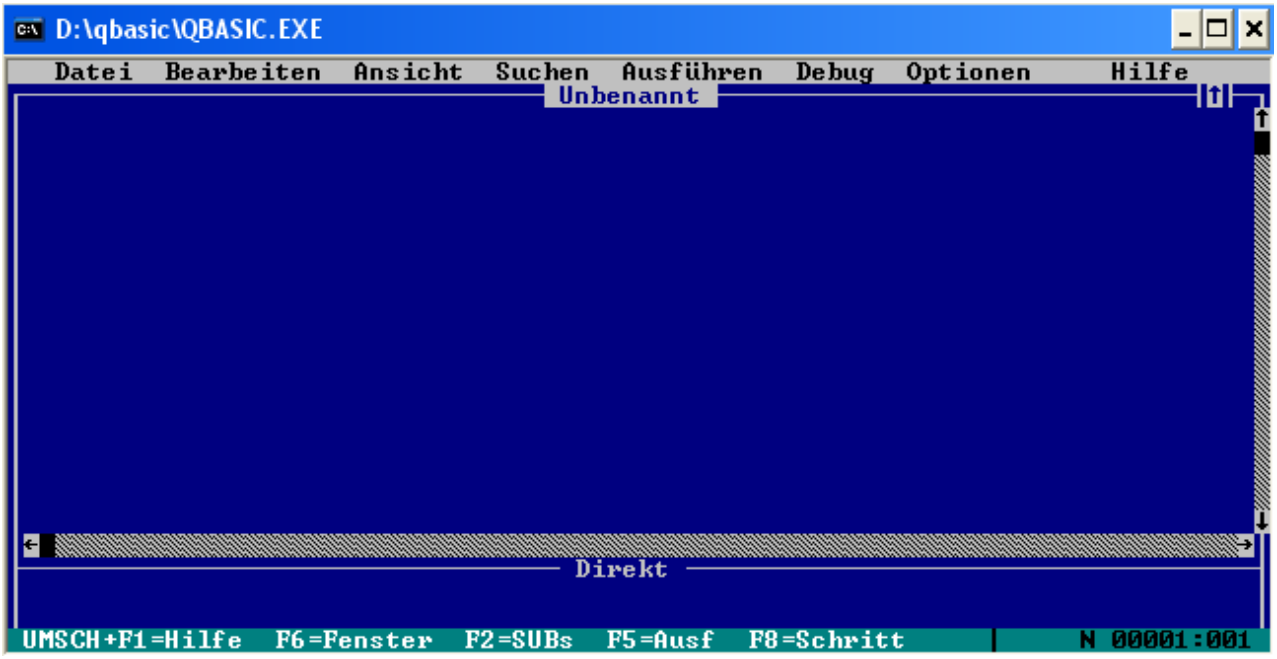
# Steuern des GSR-Interface mit QBasic




Erstellt von Jochen Sendler, Geschwister-Scholl-Realschule Mannheim

# Inhaltsverzeichnis

## 1. Die Programmiersprache QBasic

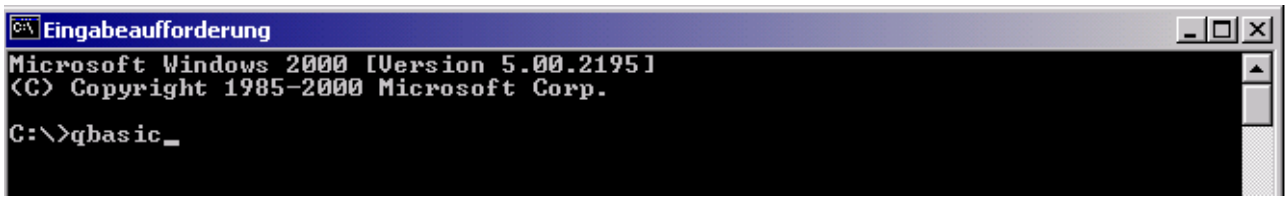
QBasic ist eine sehr alte aber dafür einfache Programmiersprache aus dem Jahre 1991. Sie wurde damals standardmäßig mit dem Betriebsystem MS-DOS mitgeliefert. Ab Windows 2000 gibt es Probleme mit dem Zugriff auf die I/O Ports (Eingabe/Ausgabe Ports) mit QBasic. Wenn man z.B. unter Windows XP die Programmiersprache QBasic startet, dann kann man zwar Programme erstellen und ablaufen lassen, hat aber keinen Zugriff auf den Parallelport und somit auf das Interface. Zwei mögliche Lösungen können Abhilfe schaffen:

1. Mit dem Tool „UserPort“ (Download über www.qbasic.de oder direkt über http://www.o-bizz.de/qbdown/qbtools/userport.zip) kann man die I/O-Ports auch unter Windows NT/2000/XP für QBasic und andere DOS-Programme zugänglich machen (Achtung: Mein PC ist dabei abgestürzt; Die Ansteuerung des Interface funktionierte danach teilweise).
2. Man erstellt sich eine DOS-BootCD mit QBasic und evtl. einem NTFS-Loader für den Zugriff auf die Festplatte (sofern sie mit NTFS und nicht mit FAT32 partitioniert wurde).

Möchte man eine aktuelle Programmiersprache verwenden, dann lässt sich das Interface auch über VB (Visual Basic) steuern. Eine entsprechende Anleitung wurde von Herrn Dinkel erstellt.

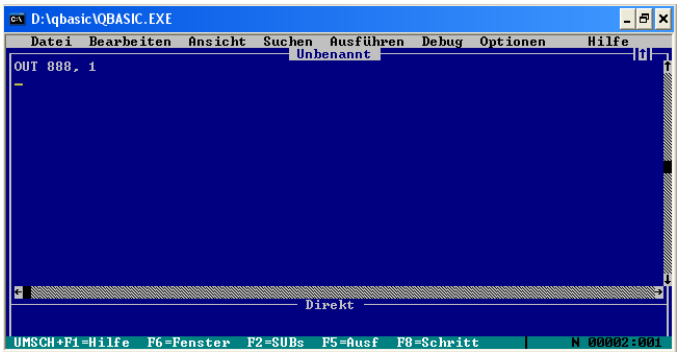
## 2. Starten von QBasic

QBasic kann man starten, indem man unter Windows XP entweder auf die QBasic.exe doppelt klickt (Variante 1 mit „UserPort“) oder bei der DOS Eingabeaufforderung (Variante 2 mit DOS-BootCD ) qbasic eingibt und mit Return bestätigt.




Den Begrüßungsdialog „Willkommen in MS-DOS-QBasic“ beendet man mit ESC.

### 3. Ansteuern einzelner LEDs




Mit dem Befehl *OUT 888, 1* lässt sich LED 1 ansteuern. Am Ende des Befehls muss man noch die Taste F5 drücken, um die Anweisung auszuführen.

Um die anderen LEDs ansteuern zu können, muss man das Dualzahlensystem kennen. Die nachfolgende Tabelle soll verdeutlichen, welche LEDs mit welcher Zahl angesteuert werden können.

