TECHNISCHE UNIVERSITÄT DRESDEN
FAKULTÄT ELEKTROTECHNIK UND INFORMATIONSTECHNIK
INSTITUT FÜR FESTKÖRPERELEKTRONIK
INSTITUTSDIREKTOR: PROF. DR.-ING. HABIL. G. GERLACH

PD Dr.-Ing. habil. Helmut Budzier
April 2017

Praktikum zur Lehrveranstaltung
Computertechnik II

Versuch
Sensoransteuerung
Aufgabe MEL 3

Inhaltsverzeichnis

1. Aufgabenstellung

Das Praktikum soll in die Funktionsweise serieller Schnittstellen von Rechnern und elektronischen Geräten sowie deren Programmierung unter Win32 einführen.
Zum Verständnis der Programmierung sind Kenntnisse des Aufbaus und der Funktionsweise der Schnittstelle erforderlich. Zur Demonstration ist der Computer mit einem Digitalmultimeter (DVM) verbunden.
Das DVM misst den Wert eines temperaturabhängigen Widerstandes. Nach Aufforderung sendet das Digitalvoltmeter diesen Wert an den Computer. Aus dem Widerstandswert wird dann die Temperatur des Widerstandes berechnet und auf dem Monitor angezeigt. Durch zyklisches Abfragen wird der angezeigte Wert ständig aktualisiert.
Im Einzelnen sind folgende Aufgaben zu lösen:

- Erstellen eines Programmablaufplans
- Initialisierung der COM-Schnittstelle
- Kommunikation mit dem Multimeter
- Umwandlung des Multimeterstrings in eine Zahl
- Berechnung der Temperatur
- Formatierung des Ausgabestrings
- Ausgabe des Strings
- Einfügen einer Zeitschleife zur Ausgabe eines neuen Messwerts ca. jede Sekunde
- Beenden des Programm mittels beliebigen Tastendrucks

Als Ergebnis ist ein lauffähiges Programm vorzuführen. Das Versuchsprotokoll soll den Programmablaufplan und den kommentierten Quelltext, sowie eine Diskussion der aufgetretenen Fehler und Ergebnisse enthalten.

2. Ablauf und Bewertung

Vorraussetzung für das Praktikum sind Kenntnisse aus den Vorlesungen „Informatik“ und „Computertechnik“, speziell zum Programmieren mit C und zum Erstellen von Programmablaufplänen.
In Vorbereitung auf das Praktikum müssen Sie sich mit der Funktionsweise der seriellen Schnittstelle vertraut machen. Es ist ein Programmablaufplan zu erarbeiten und zum Praktikumstermin mitzubringen.
Während des Praktikums findet ein Kolloquium statt, in dem der Betreuer die Kenntnisse der einzelnen Teilnehmer überprüft. Das erarbeitete Programm ist dem Betreuer vorzustellen, im Einzelschritt vorzuführen und zu erläutern. Nach dem Praktikum ist von der Gruppe ein Protokoll anzufertigen.
Die Leistung im Praktikum wird mit maximal 10 Punkten bewertet. Davon entfallen 6 Punkte auf das Kolloquium und die individuelle Leistung, 2 Punkte auf die Praktikumsdurchführung und 2 Punkte auf das Protokoll.

3. Funktionsweise serieller Schnittstellen

Serielle Schnittstellen sind sehr weit verbreitet. Viele Sensoren mit digitalem Ausgang und eine Vielzahl von Messgeräten verwenden serielle Schnittstellen. Die am weitesten verbreitete serielle Schnittstelle ist die sogenannte RS232 oder V.24. Jeder Personalcomputer hat in der Regel zwei RS232-Schnittstellen. Die Bezeichnung RS steht für Receive/Send.
Serielle Schnittstellen sind Punkt-zu-Punkt-Verbindungen. Die Informationen werden bitseriell übertragen. Dadurch kann man mit einem sehr geringen Kostenaufwand zwei Geräte miteinander verbinden. Meistens reicht schon eine 3-Drahtleitung (2 Signale + Masse) aus.
Die Geschichte der RS232-Schnittstelle beginnt 1969. Seitdem wurde der technische Standard ständig weiter entwickelt. Die aktuellen Spezifikationen stammen aus den 90-er Jahren (RS232-C). International gilt der Standard der EIA (Electrical Industries Association). Die aktuelle Norm ist die RS232-C. In Europa gelten die Standards der CCITT (Commitee Consultatif International de Telegraph et Telephone) bzw. in Deutschland die DIN 66020. Der Schnittstellen-Standard besteht aus drei Teilen:
- elektrischer Teil: legt Spannungen und Signalpegel fest (CCITT V.28)
- funktioneller Teil: legt Signale und ihre Funktionen fest (CCITT V.24)
- physikalischer Teil: legt die Steckverbinder fest (ISO 2110)

Bei der Übertragung unterscheidet man zwei Geräte:
- Daten-End-Einrichtung (DDE bzw. Data Terminal Equipment DTE) und
- Daten-Übertragungs-Einrichtung (DÜE bzw. Data Communication Equipment DCE)

Zu den DDEs gehören Terminals, Drucker und PCs. Eine DÜE ist z.B. ein Modem.

