Lisa Obermaier
Simon Thum

05.06.2019

4. QNX, IO: LEDs

a) *Benutzereingaben*

Um die Benutzereingaben abzufangen, definieren wir einen *char*, der mit *getc(stdin)* in einer Schleife eingelesen wird, solange kein *q*, für *quit*, gelesen wird. Weitere valide Eingaben sind *1* und *2*, um die ersten beiden LEDs ein- und auszuschalten. Bevor das Byte für die physischen LEDs befüllt wird, berechnen wir die logischen Werte der LEDs in den Variablen *led0* bis *led3*. *led0* und *led1* werden eingeschaltet, wenn die Keys *1* bzw. *2* gedrückt werden. *led2* wird eingeschaltet, wenn nur eine der beiden *led0* oder *led1* leuchten (XOR), *led3* wird eingeschaltet, solange eine der beiden *led0* oder *led1* leuchten (OR). Die Ausgabe am Ende der Schleife gibt die Berechnung der LEDs wieder und spiegelt eine eingeschaltetes LED mit *1* und ein ausgeschaltetes LED mit *0* wieder.

b) *Leuchtdioden auf dem Device*

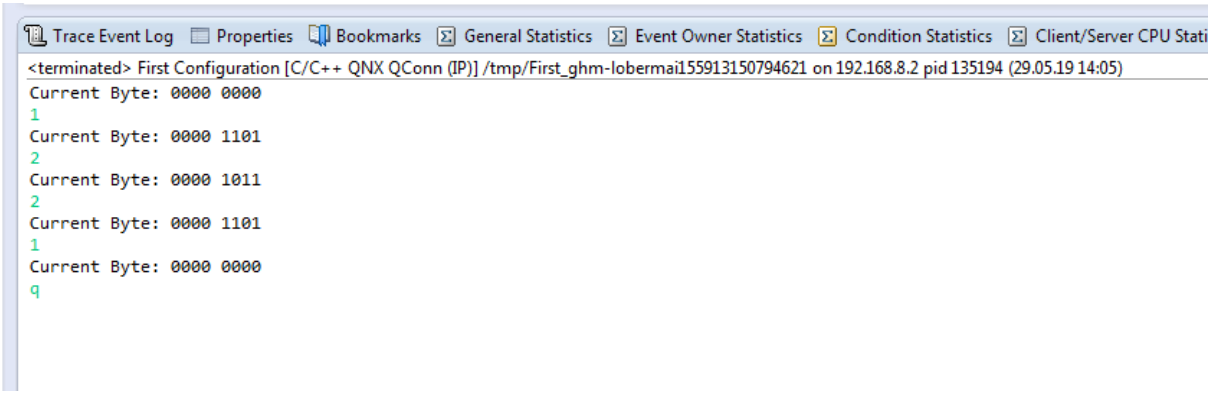
Die Leuchtdioden werden angesprochen mit einem Handler *fh*, der mit dem Befehl *open* und der Eigenschaft *write only* die Leuchtdioden in *„/dev/leds“* ansprechen kann. Der *char by* wird verwendet, um den Wert an/aus der Leuchtdioden an den *Beagle Bone* zu senden. Nach der Berechnung der vier Werte und deren Speicherung in *by* können diese mit Hilfe von *write(fh, &by, sizeof(by))* an die Leuchtdioden gesendet werden.

Wird mit *q* die while-Schleife beendet, wird dieser mit *close* auf den Handler *fh* beendet.

Um die einzelnen Bits in *by* zu setzen, werden die C-Bitshift-Operationen und logische Operatoren benutzt. Die standardmäßig auf 0 gesetzten Bits für die ersten beiden LEDs können einfach mit einer 1 verXORt werden, dies geschieht mit der Zuweisung ^=.

Die 1 ist stets als *unsigned long* deklariert, dies ist in diesem Kontext nicht notwendig, verhindert aber Overflows, sollte der Quelltext auf längere Bytemuster angewendet werden.
Ebenso ist es nicht notwendig, das erste Bit von *by* um 0 zu shiften, dies ist als Erläuterung der Position innerhalb von *by* zu sehen. Es ist davon auszugehen, dass der C-Compiler einen Nullshift ebenso wegoptimiert wie eine Multiplikation mit 1.

Für die direkte Zuweisung von 1 an der Stelle der dritten und vierten Bits wird *by* erst mit -1 geXORt, dann mit einer an die gewünschte Stelle geshifteten 1 verANDet, und das Ergebnis wieder mit dem ursprünglich *by* geXORt. Dies hat den Effekt, dass die anderen Bits nicht verändert werden, bis auf die geshiftete Stelle. Diese ist nun durch das XOR garantiert auf 1 gesetzt, da das erste XOR in Kombination mit dem AND ja berechnet hat, welcher Wert an der Stelle stehen muss, damit ein weiteres XOR eine 1 erzeugt.
Entsprechend kann mit einem ersten XOR mit 0 statt -1 garantiert werden, dass das Bit an der geshifteten Stelle 0 wird.


Trace Event Log
Properties
Bookmarks
General Statistics
Event Owner Statistics
Condition Statistics
Client/Server CPU Stati
<terminated> First Configuration [C/C++ QNX QConn (IP)] /tmp/First_ghm-lobermai155913150794621 on 192.168.8.2 pid 135194 (29.05.19 14:05)
Current Byte: 0000 0000
1
Current Byte: 0000 1101
2
Current Byte: 0000 1011
2
Current Byte: 0000 1101
1
Current Byte: 0000 0000
q