Externer Interrupt

Aufgabe 1: Was wäre wenn es keine Klingel gäbe?

Situation: Ihr seid zuhause beim Bügeln und erwartet wichtigen Besuch. Eine genaue Uhrzeit habt Ihr nicht vereinbart. Euer Besuch kann telefonisch, oder sonst wie, nicht erreicht werden. Bügeln muss aber dringend sein.

Szenario 1: Alle Klingel (Haustür, Handy usw.) funktionieren

Szenario 2: Alle Klingeln sind ausgefallen.

Diskutiert im Plenum folgende Fragen:  
a) Wie verhaltet Ihr Euch in beiden Szenarien?

b) Bei einem **Interrupt** reagiert Ihr auf ein Ereignis, beim **Polling** müsst Ihr ständig die Eingänge prüfen. Begründet, wie Ihr die beiden Szenarien zuordnet.

c) Gebt für beide Verfahren, Polling und Interrupt, Vor- und Nachteile an.

Aufgabe 2: Schauen Sie sich die Videos an und lösen Sie folgende Aufgaben mit Externem Interrupt

1. Bei einem Tastendruck an PA\_1 soll eine ihren Zustand wechseln: Aus -> Taste -> An -> Taste -> Aus …
2. Bei Tastendruck an PA\_1 soll die LED eingeschaltet werden, bei loslassen soll sie wieder ausgehen
3. Bei Tastendruck auf Taste PA\_1 soll die LED angehen, bei Tastendruck auf Taste PA\_6 soll sie wieder ausgehen.
4. Bei Tastendruck auf Taste PA\_1 soll GPIOC PortC hochzählen, bei Tastendruck auf Taste PA\_6 soll PortC herunterzählen
5. Reaktionsspiel: Im Hauptprogramm wandert eine 1, zyklisch wiederholt von PC\_0 nach PC\_7 und wieder zurück. Die Wartezeit soll jeweils 0,25 Sekunden betragen.

Interrupt-Service-Routine:

Wenn bei Tastendruck an PA\_1 gerade PC\_3 leuchtet soll ein Trefferzähler int treffer=0; hochgezählt werden. Bei einem Fehlschuss wird die Trefferzahl um 1 heruntergezählt. Minimum=0. Die Anzahl der Treffer soll auf dem LCD-Display angezeigt werden. Beachten Sie: PA\_1 benötigt PullDown.

Beachten Sie weiterhin: Die Deklarationen von Ports, treffer, LCD müssen global sein;  
#include "mbed.h"

#include "platform/mbed\_thread.h"

#include "LCD.h"

// Blinking rate in milliseconds

#define BLINKING\_RATE\_MS 250

//globale Deklarationen

int treffer=0;

lcd mylcd;

PortOut Lauflicht(PortC,0xFF);

…

1. Ampel: Jeweils 3 LED: Rot, gelb und grün an PortC bilden die Ampeln von Haupt- und Nebenstraße.

In Hauptprogramm werden nacheinander, zyklisch wiederholt, zunächst die Hauptstraße, dann die Nebenstraße rot, rot-gelb-grün-gelb-rot geschaltet. Die übrigen beiden LEDs rot und grün bilden die Fußgängerampeln. Die ISR der Taste PA\_1 (mit PullDown) setzt eine globale bool-Variable bool gedrueckt=false; auf true. Wenn gegrueckt == true, wird im Hauptprogramm, an passender Stelle, die Fußgängerampel auf grün und wieder auf rot geschaltet. Die globale Variable gedrueckt muss dabei natürlich wieder auf false gesetzt werden.

Aufgabe 3: Kontrollfragen:

1. Wie wird ein Interruptprogramm (ISR) aufgerufen?
2. Wie wird der PA\_6 als Interruptquelle festgelegt?
3. Wie legen Sie ein Unterprogramm als Interruptserviceroutine (ISR) für PA\_6 steigende Flanke fest? Wie für fallende Flanke?
4. Welche Auswirkung hat das Eintreten eines Interruptereignisses auf das Hauptprogramm?
5. Wo im Programm steht die Interruptinitialisierung?
6. Wie wird der Interrupt freigegeben?
7. Welche Wirkung hat die Interruptfreigabe?
8. Wie wird der Interrupt wieder gesperrt?

Anhang 1 Interrupt-Anweisungen:

|  |  |
| --- | --- |
| Anweisung | Bedeutung |
| InterruptIn name(Port); | Ein Port wird als Interruptquelle mit Namen *name* festgelegt Beispiel: InterruptIn taste(PA\_1); |
| name.mode(PullDown); | Der Interrupteingang bekommt einen PullDown |
| name.rise(&isr) | Bei einer **steigenden** Flanke am Port wird das Unterprogramm mit dem Namen isr (Interrupt Service Routine) automatisch aufgerufen |
| name.fall(&isr) | Bei einer **fallenden** Flanke am Port wird das Unterprogramm mit dem Namen isr (Interrupt Service Routine) automatisch aufgerufen |
| name.enable\_irq() | Freigabe des Interrupts. Nach dieser Anweisung reagiert der Mikrocontroller auf die fallende oder steigende Flanke indem er automatisch die ISR aufruft |
| name.disable\_irq() | Sperre des Interrupts. Der Mikrocontroller reagiert nicht mehr auf Ereignisse am Port. |

Anhang 2: Nützliche Anweisungen

|  |  |
| --- | --- |
| Anweisung | Bedeutung |
| PortOut Lauflicht(PortC,0xFF) | Die Bits PC\_0 .. PC\_7 werden als 8-Bit-Ausgangsport festgelegt |
| Lauflicht=1; | Bit0 wird auf 1 gesetzt |
| Lauflicht=Lauflicht<<1; | Das Bitmuster in Lauflicht wird um eine Stelle nach links verschoben |
| If (Lauflicht==0) Lauflicht=1; | Von vorne |
|  |  |

Anhang 2:

Hinweise zur Programmierung:  
Da die Ausgabe auf das LCD-Display erfolgen soll benötigen Sie folgende Library:

Ergänzen Sie in der main.cpp an den entsprechenden Stellen:



Doppelklick

Library LCD\_i2c\_GSOE suchen

Import



Library mit #include „LCD.h“  
einbinden

LCD-Display-Objekt mit Namen z.B. mylcd  
deklarieren und erzeugen



Befehle:

* mylcd.clear(); //löscht das Display
* mylcd.cursorpos(*wert*); //plaziert den Cursor  
  0..15: 1.Zeile, 64..79 2.Zeile (64=0x40)
* mylcd.printf(*Formatstrinng,Werte*); //Ausgabe. Doku: Internet printf