Die Leitungen der Schnittstelle werden in vier Klassen unterteilt:
- Signalleitungen: TxD, RxD
- Steuerleitungen: RTS, DTR, CTS, DSR, DCD, RI
- Masse- und Erdleitungen bzw. Schirm: PG, SG
- Taktleitungen (nur bei synchroner Übertragung): TxC

Die Kabellänge darf 15 m nicht übersteigen.

Folgende Bezeichnungen und Steckerbelegungen gelten:

Signal	Abkürzung			Steckerbelegung	
	USA	DIN	CCITT	25-polig	9-polig
Schutzerde	PG	E1	101	1	-
Betriebserde	SG	E2	102	7	5
Sendedaten	TxD	D1	103	2	3
Sender anfordern	RTS	S2	105	4	7
Empfänger bereitschaft	DTR	S1	108	20	4
Sendetakt	TxC	T2	114	15	-
Empfangs-daten	RxD	D2	104	3	2
Sende-bereitschaft	CTS	M2	106	5	8
Betriebs-bereitschaft	DSR	M1	107	6	6
Signalträger erkannt	DCD	M5	102	8	1
Ring-indikator	RI			22	9

Die Sendedatenleitung TxD (Transmit Data Line) überträgt die seriellen Informationen vom DTE zum DCE. Wenn keine Daten gesendet werden, muss „logisch 1“ (MARK) gesendet werden. Daten werden nur übertragen, wenn DSR, DTR, RTS und CTS eingeschaltet sind.
Die Empfangsleitung RxD (Receive Data Line) dient zum Empfang der Daten vom DCE. Wenn keine Daten übertragen werden, muss die RxD-Leitung im „logisch 1“ Zustand gehalten werden.
Wenn das DCD-Signal (Data Carrier Detect) eingeschaltet ist, empfängt der DTE vom DCE ein auswertbares Signal.
Das DTR-Signal (Data Terminal Ready) zeigt an, dass die Geräte betriebsbereit sind. Erst nach Einschalten der DTR-Signale beider Geräte kann überhaupt eine Übertragung bzw. eine Auswertung der Steuersignale beginnen.
Das DSR-Signal (Data Set Ready) wird vom DCE eingeschaltet, wenn es mit dem DTE verbunden ist.

Das DTE schaltet das RTS-Signal (Request To Send) ein, wenn es Daten senden will. Nach Empfang dieses Signals schaltet das DCE-Gerät das Signal CTS (Clear To Send) ein, wenn es empfangsbereit ist. Nach Beendigung der Datenübertragung schaltet das DTE das RTS-Signal aus. Daraufhin schaltet auch das DCE CTS aus.
Mit den beiden Signalen RTS und CTS kann ein Hardware-Handshake durchgeführt werden. Dazu muss der Datenempfänger das CTS-Signal ausschalten, wenn es keine Daten empfangen kann, bzw. wieder einschalten, wenn es wieder empfangsbereit ist. Der Sender quittiert jeweils mit RTS.
Der Ringindikator wird in Standardtelefonmodems verwendet. Es wird vom DCE eingeschaltet, wenn ein Ruf ankommt.
Alle Signale sind auf die Gerätemasse (Betriebserde) bezogen. Eine Masseleitung muss immer mitgeführt werden. Als Abschirmung der Leitungen muss die Betriebserde verwendet werden.

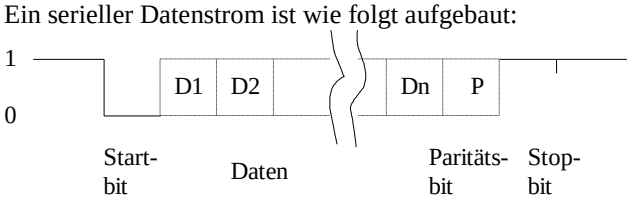
Es sind folgende elektrische Eigenschaften definiert:
- Schwellspannungen: +/- 3V
- maximale Pegelspannung +/- 12V

Daten werden negiert übertragen, d.h.:
- logisch 1: Pegel < -3 V (auch als MARK bezeichnet)
- logisch 0: Pegel > + 3 V (auch als SPACE bezeichnet)

Steuersignale werden positiv übertragen:
- Signal eingeschaltet: Pegel > +3 V
- Signal ausgeschaltet: Pegel < - 3 V

Es werden nur Daten übertragen, wenn alle Steuerleitungen eingeschaltet sind.

