HARD- UND SOFTWARETECHNIK 2

UNTERRICHTSSKRIPT - VERISON: 140830-00

Autor
Roman Gassmann

BZU Berufsschulzentrum Uster
Abteilung Elektronik

CH-8640 Uster, 30. August 2014

1 Input-/Output-Ports

1.1 Was ist ein Port

Jeder Mikrocontroller besitzt eine Vielzahl von Möglichkeiten um mit der Aussenwelt zu agieren. Die einfachste Möglichkeit bieten die IO-Ports (Input-/Output-Ports), welche jeweils 8 Pins des uCs steuern. D.h. jeder Pin (ausgenommen Speise- und Spezial-Pins) gehören zu einem Port.
Über sogenannte Register eines jeden Ports ist es nun möglich jeden einzelnen Pin des uCs zu steuern. Dabei sind diese Register nichts anderes als 2^nBit (meistens 8Bit) Speicher, welche an ganz bestimmten (bekannten) Adressen im Speicher des Mikrocontrollers liegen.
Im uC's werden nun über die verschieden internen Busse (Adress-, Steuer- und Datenbus) Einfluss auf die Register und damit auf die Pins genommen (siehe Abb. 1.1).

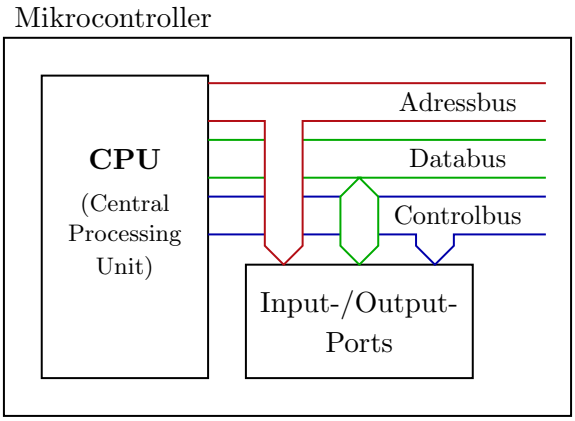


Abbildung 1.1: Mikrocontroller-Aufbau

1.2 Ouput-Data-Register (ODR)

Das Output-Register P1 (beim uVisionIDMCB32) Stellt nichts anderes als ein 8Bit Register dar welches definiert welcher PIN am LED-PORT eine 1 (EIN) oder eine 0 (AUS) hat. Dabei widerspiegelt jeweils ein Bit ein Pin (eine LED). Damit bestehen für die Einzelnen PINs die Wertigkeiten gemäss Tabelle 1.1.

BIT/PIN	7	6	5	4	3	2	1	0
Wertigkeit [DEZ]								
Wertigkeit [HEX]								

Tabelle 1.1: Wertigkeiten der PINs

1.2.1 Ansteuern

Schreiben Sie das Programm aus Listing 1.1. Variieren sie anschliessend auf Zeile 16 mit der zugewiesenen Zahl. Probieren Sie auch Zahlen wie 0x03 oder 0x17.

```
1  /*
2  Titel:      BspProgramm Ausgangs-Port
3  Datei:      P0toP1.c
4  Ersteller:  R.Gassmann
5  Funktion:   Liest die Schalter an Port 0 ein und gibt diese an Port 1 aus
6  */
7
8  // Einbindung der Bibliotheken
```

```
9  #include <stm32f10x_cl.h>    //Mikrokontrollertyp
10 #include "P0P1Touch.h"       //Header P0P1-defintionen
11
12 //Hauptprogramm
13 int main(void) {
14     InitP0P1Touch(1001);     //Touchscreen initialisieren
15     while (1) {      //Endlosschleife
16         P1 = 0x01;   //Einschalten der ersten LED
17     }
18 }
```

Listing 1.1: Beispielprogramm: Ausgangs-Port

1.2.2 Aufgaben

Versuchen Sie zu verstehen was geschieht wenn Sie bei Zeile 16 anstelle von `P1 = 0x01;` folgende Ausdrücke eingeben:

1. `P1 = 0x02 | 0x04 | 0x20;`
   Erkenntnis:

2. `P1 = 0x03 & 0x05 & 0x71;`
   Erkenntnis:

3. `P1 = 0x03 ^ 0x05;`
   Erkenntnis:

4. `P1 = 0x03 << 2;`
   Erkenntnis:

5. `P1 = 0xA0 >> 3;`
   Erkenntnis:

1.2.3 Einzelne PINs

Durch die Verwendung der in der P0P1Touch-Library enthaltenen Definitionen ist es auch möglich einzelne LED anzusteuern. Hierfür werden anschliessend an P1 mit _# der Entsprechende PIN mit der NR # angesprochen. Bsp:

```
1      P1_2 = 1;
2      P1_3 = 0;
```

Hier wird die 3 LED ein- und die 4 aus-geschaltet.

