Mikroprozessor Praktikum
Fr. HWP 2
Aufgabenblock 1

Paul Podlech 3910583

Max Wisniewski 4370074

8. November 2011

Aufgabe 1: I/O Pots

Output

A1.1.1

Klären Sie die Funktion folgender Register:

P4Sel: Mit diesem Register kann man das Signal von oder an ein internes Modul weiterleiten. Das sorgt dafür, dass man entweder Daten direkt verarbeiten kann ohne das die CPU mitarbeiten muss, oder Signale automatisch rausgeschickt werden können, wie eine Lampe die im Takt blinken soll. Dabei steht **0** für I/O Funktionalität und **1** für Modul Funktionalität.

P4Dir: Legt fest, ob man von diesem Port liest oder auf ihn raus schreiben möchte. Dabei steht:
0 für *IN*, das heißt man will lesen und
1 für *OUT*, das heißt man will schreiben.

P4Out: Hier schreibt man die Daten hinein, wenn man etwas senden möchte. Sobald Direction auf *OUT* steht, ist dieses Signal aussehn sichtbar.

P4In: Von hier kann gelesen werden, welches Signal am vierten Port anliegt, wenn die Direction am *P4Dir* auf *IN* steht. Allerdings reflektiert es auch im *OUT* Fall den Wert, den wir selber setzen.

Dies ist für für alle Ports gleich, wobei man für die anderen Ports die 4 durch die entsprechende Portnummer ersetzen muss.

A 1.1.2

Erstellen Sie eine Liste von Bitoperationen auf und geben Sie Operationen zum Setzen, Toggeln, Rücksetzen an.

Name	Beschreibung	Symbol
AND	Verunded die die Zahlen Bitweise	&
OR	Verodert die Zahlen Bitweise	|
XOR	VerXORt die zaheln Bitweise	HOCH
Komplement	Bildet das Bitweise Komplemet	~
Rechtshift	Shiftet die Bits nach rechts und füllt mit 0en	»
Linksshift	Shiftet die Bits nach links und füllt mit 0en	«

Setzen: $P4OUT \mid = 1 << k$. Setzt das k-te Bit auf 1. Um mehrere Bits zu setzen, können wir mit der Bitmaske verodern, die an allen Stellen eine 1 hat, die wir setzen wollen und eine 0 sonst.

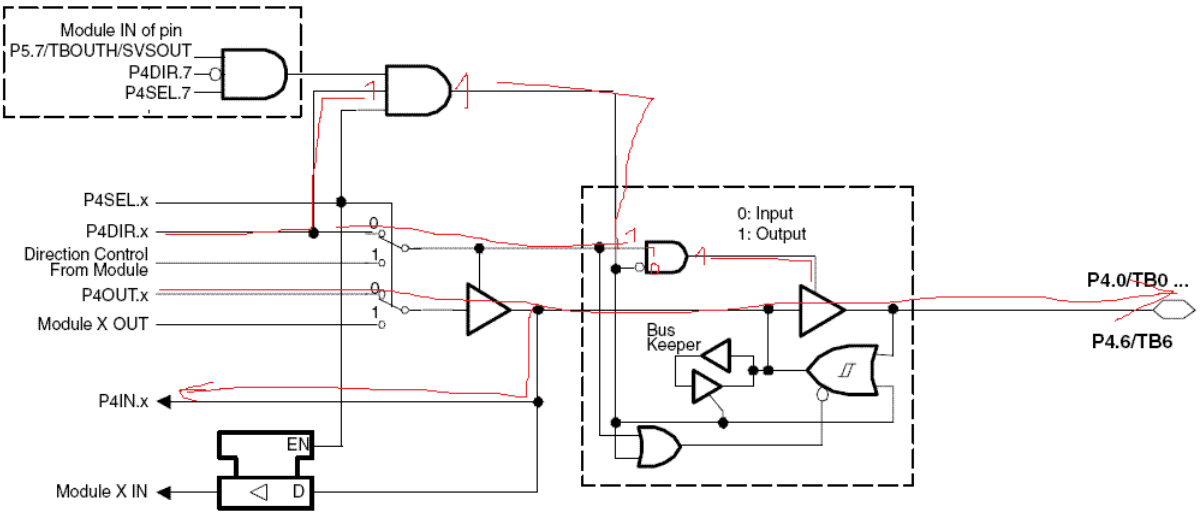
Toggeln: $P4OUT\ XOR = 1 << k$ Setzt das k-te Bit auf sein Komplement. Um mehrere Bits zu setzen, können wir mit der Bitmaske verodern, die an allen Stellen, die wir Toggeln wollen eine 1 und sonst eine 0.

Zurücksetzen: $P4OUT\ \& = \tilde{}(1 << k)$. Die Bitmaske hat an der k-ten Stelle eine 0 und sonst 1en. Damit Wird die k-te Stelle auf 0 gesetzt. Um mehrere Bits zu setzen, können wir mit der Bitmaske verunden, die an jeder Stelle eine 0 hat, die wir zurück setzen wollen und sonst 1en.