Ein serieller Datenstrom ist wie folgt aufgebaut:

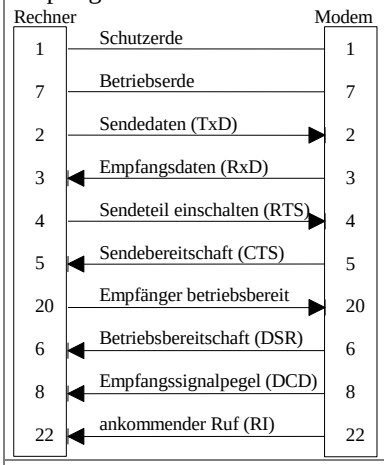


Jede Übertragung beginnt mit einem Startbit. Dann folgen 5, 6, 7 oder 8 Datenbits, ein Paritätsbit und 1, 1.5 oder 2 Stopbits. Das Paritätsbit kann entfallen. Es ist zu beachten, dass nicht alle Kombinationen zulässig sind. So sind z.B. bei 5 Datenbits zwei Stopbits nicht erlaubt. Für 6, 7 oder 8 Datenbits sind 1.5 Stopbits unzulässig.
Die Übertragung erfolgt asynchron mit 75, 110, 135, 150, 300, 600, 1200, 2400, 4800, 9600 oder 19200 Bits/s (= Baud = bps). Weitere Übertragungsgeschwindigkeiten sind nicht standardisiert. Es werden aber häufig weitere verwendet: 38400, 56000, 57600, 115200, 128000, 256000 usw.
Die Baudrate bezieht sich dabei immer auf alle übertragenen Bits, einschließlich Start-, Stop- und Paritätsbit. So können mit einer Baudrate von 19200 Baud, bei 8 Datenbits, keinem Paritätsbit und einem Stopbit maximal 15360 Datenbit/s (= 1920 Byte/s) übertragen werden.

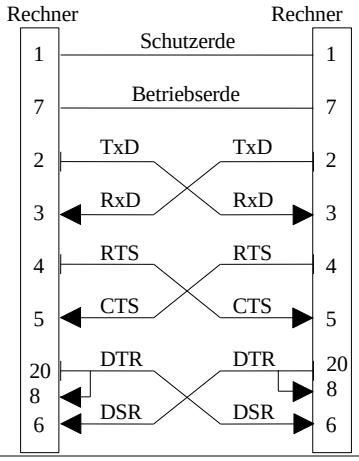
Die Festlegung des Übertragungsprotokolls erfolgt mit der Software. Die Umsetzung des Protokolls geschieht in einer UART (Universal Asynchronous Receiver/Transmitter). Diese UART ist Bestandteil eines jeden PC und wird vom Mikroprozessor direkt angesprochen. Als Standard-IC wird dabei meistens ein 16C450 oder dazu kompatibler IC (16C550, 16C750) verwendet. Die seriellen Ein- bzw. Ausgänge der UART werden dann auf einen Pegelwandler gegeben, der direkt die RS232-Signale erzeugt.

Es werden zwei Arten von Kabeln eingesetzt. Das sogenannte Modemkabel verbindet einen DCE mit einem DTE. Es werden alle Pins Eins-zu-eins verbunden. Beim Null-Modemkabel hingegen, werden die Leitungen gekreuzt. Es dient zur Verbindung zweier DTE. Auf der nächsten Seite sind die Kabelverbindungen schematisch dargestellt.
Häufig werden nicht alle Steuerleitungen ausgewertet. Nichtverwendete Leitungen werden dann weggelassen. Die einfachste Verbindung wäre eine Dreidrahtleitung mit TxD, RxD und Masse. Die offenen Eingänge müssen auf definierte Potentiale gelegt werden.
Nicht alle Anwender von RS232-Schnitstellen halten sich an die Standards. Oft werden die Steuerleitungen zur Spannungsversorgung eingesetzt. Das bekannteste Beispiel hierfür ist die serielle Computermaus.

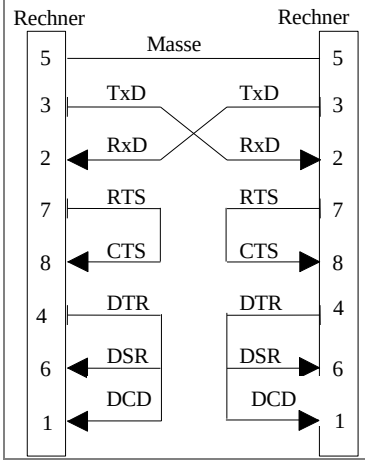
Modem-Kabel
25 poliger Steckverbinder



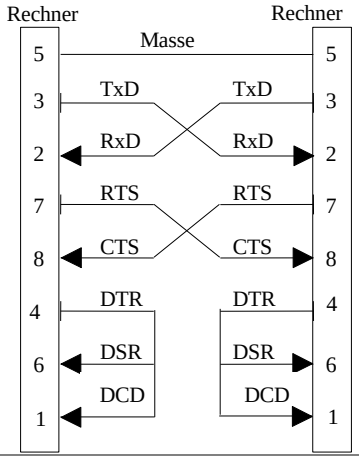
Null-Modemkabel
25 poliger Steckverbinder