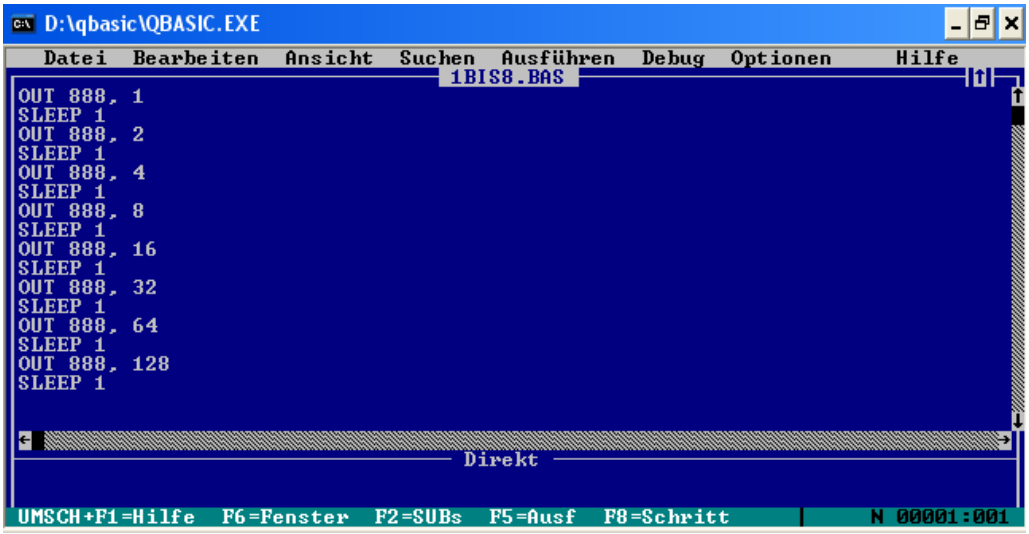
| LED 8 | LED 7 | LED 6 | LED 5 | LED 4 | LED 3 | LED 2 | LED 1 |
|---|---|---|---|---|---|---|---|
| $2^7 = 128$ | $2^6 = 64$ | $2^5 = 32$ | $2^4 = 16$ | $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^0 = 1$ |

Für die LED 2 bräuchte man also den Befehl *OUT 888, 2* für LED 3 den Befehl *OUT 888, 4* usw. Ausschalten kann man alle LEDs mit *OUT 888, 0*. Es lassen sich natürlich auch mehrere LEDs gleichzeitig ansteuern. Sollen die ersten vier LEDs leuchten, gibt man *OUT 888, 15* ein (1+2+4+8). Alle acht LEDs leuchten bei *OUT 888, 255*.

### 4. LEDs nacheinander ansteuern

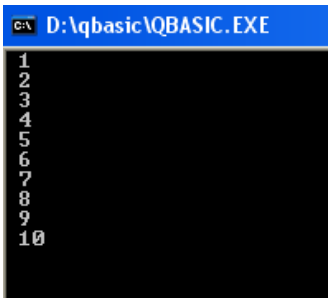
Aufgabe: „Es sollen die LEDs 1 bis 8 nacheinander für eine Sekunde leuchten und anschließend wieder ausgehen“. Dazu benötigt man den neuen Befehl *SLEEP*. Die Anweisung *SLEEP 1* lässt den Computer für eine Sekunde „schlafen“, das heißt, die Ausführung des Programms wird hier für eine Sekunde unterbrochen. Das entsprechende Programm könnte so aussehen:

```
OUT 888, 1
SLEEP 1
OUT 888, 2
SLEEP 1
OUT 888, 4
SLEEP 1
OUT 888, 8
SLEEP 1
OUT 888, 16
SLEEP 1
OUT 888, 32
SLEEP 1
OUT 888, 64
SLEEP 1
OUT 888, 128
SLEEP 1
```



## 5. Die FOR … NEXT-Schleife

Bevor wir das Programm aus Kapitel 4 mit seinen 16 Anweisungen mit Hilfe der FOR … NEXT-Schleife erheblich verkürzen, wollen wir uns die Funktion der FOR … NEXT-Schleife an einem einfachen Beispiel etwas genauer anschauen.

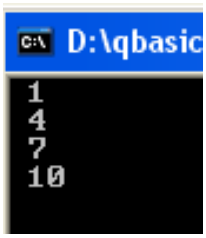


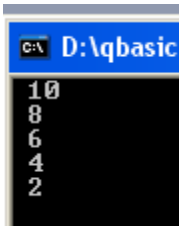

*FOR zaehler = 1 TO 10* ← Schleifenkopf
*PRINT zaehler* ← Anweisung(en)
*NEXT zaehler* ← Schleifenende