A1.1.3 Erläutern Sie anhand der Abbildung der internen Struktur einer Portleitung für die folgende Registerbelegung den Signalpfad und den Logikpegel der Portleitung P4.0

- define LED_ROT(0x01)
- P4SEL = 0x00
- P4DIR = 0x0F
- P4OUT |= LED_ROT

Daraus könen wir herraus lesen, dass Select für alle auf 0 steht, dass heißt es ist als I/O Port geschaltet. Die Direction von 0x0F heißt, dass für Bit 0 eine 1 anliegt (da die unteren 4 Bit gesetzt sind). Wenn LED_ROT mit unserem OUT verodert wird, kann man, wie in der letzen Aufgabe gezeigt, heißt dass, dass das 0te Bit gesetzt wird.



Aus unserem Bild kann man herrauslesen, das an P4.0 eine 1 herraus kommt.

A 1.1..4 Die LED leuchted, wenn wir eine Potentialdifferenz über der LED haben. Da auf der rechten Seite der LED eine Betriebsspannung (von 3V) anliegt, wie man aus dem Schaltbild lesen kann, benötigen wir ein niedrigeres Potential, damit ein Strom fließen kann. Diese wird bei 0 erreicht, da die Spannung zwischen den beiden Potentialen dann groß genug ist. Bei einem Highpegel wird diese Differenz 0 sein.

A 1.1.5 Erläutern Sie inhaltlich die Bedeutung der folgenden Codezeilen:

```
1  unsigned char a;
2  #define LEDRT (0x01)
3  P4DIR = 0x00;
4  a = 10;
5  P4OUT = a;
6  P4OUT = 0x01;
7  P4DIR = 0x07;
8  P4OUT = 0x00;
9  P4OUT |= 0x01;
10 P4OUT |= LEDRT;
11 P4OUT ^= ~LEDRT;
12 P4OUT ^= LEDRT;
```

Direction wird 2 mal gesetzt. Nach der ersten Anweisung wird alles auf eingang gesetzt und außen ist nichts zusehen. Nach der zweiten können wir an Leitung P4.0, P4.1, P4.2 ein Signal sehen.
Die Zeilen 5,6 haben keine Auswirkungen, da sie nicht sichtbar sind nach außen.
Nach Zeile 7 können wir die 1 auf P4.0 sehen, die in 6 gesetzt wurde.
Nach Zeile 8 können wir von außen an P4.0, P4.1, P4.2 eine 0 sehen.
Nach Zeile9 wurde auf P4.0 eine 1 gesetzt.
Nach Zeile 10 hat sich nichts geändert, da das Bit 0 schon vorher gesetzt wurde.
Nach Zeile 11 wurde mit dem komplement von LEDRT getoggelt. Das heißt, dass P4OUT nun 0xFF ist. Jedes Bit ist 1 und damit ist an P4.0, P4.1, P4.2 diese auch zu sehen. Die anderen Bits sind immer noch nicht auf OUT geschaltet.
Nach Zeile 12 wurde Bit 0 wieder getoggelt. Das bedeutet das an P4.0 eine 0 anliegt und an P4.1, P4.2 weiter die 1 bleibt.

A 1.1.6 Schreiben Sie ein Programm, dass die Phasen ein Ampel durchläuft. Und zwischen den einzelnen Phasen immer etwas Zeit lässt.
Wir haben uns dafür Makros für die Werte von Rot, Gelb und Grün geschrieben. Diese werden immer Program immer wieder kombiniert und dan auf die Outleitung gesetzt. Beim Aufruf haben wir als erstes darauf geachtet, dass die Leitungs Direction erst auf OUT gesetzt wird, wenn wir im OUT was etwas reingeschrieben haben.

```
1  #define ROT   (0x01) // Bitmaske fuer rote LED
2  #define GELB    (0x02) // Bitmaske fuer gelbe
       LED
3  #define GRUEN (0x04) // Bitmaske fuer gruene LED
4
5  void aufgabe1(){
6    P4SEL   = 0x00;  // setze den gesammten Port
         auf OUT
7    P4DIR   = 7  // setzt LED Leitungen auf OUT
8    P4OUT = ~ROT; //setzt die rote LED
9    wait(50000);
10   wait(50000);
11   P4OUT = ~( ROT | GELB); //setzt sowohl die
         rote als auch die gelbe LED
12   wait(50000);
13   P4OUT =   ~(GRUEN);  //setzt die gruene LED
14   wait(50000);
15   wait(50000);
16   P4OUT = ~(GELB)  // gelbe LED auf an
17   wait(50000);
18 }
```

Wir haben uns an dieser Stelle für die einfache Lösung entschieden und überschreiben alle Bits, die wir nicht setzten wollen. Dies könnte man umgehen indem man die Bits an sich setzt. Da in unserem Programm aber nicht mehr passiert, ist das an dieser Stelle vertretbar.

Unsere Beobachtungen waren: Es funktioniert.