minimales Null-Modemkabel
9-poliger Steckverbinder

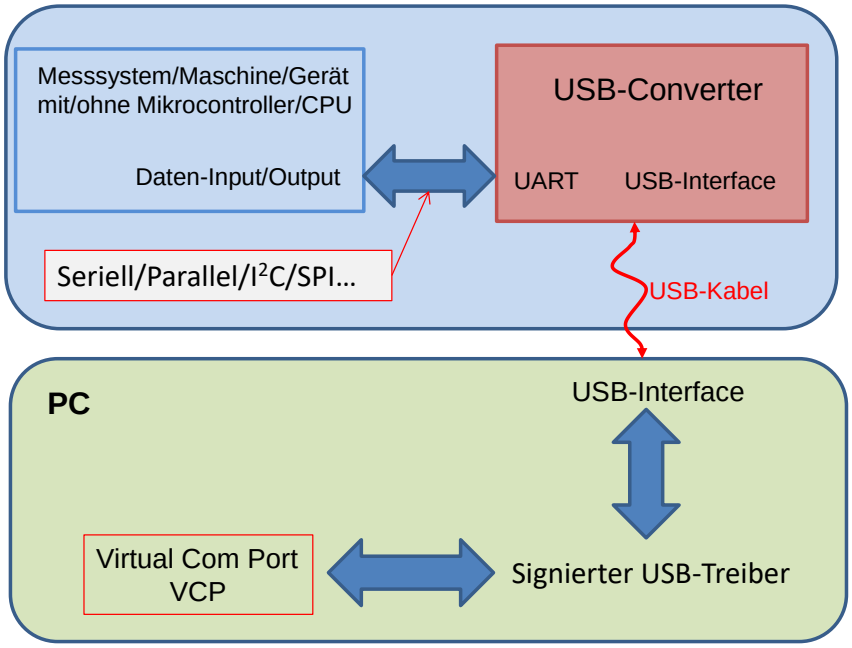


Null-Modemkabel mit Handshake-Signalen
9-poliger Steckverbinder



4. Serielle Schnittstellen im modernen PC

Obwohl die heutigen Home-PCs keine externen RS232-Schnittstellen mehr aufweisen, hat das serielle Interface auch weiterhin eine große Bedeutung. Vor allem in der Industrie sind serielle Schnittstellen weit verbreitet. Dort allerdings oftmals mit anderen elektrischen Parametern (RS422, RS485), aber mit der oben beschriebenen Softwareprogrammierung.
Fast alle Messgeräte, die über USB an einen PC angeschlossen werden, erstellen im PC eine virtuelle serielle Schnittstelle (Virtual COM Port, VCP) (teilweise auch mit Steuerleitungen). Folgendes Bild zeigt das Prinzip:



Das hat verschiedene Vorteile:

- Ein einfacher Messfühler mit z.B. einer I^2C-Schnittstelle kann mit einem USB-Converter-IC direkt an einen PC angeschlossen werden.
- Jedes USB-Gerät muss einen signierten Treiber für Windows (möglichst für alle Windowsversionen!) mitbringen. Hersteller von USB-Converter-ICs, wie z.B. FTDI, stellen diesen kostenfrei zur Verfügung. Er installiert eine oder mehrere virtuelle serielle Schnittstelle (VCP).
- Da VCPs sich wie „normale“ serielle Schnittstellen verhalten und programmiert werden, bedarf es keine Kenntnisse über die USB-Treiber-Programmierung. Dies erlaubt eine einfache Einbindung der Geräte in eigene aber auch in kommerzielle Programme zur Laborautomatisierung, wie z.B. LabView.

Alle „besseren“ Mainboards, z.B. für Server, haben ein serielles Interface integriert (Pfostensteckverbinder auf dem Board, ohne genormte Anschlussbelegung). Es dient zum Debugging der Hardware und des BIOS über ein Terminal. Deshalb findet sich im Gerätemanager fast immer ein Kommunikationsanschluss COM1.
Soll ein Gerät mit einem COM-Port an einen PC ohne COM-Port angeschlossen werden, können Adapterkabel mit im Steckverbinder integrierten USB-Converter eingesetzte werden.

5. Programmierung serieller Schnittstellen unter Win32

Die seriellen Schnittstellen sind fest in das File-I/O-System von Windows eingebunden. Die seriellen Schnittstellen und auch der parallele Port gehören zu den allgemeinen Kommunikationsressourcen des Computers. Alle File-Ein/Ausgabefunktionen können auf die seriellen Schnittstellen angewandt werden. Man kann also auf die Schnittstellen zugreifen, als seien sie Dateien. Zusätzlich existiert eine Reihe von Funktionen, die dem Lesen und dem Setzen der Schnittstellenparameter dienen.
Die Arbeit mit einer Kommunikationsressource läuft wie folgt ab:

1. Öffnen eines Handles, der mit der Ressource verbunden ist,
2. Abfrage und Setzen der Parameter der Ressource,
3. Lesen oder Schreiben,
4. Schließen des Handles und damit der Ressource.

Ein Handle ist eine eindeutige Nummer, die das Betriebssystem vergibt und die eine Ressource identifiziert.