Was passiert? Mit Hilfe der FOR … NEXT-Schleife und des PRINT-Befehls kann man die Zahlen von 1 bis 10 auf dem Monitor ausgeben. Bei einer FOR … NEXT-Schleife (auch Zählschleife genannt) ist von Anfang an klar, wie oft die Anweisung(en) zwischen dem Schleifenkopf (FOR) und dem Schleifenende (NEXT) ausgeführt werden. Im Schleifenkopf *FOR zaehler = 1 TO 10* wird der Variablen *zaehler* der Startwert 1 zugewiesen. Eine Variable kann man sich wie ein kleines „Körbchen“ vorstellen, in das man etwas hineinlegt. Der Endwert der Schleife wird auf 10 gesetzt. Damit gibt man an, wie oft die Schleife durchlaufen werden soll (von 1 bis 10, also 10 Mal). Im Schleifenkopf wird jede Runde überprüft, ob die Schleife verlassen werden soll, nämlich dann, wenn der Inhalt der Variablen *zaehler* größer als 10 ist. Da dies im ersten Durchlauf noch nicht der Fall ist, folgt die Anweisung in der Schleife, der Printbefehl. Jetzt wird auf dem Monitor der Inhalt der Variablen *zaehler* ausgegeben. Die Zahl 1. Anschließend folgt die Anweisung *NEXT zaehler*. Hier wird die Variable *zaehler* um 1 erhöht und es folgt ein Sprung hoch zum Schleifenkopf. Dort wird überprüft, ob der Inhalt von *zaehler* schon größer als 10 ist (zaehler hat den Inhalt 2). Da dies noch nicht der Fall ist, folgt der Printbefehl mit der Monitorausgabe 2. Bei *NEXT zaehler* wird die Variable *zaehler* wieder um 1 erhöht (ist jetzt also 3) und es folgt der Sprung zum Schleifenkopf. Dies wiederholt sich solange bis der Inhalt von *zaehler* größer als 10 ist. Dann wird die Schleife sofort verlassen. Befehle zwischen *FOR* und *NEXT* sollen eingerückt eingegeben werden.

Die Schrittweite, also der Wert um den die Variable *zaehler* erhöht wird, ist standardmäßig 1. Soll eine andere Schrittweite genommen werden, braucht man den Befehl *STEP* im Schleifenkopf.

```
FOR zaehler = 1 TO 10 STEP 3
  PRINT zaehler
NEXT zaehler
```




Natürlich kann die Schrittweite auch negativ sein (z.B. -2), wenn “runtergezählt” werden soll.

## 6. Mit der FOR … NEXT-Schleife LEDs nacheinander ansteuern

Jetzt können wir die FOR … NEXT-Schleife nutzen, um die LEDs leuchten zu lassen. Dazu brauchen wir aber nacheinander nicht die Zahlen von 1 bis 8 sondern 1, 2, 4, 8,16, 32 usw.

| LED 8 | LED 7 | LED 6 | LED 5 | LED 4 | LED 3 | LED 2 | LED 1 |
|---|---|---|---|---|---|---|---|
| $2^7 = 128$ | $2^6 = 64$ | $2^5 = 32$ | $2^4 = 16$ | $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^0 = 1$ |

Wie können diese Zahlen erzeugt werden? Ein Blick auf die Tabelle zeigt die Lösung. Man nimmt als *zaehler* die Hochzahlen von 0 bis 7. Die Schleife könnte dann so aussehen:

```
FOR zaehler = 0 TO 7
  OUT 888, 2 ^ zaehler
  SLEEP 1
NEXT zaehler
```

Die nachfolgende Tabelle soll die Funktion der Schleife verdeutlichen. Das „Hochzeichen“ befindet sich unterhalb der Esc-Taste. Man drückt erst die „Hochzeichen-Taste“ und dann die Leertaste.

| LEDs | zaehler | 2^zaehler |
|---|---|---|
| LED 1 | 0 | $2^0 = 1$ |
| LED 2 | 1 | $2^1 = 2$ |
| LED 3 | 2 | $2^2 = 4$ |
| LED 4 | 3 | $2^3 = 8$ |
| LED 5 | 4 | $2^4 = 16$ |
| LED 6 | 5 | $2^5 = 32$ |
| LED 7 | 6 | $2^6 = 64$ |
| LED 8 | 7 | $2^7 = 128$ |

## 7. Die DO … LOOP-Schleife

Die DO … LOOP-Schleife ist eine sog. Endlosschleife. Wird keine Abbruchbedingung formuliert oder erreicht, werden die Anweisungen zwischen DO und LOOP endlos wiederholt.

