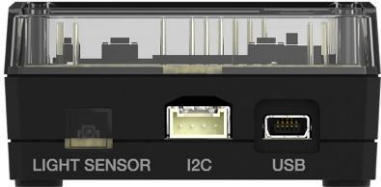





Lektion 3: Helligkeit, If und While	Übung 1: Messung der Helligkeit	
<p>In der ersten Stunde der Lektion 3 wird der eingebaute Lichtsensor (HELLIGKEIT) untersucht und der Befehl Output(des TI-Basic eingeführt, um eine Technik zu zeigen, mit der man Zahlen verschiedener Länge sehr wirkungsvoll auf dem Display darstellen kann.</p>	<p>Lernziele:</p> <ul style="list-style-type: none"> • Auslesen des Helligkeit-Sensors. • Einführung der While – Schleife. • Verwendung des Befehls Output(. • Einführung des Befehls toString(und der Verkettung. 	
<p>In den vorangegangenen Stunden wurden nur Befehle an den TI-Innovator™ Hub gesendet, die einen Einfluss auf die eingebauten Geräte LIGHT, COLOR und SOUND haben.</p> <p>In dieser Lektion wird nun der eingebaute Lichtsensor verwendet, und die von ihm gelieferten Werte sollen verwendet werden, um die Helligkeit (in ungeeichten Werten) anzuzeigen. Der Lichtsensor liefert Werte im Bereich von 0 bis 100 in Form von Dezimalzahlen.</p> <p>Man benötigt ZWEI Befehle, um den Lichtsensor auszulesen:</p> <ul style="list-style-type: none"> • Send("READ BRIGHTNESS") • Get(B) 		
<p>Erste Programmschritte:</p> <ol style="list-style-type: none"> 1. Man legt ein neues Programm mit dem Namen HELL1 an. 2. Die Befehle ClrHome und Disp werden hinzugefügt, um die Überschrift wie abgebildet zu erzeugen. 3. Über die Taste <code>[prgm]</code> wird das Menü HUB ausgewählt. 4. Es wird erst 2: Send("READ... und dann 1: BRIGHTNESS ausgewählt. 5. Erneutes Drücken von <code>[prgm]</code> und Wahl des Menüs HUB. 6. Auswahl von 5: Get(und Vervollständigung durch B). 		
<p>Funktionsweise:</p> <ul style="list-style-type: none"> • READ BRIGHTNESS veranlasst den TI-Innovator™ Hub den Wert der Helligkeit zu lesen und in einem eingebauten "Buffer" (Puffer) zu speichern. • Get(B) ist ein Befehl, mit dem der im Puffer auf dem TI-Innovator™ Hub gespeicherte Wert an den TI-84 Plus CE-T weitergegeben und in einer Variablen gespeichert wird. Zulässig sind die Variablennamen A..Z und Θ (Theta); im Beispiel hat die Variable den Namen B. 		
<p>Hinweis: Ein "Buffer"(Puffer) ist ein Speicherbereich auf dem TI-Innovator™ Hub, in dem zeitweise ein Wert gespeichert wird, bis ein neuer READ-Befehl eintrifft. Deshalb ist es sehr sinnvoll, unmittelbar auf den READ-Befehl einen GET-Befehl folgen zu lassen, damit der Wert einer Variablen auf dem Taschenrechner zugewiesen wird. Es ist möglich mehrere Werte zugleich vom TI-Innovator™ Hub auszulesen und in einer Liste für die weitere Bearbeitung auf dem Taschenrechner zu speichern, aber das soll in dieser Einführung nicht weiter verfolgt werden.</p>		
<p>Die While - Schleife:</p> <p>Die While...End - Schleife (<code>[prgm]</code> STRG Menü) wird verwendet, um eine Gruppe von Befehlen durchzuführen solange eine <i>Bedingung</i> wahr ist. Eine <i>Bedingung</i> ist ein logischer Ausdruck, der entweder wahr oder falsch ist. Im Menü test (2nd</p>		



math) findet man die dazu notwendigen relationalen und logischen Operatoren:

- Die relationalen Operatoren sind =, ≠, <, >, ≤, and ≥.
- Die logischen Operatoren sind **und**, **oder**, **xoder** und **nicht**(.

Die Operatoren können verwendet werden, um z.B. Bedingungen wie **x>0** und **y>0** zu bilden.

Eine einfache **While** – Schleife wird im Programm verwendet. Sie bricht ab, sobald der Helligkeitswert kleiner als 1 ist. Man kann damit das Programm beenden, indem man einfach den Helligkeitssensor mit der Hand abdeckt.