Die direkte Programmierung der UART übernimmt das Betriebssystem. Dadurch können unabhängig von der konkreten Hardware Anwendungsprogramme geschrieben werden. Das Betriebssystem richtet zum Lesen und zum Schreiben zwei Zwischenspeicher ein, den sogenannten Ein- und -ausgabepuffer. Die Filefunktionen lesen und schreiben in diese Puffer. Die Datenein- und -ausgabe erfolgt wiederum durch das Betriebssystem. Das Anwenderprogramm kann sich durch das Betriebssystem informieren lassen, wenn bestimmte Ereignisse eintreten. Solche Ereignisse sind z.B. alle Daten gesendet, ein Zeichen empfangen oder auch ein bestimmtes Zeichen empfangen.

Es werden im Folgenden die Funktionen nur soweit beschrieben, wie sie im Praktikum benötigt werden. Vollständige Informationen enthält die Online-Hilfe des Compilers bzw. ist in der Fachliteratur zu finden.

5.1. Allgemeine Filefunktionen

Die Filefunktionen lassen sich auf alle Quellen bzw. Ziele von Daten anwenden. Dazu gehören auch die seriellen und parallelen Schnittstellen. Alle Systemressourcen werden mit ihrem Namen angesprochen: Files mit dem Dateinamen und die Schnittstellen mit COM1, COM2, LPT1 usw.
Zuerst muss eine Systemressource mit CreateFile() geöffnet werden:

```
HANDLE CreateFile( LPCTSTR lpFileName,  DWORD dwDesiredAccess,
  DWORD dwShareMode, LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition, DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile );
```

Die Funktion benötigt verschiedene Parameter:

- Filenamen, z.B. “COM1“
- Zugriffsmode: Wir wollen Lesen und Schreiben:
  dwDesiredAccess = GENERIC_READ | GENERIC_WRITE.
- Die Schnittstelle muss exklusiv geöffnet werden:
  dwShareMode = 0.
- Selbstverständlich muss die Schnittstelle bereits existieren:
  dwCreationDisposition = OPEN_EXISTING.
- Alle anderen Parameter werden NULL gesetzt.

Der Rückgabewert ist bei Erfolg der Handle der Schnittstelle. Schlägt das Öffnen der Schnittstelle fehl, ist der Rückgabewert gleich INVALID_HANDLE.
Beispiel:

```
HANDLE hCOM;
hCOM = CreateFile("COM1", GENERIC_READ | GENERIC_WRITE, 0, NULL,
                  OPEN_EXISTING, NULL, NULL);
if (hCOM == INVALID_HANDLE_VALUE) {
        //Hier Fehlerbehandlung
}
```

Das Gegenstück zum Öffnen der Schnittstelle ist das Schließen:

```
BOOL CloseHandle( HANDLE hObject   );
```

Diese Funktion benötigt nur den gültigen Handle. Bei erfolgreichen Schließen der Schnittstelle gibt die Funktion *TRUE* zurück, ansonsten *FALSE*. Der Handle ist anschließend ungültig und darf nicht mehr verwendet werden.

Das Lesen und Schreiben erfolgt via Read/WriteFile:

```
BOOL ReadFile( HANDLE hFile,  LPVOID lpBuffer,
  DWORD nNumberOfBytesToRead,  LPDWORD lpNumberOfBytesRead,
  LPOVERLAPPED lpOverlapped  );
```

```
BOOL WriteFile( HANDLE hFile,  LPCVOID lpBuffer,
  DWORD nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten,
  LPOVERLAPPED lpOverlapped );
```

hFile ist der bereits bekannte Handle der Ressource. Die Daten werden aus lpBuffer gelesen bzw. in lpBuffer geschrieben. Die Anzahl der Bytes, die gelesen bzw. geschrieben werden sollen, werden in nNumberOfBytes übergeben. Die Anzahl der Bytes die wirklich gelesen/geschrieben worden sind, werden in lpNumberOfBytesRead/Written zurückgegeben. Der letzte Parameter wird NULL gesetzt.
Der Rückgabeparameter gibt an, ob die Funktionen ausgeführt werden konnten. ReadFile kehrt erst zurück, wenn alle Daten gelesen wurden. Die Funktion versucht solange zu lesen, bis Time out kommt. Standardmäßig steht Time out auf Unendlich! Deshalb sollte man immer überprüfen, wieviel Bytes gelesen werden können. WriteFile schreibt in einen Zwischenspeicher und kehrt sofort zurück. Normalerweise sollte die Anzahl der zu schreibenden/lesenden Bytes gleich der wirklich geschriebenen/gelesenen Bytes sein.
Das folgende Beispiel zeigt Lesen und Schreiben in eine Datei:

```
HANDLE handle;
char  Puffer[100];
DWORD dw;
//Öffnen der existierenden Datei Test.text:
handle = CreateFile("Test.text", GENERIC_READ | GENERIC_WRITE, 0, NULL,
                        OPEN_EXISTING, NULL, NULL);
if (handle == INVALID_HANDLE_VALUE) {
        //Hier Fehlerbehandlung
}
//Lesen aus der Datei:
if (!ReadFile(handle, Puffer, 100, &dw, NULL)) {
        //Hier Fehlerbehandlung
}
if (dw != 100) {
        //Hier Fehlerbehandlung
}
//An dieser Stelle können die Daten verändert werden.
//Anschließend werden sie zurückgeschrieben:
if (!WriteFile(handle, Puffer, 100, &dw, NULL)) {
        //Hier Fehlerbehandlung
}
if (dw != 100) {
        //Hier Fehlerbehandlung
}
//Abschließend wird die Datei geschlossen:
if (!CloseHandle(handle)) {
        //Hier Fehlerbehandlung
}
```