| | |
|---|---|
| **Let zahl = 1** | Der Variablen zahl wird der Wert 1 zugewiesen |
| **CLS** | CLS (Clear Screen) löscht den Bildschirm |
| **DO** | Der Schleifenkopf |
| **Print zahl** | Der Inhalt der Variablen zahl wird auf dem Bildschirm ausgegeben |
| **zahl = zahl +1** | Der Wert der Variablen zahl wird um 1 erhöht |
| **LOOP until zahl = 100** | Das Schleifenende mit Abbruchbedingung. Die Anweisungen in der Schleife werden wiederholt bis zahl = 100 ist |

## 8. Die INKEY$-Funktion

Die INKEY-Funktion speichert die letzte Taste, die gedrückt wurde. Da man die INKEY-Funktion nicht direkt abfragen kann, muss deren Inhalt einer STRING-Variablen (eine Variable, in der man Zeichen speichern kann) zugewiesen werden. Z.B. so: a$ = INKEY$

```
LET zahl = 1
CLS
DO
   PRINT zahl
   zahl = zahl + 1
   a$ = INKEY$
LOOP UNTIL a$ = "e"
```

Jetzt kann man eine Abbruchbedingung in einer DO … LOOP-Schleife mit *a$* formulieren. Die Anweisungen in der DO … LOOP-Schleife werden wiederholt bis die Taste e gedrückt wird.

Will man die Schleife bei irgendeinem Tastendruck verlassen, kann man die Abbruchbedingung wie in dem Beispiel rechts formulieren: while a$ = ’’’’ [solange a$ leer ist].

```
LET zahl = 1
CLS
DO
   PRINT zahl
   zahl = zahl + 1
   a$ = INKEY$
LOOP WHILE a$ = ""
```

Die Abbruchbedingung kann auch im Schleifenkopf hinter DO stehen.

## 9. Der INPUT-Befehl

Mit dem INPUT-Befehl kann man, wie beim PRINT-Befehl, einen Text auf dem Bildschirm ausgeben. Anschließend jedoch erwartet der Computer eine Tastatureingabe, die in einer Zahlen- oder Stringvariablen gespeichert wird. Z.B. so:

```
CLS
INPUT "Bitte geben Sie eine Zahl ein, die quadriert werden soll: ", zahl
PRINT "Das Quadrat von"; zahl; "ist"; zahl ^ 2
```

Auf dem Bildschirm erscheint der Text *Bitte geben Sie eine Zahl ein, die quadriert werde soll:* und der Cursor wartet auf eine Tastatureingabe, die mit der Return-Taste abgeschlossen werden muss. Dann wird diese Ziffer in der Zahlenvariable *zahl* gespeichert. In der nächsten Zeile gibt der PRINT-Befehl erst den Text *Das Quadrat von* aus, dann den Inhalt der Variablen *zahl*, anschließend das Wort *ist* und zum Schluss das Quadrat der eingegebenen Zahl.

```
Bitte geben Sie eine Zahl ein, die quadriert werden soll: 9
Das Quadrat von 9 ist 81
```

Verwendet man bei dem PRINT-Befehl ein Semikolon (Strichpunkt), dann werden Bildschirmausgaben lückenlos aneinander gereiht. Bei Kommas erfolgt die Ausgabe eine Spalte weiter.

