Informationen zur Klausur Mikroprozessortechnik

1 PI/T

1.1 Parallele Ein- und Ausgabe

1.1.1 Grundsätzliche Schritte

```
PIT1(PGCR)  =   0x00
```

1.1.2 Submodi und H2 bzw. H4 für Port A bzw. B einstellen

```
PIT1(PACR)  =   0xXX
PIT1(PBCR)  =   0xXX
```

XX steht hier für eine binär codierte 8 Bit Zahl.
Alle nicht erwähnten Bits müssen 0 bleiben.

BIT N	Funktion
7	immer 1
5	1 um H2 bzw. H4 als Ausgang zu nutzen. Sonst Eingang
3	1 für H2 bzw. H4 LOW, sonst HIGH

1.1.3 Richtung der Signale

```
PIT1(PADDR) =   0xXX
PIT1(PBDDR) =   0xXX
```

XX steht hier für eine binär codierte 8 Bit Zahl.
Das N-te Bit bezieht sich auf PA<N> bzw. PB<N>.

Wert des N-ten Bits	Richtung
0	Eingang
1	Ausgang

1.1.4 Abfrage von Eingangssignalen

In `PADR` bzw. `PBDR` ist das N-te Bit gesetzt, wenn der entsprechende Eingang des Portes A bzw. B auf HIGH liegt. **Der entsprechende Pin muß auch die RICHTUNG Eingang haben (s. o.).**

Abfrage, ob an H1, H2, H3 bzw. H4 ein HIGH-Signal anliegt, mittels eines IF-Konstruktes:

`if (PIT1(PSR) & 0x10)` Abfrage H1
`if (PIT1(PSR) & 0x20)` Abfrage H2
`if (PIT1(PSR) & 0x40)` Abfrage H3
`if (PIT1(PSR) & 0x80)` Abfrage H4
H2 und H4 müssen zuvor als Eingänge programmiert worden sein! (siehe 1.1.2)

1.1.5 Setzen von Ausgangssignalen

```
PIT1(PADR) =   0xXX
PIT1(PBDR) =   0xXX
```

XX steht hier für eine binär codierte 8 Bit Zahl.
Wird das N-te Bit gesetzt, ist der Logikpegel an PA<N> bzw. PB<N> HIGH, sonst LOW. **Der entsprechende Pin muß auch die RICHTUNG Ausgang haben (s. o.).**
Um *nur* das N-te Bit zu setzen bzw. zu löschen kann folgender C-Code verwendet werden:

```
PIT1(PADR) |=   0xXX;     /*setzen*/
PIT1(PADR) &=   ~(0xXX);  /*löschen*/
```

In XX müssen die Bits gesetzt (sprich 1) sein, die gesetzt oder gelöscht werden sollen. Gleiches gilt natürlich auch für jedes andere Register, solange es gelesen *und* beschrieben werden kann.
Zum Setzen bzw. Löschen der H2 bzw. H4 - Leitung siehe 1.1.2.

1.2 Timer

1.2.1 Grundsätzliche Schritte

```
PIT1(TCR)  =   0xXX
```

XX steht hier für eine binär codierte 8 Bit Zahl.
Alle nicht erwähnten Bits müssen 0 bleiben.

BIT N	Funktion
0	1 für Timerfreigabe und 0 für Sperren des Timers
[2,1]	[0,0] Normale Funktion: Clock wird durch den Vorteiler geschickt. [0,1] Es wird nur gezählt, wenn TIN/PC2 = HIGH. [1,0] PC2 wird zu TIN und ersetzt die Clock. [1,1] wie [1,0] allerdings ohne Vorteiler
4	Wenn 0 lädt der Zähler nach Erreichen von 0 wieder den Wert aus dem `CPR`. Sonst läuft er durch (000001, 000000, FFFFFF, FFFFFE).
6	1 →PC3 wird zu TOUT, an dem bei jedem Nulldurchgang des Timers der Pegel invertiert wird.

Der Vorteiler teilt das Taktsignal des Timers 1:32!

1.3 Counter Preload Register beschreiben

Das Counter Preload Register ist ein 24 Bit Register, bei dem jeweils die oberen (`CPRH`), mittleren (`CPRM`) und unteren (`CPRL`) 8 Bits gesondert geschrieben werden müssen. Der folgende Ausschnitt aus einem C-Programm verdeutlicht dies, indem der Wert der Variable `W` in das `CPR` übertragen wird. **Der Wert 0 ist nicht erlaubt!**

```
long W=12345;
PIT1(CPRH)  = W>>16;
PIT1(CPRM)  = W>>8;
PIT1(CPRL)  = W;
```

Das Laden des Zählers dauert immer einen Zählertakt! Das gilt auch, wenn der Zähler bei 0 angekommen ist, und sich neu lädt!

1.3.1 Abfragen des aktuellen Zählerstandes

Der 24 Bitwert des Zählers ist auch hier wieder in 3 8-Bitwerte aufgeteilt. Folgendes C-Beispiel demonstriert dies:

```
long W=( ((long) PIT1(CNTRH)) << 16 ) +
       ( ((long) PIT1(CNTRM)) << 8 ) + PIT1(CNTRL);
```

Das Auslesen sollte nur bei gesperrtem Timer erfolgen (s. o.).

1.4 Nulldurchgang abfragen

Wenn der Zähler den Wert 0 erreicht, wird das ZDS-Bit gesetzt (0-te Bit im TSR). Zum Löschen dieses Bits muß eine **1** auf das 0-te Bit im TSR geschrieben werden!

1.5 Zeitberechnung

Aufgabe:
Es soll eine Zeit von t dauern, vom Starten des Timers, bis zum Setzen des ZDS-Bits! Welcher Wert muß in das Counter Preload Register geladen werden?
Es wird angenommen, dass der Timer mit einen Takt von $\frac{8\text{Mhz}}{32}$ (Vorteiler 1:32) versorgt wird.

Lösung:
Zeit pro Takt: $\frac{32}{8 \cdot 10^6}\text{s} = 4\mu\text{s}$
Erforderlicher Wert für das Counter Preload Register, um eine Zeit von t zu erreichen:

$$\frac{t}{4\mu\text{s}} - 1$$

Wenn der Timer sich den Wert aus dem `CPR` holt, braucht er dafür einen Takt. Daher die -1!