII-Befehlssatz	„Disk Full“<cr>
	Mikrocontroller-Befehlssatz	„DF“,$0D
Datei ist noch geöffnet	Erweiterter ASCII-Befehlssatz	„File open“<cr>
	Mikrocontroller-Befehlssatz	„FO“, $0D

Tabelle 2: Die Liste der möglichen Fehlermeldungen des STI 100

Tabelle 3: Baudraten-Tabelle für die UART-Schnittstelle			
Baudrate	**1. Byte**	**2. Byte**	**3. Byte**
300	$10	$27	$00
600	$88	$13	$00
1200	$C4	$09	$00
2400	$E2	$04	$00
4800	$71	$02	$00
9600*	$38	$41	$00
19.200	$9C	$80	$00
38.400	$4E	$C0	$00
57.600	$34	$C0	$00
115.200	$1A	$00	$00
230.400	$0D	$00	$00
460.800	$06	$40	$00
921.600	$03	$80	$00
1.000.000	$03	$00	$00
1.500.000	$02	$00	$00
2.000.000	$01	$00	$00
3.000.000	$00	$00	$00
* Nach einem Reset ist die Baudrate auf 9600 Baud eingestellt.			

Tabelle 3: Die Daten für die Einstellung der Übertragungsrate

teien belegen immer mindestens einen Cluster, ist die Datei größer, werden mehrere Cluster verwendet. In einer Tabelle wird festgehalten, welche Cluster von einer Datei belegt werden oder frei sind.

Der USB-Controller des STI 100

Der USB-Controller des Typs Vinculum VNC1L (Abbildung 1) ist eine Ein-Chip-Lösung, die alle oben genannten Anforderungen zur Nutzung eines USB-Sticks erfüllt. Der VNC1L integriert einen USB-Host-Controller sowie einen 8-Bit-Mikrocontroller mit 32-Bit-Coprozessor und 64-kB-Programmspeicher (Flash). Dadurch ist dieser Chip sehr universell und auf unterschiedliche Anforderungen anpassbar. Der Hersteller stellt unterschiedliche Firmwareversionen zur Verfügung, die prinzipiell wie in Abbildung 2 dargestellt aufgebaut sind. Die Daten werden vom USB-Port gelesen und über die gewählte Schnittstelle ausgegeben. Das Schreiben von Daten erfolgt entsprechend in umgekehrter Richtung. Die Firmware steuert dabei den Datenverkehr, indem es die Befehle über den Command-Monitor empfängt und die entsprechenden Lese- oder Schreibaktionen auslöst.

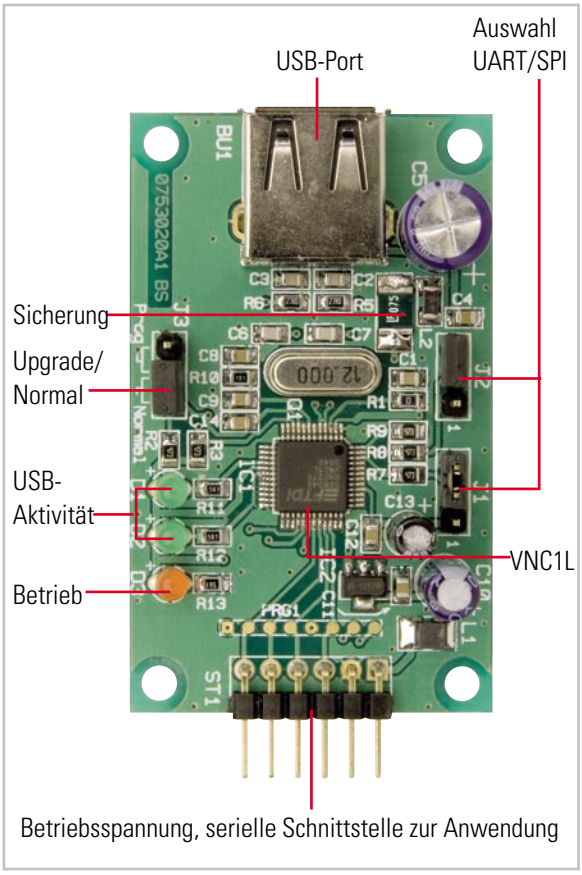
Der Befehlssatz

Um Zugriff auf Dateien und Ordner des USB-Sticks zu erhalten, ist ein umfangreicher Befehlssatz in den VNC1L integriert. Der Befehlssatz ist in zwei Versionen vorhanden, einmal als erweiterter ASCII-Befehlssatz, der andere ist als 8-Bit-Befehlssatz, zur vereinfachten Nutzung durch Mikrocontroller, ausgeführt. Mit den ASCII-Befehlen ist eine bequeme Nutzung auch per Terminalprogramm möglich.

Tabelle 4: Geräteeigenschaften
„USB VID = $", 2 Bytes in ASCII, $0D
„USB PID = $", 2 Bytes in ASCII, $0D
„Vendor Id = $", 8 bytes in ASCII, $0D
„Product Id = $", 16 Bytes in ASCII, $0D
„Revision Level = $", 4 Bytes in ASCII, $0D
„I/F = "SCSI" or „ATAPI" in ASCII, $0D
„FAT12" or „FAT16" or „FAT32" in ASCII, $0D
„Bytes/Sector = $", 2 Bytes in ASCII, $0D
„Bytes/Cluster = $", 3 Bytes in ASCII, $0D
„Capacity = $", 4 Bytes in ASCII, $0D
„Free Space = $", 4 Bytes in ASCII, $0D

Tabelle 4: Diese Geräteinformationen sind über den Debug-Befehl IDD verfügbar

Eine Übersicht der wichtigsten Befehle ist in Tabelle 1 dargestellt, welche die Syntax, Beschreibung und die entsprechenden Rückgaben enthält. Eine ausführliche Beschreibung der Firmware erfolgt im „Vinculum Firmware User Manual", das unter www.vinculum.com/documents.html#vfwspecs erhältlich ist. In Tabelle 2 sind mögliche Fehlermeldungen des VNC1L dargestellt.
Im Teil 2 beschäftigen wir uns mit der Schaltungstechnik, dem Nachbau sowie der Inbetriebnahme des STI 100. ELV



Anschluss-, Bedien- und Anzeigemöglichkeiten des STI 100