5.2. Schnittstellenspezifische Funktionen

Nach dem Öffnen der Schnittstelle müssen die Parameter der Schnittstelle, wie z.B. die Baudrate, eingestellt werden. Dazu gibt es einige spezielle Funktionen, die im Folgenden vorgestellt werden.
Zum Lesen und zum Setzen der Einstellungen dienen die Funktionen:

```
BOOL GetCommState( HANDLE hFile,  LPDCB lpDCB );

BOOL SetCommState( HANDLE hFile,    LPDCB lpDCB );
```

hFile ist der bereits bekannte Handle der Schnittstelle. Vom besonderen Interesse ist die Struktur *DCB*. In dieser Struktur sind alle Schnittstellenparameter enthalten. Der Rückgabewert der Funktion ist *TRUE*, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten *FALSE*.
Nach dem Öffnen der Schnittstelle werden die Schnittstellenparameter auf die Defaultwerte des Betriebssystems gesetzt. Diese liest man mit *GetCommState()* und setzt dann die eigenen Parameter. Anschließend schreibt man die DCB-Struktur mit *SetCommState()* zurück.
Eingestellt werden müssen:

Bezeichnung	Parameter	Erläuterung
Baudrate	*BaudRate*	*CBR_110,CBR_19200,CBR_300, CBR_38400,CBR_600,CBR_56000, CBR_1200,CBR_57600,CBR_2400, CBR_115200,CBR_4800, CBR_128000,CBR_9600, CBR_256000,CBR_14400*
Parität	*Parity*	*EVENPARITY*: gerade Parität *ODDPARITY*: ungerade Parität *NOPARITY*: keine Parität *SPACEPARITY*: Space Parität *MARKPARITY*: Mark Parität
Stopbits	*StopBits*	*ONESTOPBIT*: ein Stopbit *ONE5STOPBITS*: 1.5 Stopbits *TWOSTOPBITS*: zwei Stopbits
Datenbits	*ByteSize*	5, 6, 7 oder 8
RTS-Steuerung	*fRtsControl*	*RTS_CONTROL_DISABLE*: RTS-Signal aus *RTS_CONTROL_ENABLE*: RTS-Signal ein *RTS_CONTROL_HANDSHAKE*: Handshake RTS-CTS *RTS_CONTROL_TOGGLE*: RTS ein, wenn Daten gesendet werden

Folgendes Beispiel zeigt das Setzen der Baudrate:

```
DCB dcb;
if (!GetCommState(hCOM, &dcb)) {      // DCB holen
        // Hier Fehlerbehandlung
}
dcb.BaudRate = CBR_1200;              // 1200 Bd einstellen

if (!SetCommState(hCOM, &dcb)) {      //Parameter setzen
        // Hier Fehlerbehandlung
}
```

Neben dem Setzen der Parameter ist auch der aktuelle Zustand der Schnittstelle von Interesse. Dazu gehört u. a., wie viele Daten gesendet wurden, wie viele Daten empfangen wurden und wie die Pegel der Steuerleitungen sind. Diese Informationen erhält man mit der Funktion:

```
BOOL ClearCommError(  HANDLE hFile,  LPDWORD lpErrors,
            LPCOMSTAT lpStat  );
```

hFile ist wieder der bekannte Schnittstellenhandle. Der Parameter *lpErrors* liefert einen Fehlercode, wie z.B. *CE_RXPARITY* (Paritätsfehler). Der gesuchte Zustand ist in *lpStat* codiert. Der Rückgabewert der Funktion ist *TRUE*, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten *FALSE*.