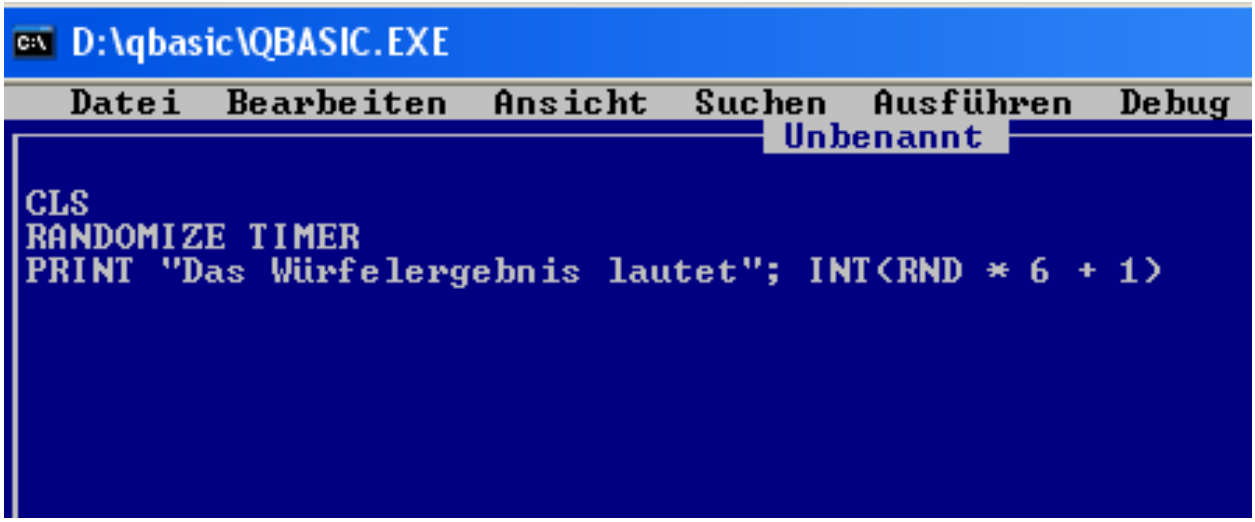
### 10. IF … THEN … ELSE (Wenn … Dann … Sonst)

Manchmal sollen Programme nicht nur linear ablaufen (Anweisung für Anweisung), sondern verzweigen, wenn Entscheidungen getroffen werden. Hier ein Beispiel:


Die IF … THEN-Anweisung funktioniert alleine, kann aber um die Anweisung ELSE erweitert werden. Benötigt eine einzelne IF … THEN … ELSE-Anweisung mehrere Zeilen, dann muss sie mit END IF abgeschlossen werden.

### 11. Zufallszahlen erzeugen (RANDOMIZE TIMER, RND, INT)

Mit dem Befehl RANDOMIZE TIMER wird ein Zufallsgenerator gestartet. RND erzeugt eine Zahl zwischen 0 und 1 (z.B. 0.5678). Mit dem INT-Befehl kann man die Kommazahl in eine ganze Zahl umwandeln.