Eine andere Möglichkeit, ein laufendes Programm abzubrechen, hat man, indem man die Taste **ON** drückt. Dann wird eine Fehlermeldung ausgegeben: **Fehler: Abbruch** mit den Optionen **1:Abbruch** und zurück zum Startbildschirm oder **2:Gehezu** der Anweisung im Programmeditor, wo die Taste **ON** das Programm beendet hat. Das ist ein bequemer Weg, um Fehler aufzuspüren und das Programm weiter zu editieren.

```
NORMAL FLS AUTO REELL BOGENM MP
FEHLER: ABBRUCH
1:Abbruch
2:Gehezu
Aktion gestoppt.
```

Hinzufügen von While – Schleife und Output(:

1. Vor dem Befehl **Send(** werden die folgenden zwei Befehle eingefügt:

:2→B

:While B>1 [> aus dem Menü **test**]

- Solange die *Bedingung* **B > 1** wahr ist, wird der Lichtsensor fortlaufend ausgelesen. Wird sie falsch, wenn nicht mehr genug Licht den Sensor trifft (etwa weil er durch eine Hand abgedeckt wird), bricht die Schleife ab und das Programm endet. Der erste Befehl weist **B** den definierten Wert 2 zu, um die Schleife zu starten.

2. Wie die **For** – Schleife benötigt auch die **While** - Schleife ein **End**, das hinter dem Befehl **Get(B)** eingefügt wird.

```
NORMAL FLS AUTO REELL BOGENM MP
PROGRAM:HELL1
:ClrHome
:Disp "HELLIGKEIT"
:2→B
:While B>1
:Send("READ BRIGHTNESS")
:Get(B)
:
:
:End
```

Hinweis: Es ist immer gut, wenn man in Programmen dafür sorgt, dass es einen Weg aus einer Schleife gibt, damit keine Endlosschleifen entstehen. Man sollte bedenken, dass **End** nicht das Ende des Programmes ist sondern lediglich das Ende einer Steuerungs-Struktur wie **If...Then...End**, **For...End**, **While...End** und **Repeat...End**. In größeren Programmen wird **End** auch auf mehreren Ebenen in verschachtelten Strukturen verwendet. Der Prozessor „weiß“, welches **End** zu welcher Struktur gehört, und der Programmierer muss den richtigen Programmcode dazu schreiben.

3. Unmittelbar *nach* dem Befehl **Get(** wird der Befehl **Output(** hinzugefügt, den man über die Taste **[prgm]** im Menü **E/A** bei **6: Output(** findet.

- Der Befehl **Output(** ermöglicht es genau festzulegen wo auf dem Display etwas angezeigt werden soll. Die Syntax ist:
- **Output(<Zeilennummer>, <Spaltennummer>, <Text oder Variable>)**

Beispiele:

- **Output(3,7,“HALLO”)** setzt den Buchstaben **“H”** auf die Zeile 3 und Spalte 7 des Displays. Der Rest des Wortes folgt dem **H**.
- **Output(5,10,B)** setzt den Wert der Variablen **B** so, dass die erste Ziffer in Zeile 5 Spalte 7 beginnt – der Rest folgt.

```
NORMAL FLS AUTO REELL BOGENM MP
PROGRAM:HELL1
:ClrHome
:Disp "HELLIGKEIT"
:2→B
:While B>1
:Send("READ BRIGHTNESS")
:Get(B)
:
:Output(5,10,B)
:End
```



4. Nun kann man den Editor verlassen und das Programm starten.
- Man sieht die Überschrift und einen Wert in der Mitte des Displays, der sich mit der Lichtintensität ändert, die der Sensor ausliest. Aber: die angezeigten Zahlen müssen nicht richtig sein!

Der Befehl **Output(** löscht keine nachfolgenden Zeichen, wenn eine kürzere Zahl dargestellt wird. Ist z.B. die dargestellte Zahl 1.23456 und die nächste 77, so erhält man 7723456. Aber man kann dieses Problem durch eine kleine Modifikation des Programmes lösen.



Dazu wird die Variable B in eine Zeichenkette (string) verwandelt, zu der noch ein paar Leerzeichen addiert werden, so dass der komplette vorherige Wert auf dem Display durch Überschreiben mit Leerzeichen ausgelöscht wird.

Man findet den Befehl **toString(** im Menü **E/A**.

Die endgültige Fassung des Befehles Output(ist:

Output(5,10,toString(B)+" ")

Es wurden 10 Leerzeichen zwischen den Anführungsstrichen eingefügt.

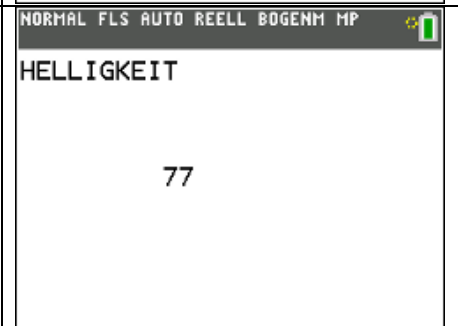


Lässt man nun das Programm laufen, so sieht man, dass einige Werte kürzer sind als andere, da die Leerzeichen, die zu der *Zeichenkette* B addiert wurden, die vorherigen Zeichen überschrieben haben.

Verkettung:

Das im Befehl **Output(** verwendete **+** – Zeichen ist nicht im Sinne einer üblichen Addition zu verstehen. Hier verkettet (hängt aneinander) es zwei Zeichenketten.

Die Leerzeichen in den Anführungsstrichen werden an das Ende der Zeichenkette B angehängt.



Hinweis: Stringoperationen werden für die Beispiele eigentlich nicht benötigt, aber man sollte sich klarmachen, wie durch die Möglichkeiten, die in einer Programmiersprache stecken, die Programmierung erheblich erweitert werden kann.