1.2.4 Aufgaben

Der Mikrocontroller des MCB32 Boards läuft standardmässig mit 72 MHz. Mit dieser hohen Taktrate benötigt man 0.17 μs bis 0.5 μs um ein Port von 0 auf 1 zu schalten. Um in den nächsten Übungen dennoch schlaue Anzeigen zu erhalten, müssen systematisch Verzögerungsschleifen eingesetzt werden. Dabei ist die wohl einfachste Methode um eine Verzögerung zu erhalten, den Mikrocontroller mit zählen zu beschäftigen (gemäss Listing 1.2). Diese Methode zur Verzögerung ist zwar sehr einfach um zusetzten jedoch alles andere als Ressourcensparend! Und sollte daher ausschliesslich für Übungszwecke eingesetzt werden.

```
1  long t;                    //  Immer long und auf 0 pruefend
2  for ( t =        12; t > 0 ; t-- );  //  1.0 s
3  for ( t =     12000; t > 0 ; t-- );  //  1.0ms
4  for ( t =    120000; t > 0 ; t-- );  //  10.0ms
5  for ( t =   1200000; t > 0 ; t-- );  //  100ms
6  for ( t = 12000000; t > 0 ; t-- );  //  1.00s
```

Listing 1.2: Quick-n-Dirty Verzögerungsschleifen

In einem nächsten Schritt sollen nun mit dem Einsatz von Verzögerungsschleifen folgende Programme erstellt werden:

Binärzähler
: Es soll ein Zähler erstellt werden welcher im Binärcode zählt. Nach dem erreichen des wertes 0xFF = 255, soll wieder bei Null angefangen werden.

Lauflicht
: Es soll ein Lauflicht erstellt werden. Dabei wird in jedem Schritt die nächste Led eingeschaltet. Nach der letzten Led soll wieder die Erste leuchten.

Knight Rider
: Es soll das nur zu gut bekannte Knight-Rider Licht erstellt werden.

1.3 Input-Data-Register (IDR)

Wie das Output-Data-Register ist das Input-data-Register nichts anderes als eine Speicherstelle im Mikrocontroller. Im Gegensatz zum ODR welches einzelne Pins ein und aus schalten kann, detektiert das IDR ob ein Pin auf 5 V (Logisch 1) oder 0 V (Logisch 0) gezogen wird. Die Wertigkeit der einzelnen Pins bleibt dabei gemäss Tabelle 1.1 bestehen.

1.3.1 Ansteuern

Schreiben Sie das Programm aus Listing 1.3. Diese Programm ermöglicht es über die Tasten die Entsprechende LED ein bzw. aus zu schalten.

```
1  /*
2  Titel:      Port Durchschaltung
3  Datei:      P0toP1.c
4  Ersteller:  R.Gassmann
5  Funktion:   Liest die Schalter an Port 0 ein und gibt diese an Port 1 aus
```

```
6  */
7
8  // Einbindung der Bibliotheken
9  #include <stm32f10x_cl.h>   //Mikrokontrollertyp
10 #include "P0P1Touch.h"      //Header P0P1-defintionen
11
12 //Hauptprogramm
13 int main(void) {
14     InitP0P1Touch(1001);    //Touchscreen initialisieren
15     while (1) {     //Endlosschleife
16         P1 = P0;    //Einschalten der ersten LED
17     }
18 }
```

Listing 1.3: Beispielprogramm: Port Durchschaltung

1.3.2 Aufgabe

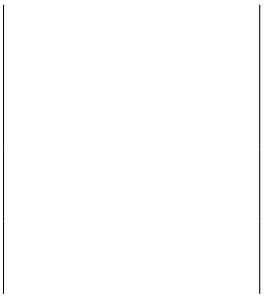
Wie können die Schalter "invertiert" an den Ausgang gegeben werden. Bsp. Wenn Schalter 0 ein ist soll die Led 0 aus sein. Versuchen Sie arithmetisch, logisch und mathematisch dieses Problem zu Lösen.

Analyse des Invertierens

Arithmetisch

Mathematisch

Logisch



1.4 Projekt-Auftrag

1.4.1 Veränderbares Night-Rider-Licht

Erstellen Sie ein Night-Rider-Licht gemäss Aufgabe 1.2.4. Modifizieren Sie dieses Programm so, dass Sie über 4-Tasten die Geschwindigkeit verändern können.

1.4.2 Soft Blink

Erstellen Sie ein Blinklicht (alle LED zusammen), welches nicht hart ein und aus schaltet, sondern weiche Übergänge enthält. Es ist klar, dass Sie dabei nicht die Spannung oder gar den Strom am uC-Ausgang variieren können. Um also eine LED "halb" einzuschalten, müssen Sie ein sogenanntes PWM Signal generieren. Erkundigen Sie sich im Internet was PWM bedeutet. Implementieren Sie ein PWM-Signal mittels for-Loops für die Ansteuerung ihrer LEDs. Um hier keine zeitlichen Verluste für die Aktualisierung des TFT's zu erhalten ersetzten Sie die Zeile

```
1     InitP0P1Touch(1001);     //Touchscreen initialisieren
```

mit folgender Zeile:

```
1     InitP0P1(); //Port0 und 1 initialisieren
```

es wird dann nicht mehr das Display initialisiert sondern bloss die Ein- und Ausgänge.

1.4.3 Softlauflicht*

Erstellen Sie das originale Night-Rider-Licht. Es handelt sich dabei nicht um eine einzelne LED die leuchtet, sondern um eine Gruppe (gemäss Abb. 1.2). Initialisieren Sie auch hier nur die Ports und nicht das LCD.

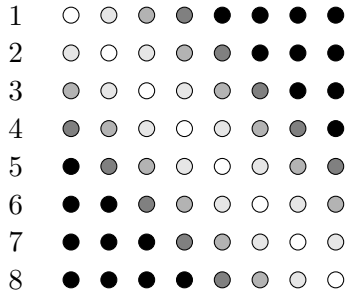


Abbildung 1.2: Originales Night-Rider-Licht

1.4.4 Multi-Night-Rider*

Es soll das an den Schalter eingestellte Muster im normalen Night-Rider Mode eingesetzt werden. Die Änderung des Musters soll dabei jeder Zeit möglich sein. Ist kein Schalter aktiv, so bleibt das ganze stehen. Werden alle Schalter eingeschaltet, so leuchten alle LED.