Die Struktur *COMSTAT* enthält den Zustand der Steuerleitungen und weitere Informationen. U.a. sind folgende Werte implementiert:

Signal/Wert	Name
CTS	*fCtsHold*
DSR	*fDsrHold*
Anzahl empfangener Zeichen	*cbInQue*
Anzahl gesendeter Zeichen	*cbOutQue*

Die Anzahl der empfangenen Zeichen bezieht sich dabei auf die Zeichen, die zwar empfangen wurden, aber vom Programm noch nicht gelesen wurden, und damit noch im Eingabepuffer stehen.
Folgendes Beispiel fragt die Anzahl der Zeichen ab, die noch gelesen werden müssen und liest diese dann:

```
COMSTAT status;
DWORD  dw;
DWORD anzahl;
char puffer[];
...
ClearCommError(hCOM, &dw, &status);
anzahl = status.cbInQue;
if (anzahl)
        ReadFile(hCOM, puffer, anzahl , &dw, NULL);
```

Die Größe der Ein- und Ausgabepuffer läßt sich mit

```
BOOL SetupComm( HANDLE hFile,  DWORD dwInQueue,  DWORD dwOutQueue  );
```

festlegen. Programme die mit sehr großen Datenpaketen arbeiten, sollten die Puffergröße einstellen. Ein Programm, das mit 1024-Byte-Paketen arbeitet, wird als Puffergröße ein Vielfaches von 1024 einstellen. Die Eigenschaften der seriellen Schnittstellen fragt folgende Funktion ab:

```
BOOL GetCommProperties(HANDLE hFile, LPCOMMPROP lpCommProp ) ;
```

Die Struktur COMMPROP enthält u.a. die Größe der Puffer und deren Maximum, die maximal einstellbare Baudrate, die Art der Schnittstelle (RS232, RS 485 usw.) und weitere Eigenschaften.

6. Versuchsaufbau und Durchführung

Die Temperatur soll mit einem Widerstandsthermometer bestimmt werden. Dazu wird ein temperaturabhängiger Widerstand aus Platin verwendet. Dessen Temperaturkoeffizient ist ca. $4*10^{-3}\ K^{-1}$.
Zum Einsatz kommt ein Platinwiderstand vom Typ PT100. Er hat bei 0 °C einen Widerstand von 100,0 Ohm. Sein temperaturabhängiger Widerstand errechnet sich aus:

$$R(\vartheta) = R_0\,(1+\alpha(\vartheta-\vartheta_0)+\beta(\vartheta-\vartheta_0)^2)$$

$$R_0 = 100{,}0\,\Omega$$

$$\alpha = 3{,}90802*10^{-3}K^{-1}$$

$$\beta = 0{,}580195*10^{-6}K^{-2}$$

$$\vartheta_0 = 273{,}1K$$

$$\vartheta : Temperatur in K$$

Der Widerstand wird mit einem Handmultimeter (DVM) gemessen. Am IFE stehen zwei Typen von Multimetern zur Verfügung:

- Protek 506
- PeakTech 4390

Beide Multimeter besitzen eine RS232-Schnittstelle und sind über ein Modemkabel mit dem PC verbunden. Welches DVM verwendet werden soll, wird beim Versuchsbeginn vorgegeben.

Das DVM ist auf Widerstandsmessung einzustellen. Beim Protek 506 ist zusätzlich die RS232 einzuschalten. Nach dem Initialisieren der Schnittstelle des PC muss dem DVM zuerst ein Zeichen gesendet werden. Daraufhin sendet das DVM einen Antwortstring, der den Widerstandswert enthält. Aus diesem String wird dann der Widerstand gelesen und mittels obiger Formel (nach ϑ umstellen!) die Temperatur berechnet. Die Anzeige erfolgt in der Form:

Temperatur = 23.4 Grad

Die Temperaturanzeige soll ca. aller Sekunde aktualisiert werden. Mit einem beliebigen Tastendruck wird das Programm beendet.

Der PC setzt als Betriebssystem Windows (Version 7 oder 10) ein. Das Programm wird mit der integrierten Entwicklungsumgebung Microsoft Visual Studio C++ erstellt. Vorzugsweise wird eine Console-Applikation mit ausschließlicher Textausgabe verwendet. Besonderer Wert ist auf eine Fehlerbehandlung zulegen, so dass z.B. nach Ausschalten des DVM das Programm ordentlich beendet werden kann.

5.1. Beschreibung der Multimeter

a) DVM **Protek 506**

Zur Kommunikation des PC mit dem DVM muss das DVM umgeschaltet werden. Dies geschieht folgendermaßen:

1. Solange die Menutaste drücken bis die Anzeige „RS232“ blinkt,
2. Return-Taste drücken.

Die Schnittstelle arbeitet mit den Parametern:

Datenrate: 1200 Baud
Datenbits: 7
Parität: keine
Stopbits: 2

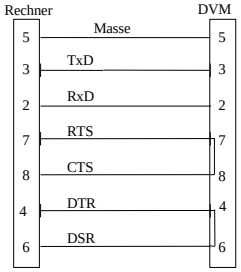
Sendedaten:
Es ist das Zeichen D zu senden. Daraufhin sendet das DVM sofort den Antwortstring.

Datenformat des DVM-Strings:
Es wird eine Bytefolge mit maximal 15 Byte gesendet. Folgende Tabelle zeigt drei Beispiele:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e
1.	R	E	S		3	.	9	9	9		M	o	h	m	CR
2.	R	E	S		3	9	9	.	9		o	h	m	CR	
3.	R	E	S		O	L		CR							

wobei CR das ASCII-Zeichen 13 ist.