## 12. Der ASC ( )-Befehl

Der ASC( )-Befehl wandelt das erste Zeichen einer Stringvariablen in den entsprechenden Zahlenwert der ASCII-Code-Tabelle um. So erzeugt ASC(''Z'') die Zahl 90.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | NUL | 033 | ! | 066 | B | 099 | c | 132 | ä | 165 | Ñ | 198 | ã | 231 | þ | | | | |
| 001 | Start Of Header | 034 | " | 067 | C | 100 | d | 133 | à | 166 | ª | 199 | Ã | 232 | Þ | | | | |
| 002 | Start Of Text | 035 | # | 068 | D | 101 | e | 134 | å | 167 | º | 200 | ╚ | 233 | Ú | | | | |
| 003 | End Of Text | 036 | $ | 069 | E | 102 | f | 135 | ç | 168 | ¿ | 201 | ╔ | 234 | Û | | | | |
| 004 | End Of Transmission | 037 | % | 070 | F | 103 | g | 136 | ê | 169 | ® | 202 | ╩ | 235 | Ù | | | | |
| 005 | Enquiry | 038 | & | 071 | G | 104 | h | 137 | ë | 170 | ¬ | 203 | ╦ | 236 | ý | | | | |
| 006 | Acknowledge | 039 | | 072 | H | 105 | i | 138 | è | 171 | ½ | 204 | ╠ | 237 | Ý | | | | |
| 007 | Bell | 040 | ( | 073 | I | 106 | j | 139 | ï | 172 | ¼ | 205 | ═ | 238 | ¯ | | | | |
| 008 | Backspace | 041 | ) | 074 | J | 107 | k | 140 | î | 173 | ¡ | 206 | ╬ | 239 | ´ | | | | |
| 009 | Horizontal Tab | 042 | * | 075 | K | 108 | l | 141 | ì | 174 | « | 207 | ¤ | 240 | - | | | | |
| 010 | Line Feed | 043 | + | 076 | L | 109 | m | 142 | Ä | 175 | » | 208 | ð | 241 | ± | | | | |
| 011 | Vertical Tab | 044 | , | 077 | M | 110 | n | 143 | Å | 176 | ░ | 209 | Ð | 242 | _ | | | | |
| 012 | Form Feed | 045 | - | 078 | N | 111 | o | 144 | É | 177 | ▒ | 210 | Ê | 243 | ¾ | | | | |
| 013 | Carriage Return | 046 | . | 079 | O | 112 | p | 145 | æ | 178 | ▓ | 211 | Ë | 244 | ¶ | | | | |
| 014 | Shift Out | 047 | / | 080 | P | 113 | q | 146 | Æ | 179 | │ | 212 | È | 245 | § | | | | |
| 015 | Shift In | 048 | 0 | 081 | Q | 114 | r | 147 | ô | 180 | ┤ | 213 | ı | 246 | ÷ | | | | |
| 016 | Delete | 049 | 1 | 082 | R | 115 | s | 148 | ö | 181 | Á | 214 | Í | 247 | ¸ | | | | |
| 017 | -- frei -- | 050 | 2 | 083 | S | 116 | t | 149 | ò | 182 | Â | 215 | Î | 248 | ° | | | | |
| 018 | -- frei -- | 051 | 3 | 084 | T | 117 | u | 150 | û | 183 | À | 216 | Ï | 249 | ¨ | | | | |
| 019 | -- frei -- | 052 | 4 | 085 | U | 118 | v | 151 | ù | 184 | © | 217 | ┘ | 250 | · | | | | |
| 020 | -- frei -- | 053 | 5 | 086 | V | 119 | w | 152 | ÿ | 185 | ╣ | 218 | ┌ | 251 | ¹ | | | | |
| 021 | Negative Acknowledge | 054 | 6 | 087 | W | 120 | x | 153 | Ö | 186 | ║ | 219 | █ | 252 | ³ | | | | |
| 022 | Synchronous Idle | 055 | 7 | 088 | X | 121 | y | 154 | Ü | 187 | ╗ | 220 | ▄ | 253 | ² | | | | |
| 023 | End Of Transmission Block | 056 | 8 | 089 | Y | 122 | z | 155 | ø | 188 | ╝ | 221 | ¦ | 254 | ■ | | | | |
| 024 | Cancel | 057 | 9 | 090 | Z | 123 | { | 156 | £ | 189 | ¢ | 222 | Ì | 255 | | | | | |
| 025 | End Of Medium | 058 | : | 091 | [ | 124 | \| | 157 | Ø | 190 | ¥ | 223 | ▀ | | | | | | |
| 026 | Substitude | 059 | ; | 092 | \ | 125 | } | 158 | × | 191 | ┐ | 224 | Ó | | | | | | |
| 027 | Escape | 060 | < | 093 | ] | 126 | ~ | 159 | ƒ | 192 | └ | 225 | ß | | | | | | |
| 028 | File Seperator | 061 | = | 094 | ^ | 127 | ⌂ | 160 | á | 193 | ┴ | 226 | Ô | | | | | | |
| 029 | Group Seperator | 062 | > | 095 | _ | 128 | Ç | 161 | í | 194 | ┬ | 227 | Ò | | | | | | |
| 030 | Record Seperator | 063 | ? | 096 | ` | 129 | ü | 162 | ó | 195 | ├ | 228 | õ | | | | | | |
| 031 | Unit Seperator | 064 | @ | 097 | a | 130 | é | 163 | ú | 196 | ─ | 229 | Õ | | | | | | |
| 032 | | 065 | A | 098 | b | 131 | â | 164 | ñ | 197 | ┼ | 230 | µ | | | | | | |

Das Programm aus Kapitel 10 kann mit Hilfe des ASC( )-Befehls verbessert werden. Will man sicher sein, dass nur Zahlen von 1 bis 8 eingegeben werden und nicht ausversehen Buchstaben, dann muss man den INPUT-Befehl in die DO … LOOP-Schleife packen. Dieser wird solange wiederholt, solange die Bedingung in der LOOP-Anweisung erfüllt ist. Der Zahlenwert von ASC(eingabe$) muss entweder < (kleiner) als 49 sein oder > (größer) als 56. Das wäre der Fall, wenn man keine Zahl von 1 bis 8 eingibt.