Kabel:



b) DVM **PeakTech 4390**

Schnittstellenparameter:
Datenrate: 9600 Baud
Datenbits: 7
Parität: keine
Stopbits: 2
Zusätzlich ist RTS auszuschalten.

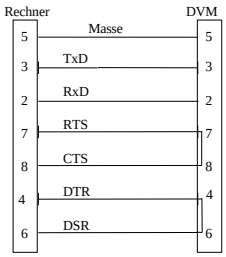
Sendedaten:
Es ist das Zeichen D zu senden. Daraufhin sendet das DVM sofort den Antwortstring.

Datenformat des DVM-Strings:
Es werden 4 Bytefolgen mit jeweils 14 Byte gesendet. Folgende Tabelle zeigt ein Beispiel:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d
1.	O	H			0	2	9	.	8	k	O	h	m	CR
2.														CR
3.					0	2	9	.	8					CR
4.					0	2	9	.	8					CR

wobei CR wiederum das ASCII-Zeichen 13 ist.

Kabel:



7. Kolloquiumsschwerpunkte

Im Kolloquium stehen folgende Sachverhalte im Mittelpunkt:

- Aufbau und Funktion eines Computers
- Funktionsweise der seriellen Schnittstelle
- Programmierung in C:
  Stringverarbeitung
  Formatierung von Ausgaben

8. Literaturhinweise

Hardware:

- Wittgruber, Friedrich
  Digitale Schnittstellen und Bussysteme
  Friedr. Vieweg & Sohn Verlagsgesellschaft, Braunschweig/Wiesbaden, 1999
- Blank, Hans-Joachim
  Sensoren am PC
  Markt&Technik Buch- und Softwareverlag, München, 1996
- Tränkler, Hans-Rolf
  Taschenbuch der Meßtechnik
  R.Oldenbourg Verlag, München/Wien, 1996

Software:

- Kruglinsky, David J.; Wingo, Scot; Shepherd, George
  Inside Visual C++
  Microsoft Press, 1999
- http://msdn.microsoft.com/default.asp

9. Arbeits- und Brandschutzhinweise

8.1. Vorbeugende Maßnahmen

- Die Praktikumsteilnehmer haben sich so zu verhalten, dass Gefahrensituationen und Unfälle vermieden werden.

- Die Befugnis zum Bedienen und Nutzen von Geräten ist auf die zugewiesenen Praktikumplatz beschränkt.

- Veränderungen und Eingriffe in die zum Praktikumsaufbau gehörenden Geräte sind nicht erlaubt.

- Der Anschluss und der Betrieb privater Geräte in den Praktikumsräumen sind verboten.

- Defekte an Geräten oder Gebäudeeinrichtungen sind unverzüglich dem Betreuer mitzuteilen. Betroffene Geräte sind außer Betrieb zu nehmen. Andere Personen sind vor Gefahren zu warnen.

- Den Anweisungen der Praktikumsbetreuer bzw. einer anderen aufsichtsführenden Person ist unbedingt Folge zu leisten.

- Betriebsfremde dürfen sich nur mit Erlaubnis des Praktikumsbetreuers in den Praktikumsräumen aufhalten.

- Rauchen und Umgang mit offenem Licht ist nicht gestattet.

- Essen und Trinken in den Praktikumsräumen ist nicht gestattet.

- Nach Ende des Praktikums ist der Arbeitsplatz im sauberen und aufgeräumten Zustand zu hinterlassen.

- Außergewöhnliche Ereignisse bzw. besondere Vorkommnisse sind umgehend dem Betreuer zu melden.

8.2. Verhalten im Brandfall

Beachten der richtigen Reihenfolge:

MELDEN -> RETTEN -> LÖSCHEN

1. Feuer melden:
   - *Telefonische Brandmeldung:
     - Notruf 112 der Feuerwehr
     - Notruf HA 4515 der Technischen Leitzentrale der TUD
   - * Deutliche, genaue und vollständige Angaben:
     - Wo brennt es?
     - Was brennt?
     - Angaben zu verletzten oder gefährdeten Personen
     - Wer meldet?
2. Personen retten:
   - * Erste Hilfe leisten
   - * Weitere Hilfe organisieren, medizinische Hilfe anfordern
   - * Gefahrenbereiche räumen; Fluchtwege benutzen;
   - * Keine Aufzüge benutzen
   - * Andere Personen warnen
   - * Sammelplatz aufsuchen (Platz vor Turmeingang BAR)
   - * Behinderten und älteren Personen helfen
3. Löschversuche unternehmen:
   - * Feuerlöscher verwenden
     (Standorte: Gänge des BAR)
   - * Fenster und Türen schließen, aber nicht abschließen
   - * Möglichst elektrischer Verbraucher abschalten

8.3. Rufnummern für Notfälle

Rettungsdienst: 112
Polizei: 110
TUD-Notruf: HA 34515
Betriebsärztin Dr. med. Friedmann-Ketzmerick: HA 36199