ASC.BAS

```
CLS
DO
 CLS
 INPUT "Welche LED soll leuchten? ", eingabe$
LOOP WHILE ASC(eingabe$) < 49 OR ASC(eingabe$) > 56
led = ASC(eingabe$) - 48

IF led = 1 THEN ledhell = 1
IF led = 2 THEN ledhell = 2
IF led = 3 THEN ledhell = 4
IF led = 4 THEN ledhell = 8
IF led = 5 THEN ledhell = 16
IF led = 6 THEN ledhell = 32
IF led = 7 THEN ledhell = 64
IF led = 8 THEN ledhell = 128
OUT 888, ledhell
```

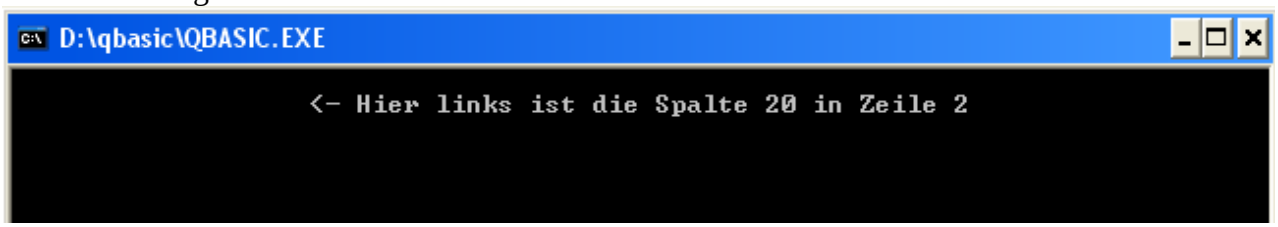
## 13. Positionsgenaue Bildschirmausgabe mit der TAB( )-Funktion

Um eine übersichtliche und genaue Bildschirmausgabe mit PRINT oder INPUT zu erhalten, kann man die TAB( )-Funktion verwenden. In QBasic ist der Bildschirm in 80 Spalten und 25 Zeilen aufgeteilt.

TAB.BAS

```
CLS
PRINT
PRINT TAB(20); "<- Hier links ist die Spalte 20 in Zeile 2"
```

So sieht das Ergebnis aus:




## 14. Abfrage der Eingabeleitungen


## 15. Aufgaben für die Programmierung des GSR-Interface

a) Die LEDs 1 bis 8 sollen nacheinander für 1 Sekunde leuchten.

b) Der Benutzer soll die Leuchtdauer der LEDs in Aufgabe a) über den Bildschirm selbst bestimmen können.

c) Es sollen nacheinander die LEDs 1 bis 8 angehen und leuchten.

d) Die LEDs 1 bis 8 sollen nacheinander für 1 Sekunde leuchten. Nachdem LED 8 geleuchtet hat, sollen wieder die LEDs 7 bis 1 für 1 Sekunde leuchten.

e) Die LEDs in Aufgabe d) sollen solange leuchten bis die Taste E gedrückt wird.

f) Es sollen immer zwei LEDs gleichzeitig leuchten. LED 1+8, LED 7+2, LED 6+3 usw..

g) Das Programm aus Aufgabe f) soll jederzeit mit der E-Taste unterbrochen werden können.

h) Über den Bildschirm soll der Benutzer eine Zahl von 1 bis 6 raten, die mit einem zufälligen Würfelergebnis verglichen wird.

i) Nach Eingabe einer Note als Zahl (z.B. 3) soll die „Verbalnote“ (hier: befriedigend) erscheinen.