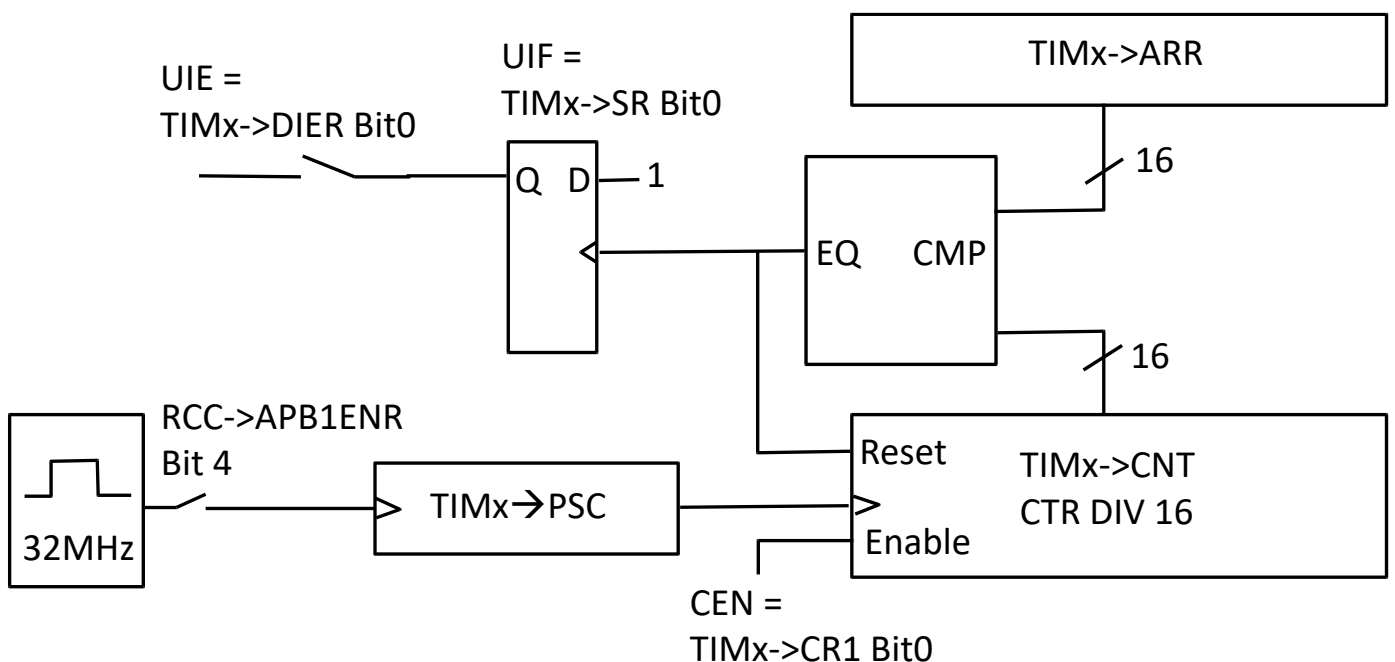


Liste der Timerbefehle

Befehl	C
Timer mit Takt versorgen	RCC->APB1ENR = 0b10000; //TIM6 RCC->APB1ENR = 0b100000; //TIM7
Timerstarten	TIMx->CR1=1; //setzt CEN auf 1
Timer stoppen	TIMx->CR1=0; //setzt CEN auf 0
Autoreloadregister mit <i>Wert</i> laden (soweit zählt der Timer bevor er wieder mit 0 beginnt)	TIMx->ARR= <i>Wert</i> ;
Prescaler einstellen Wert=31 bedeutet Zählperiode 1µs Wert 31999 bedeutet Zählperiode 1ms	TIMx->PSC= <i>Wert</i> ;
Zähler auf 0 setzen (auch andere Werte sind möglich)	TIMx->CNT=0;
Update Interrupt Flag (UIFx) zurücksetzen	TIMx->SR=0;
Timerinterrupt freigeben	TIMx->DIER=1;
Update Interrupt Flag (UIFx) abfragen	if (TIMx->SR==1) { }

Hinweise:

- Bei Timer TIM6 x=6, bei Timer TIM7 x=7



Aufgaben:

1. PC_0 soll mit einer Frequenz $f=1\text{Hz}$ blinken ($0,5\text{s} = 500\text{ms}$ Aus und ebenso lange an).

Vorgehensweise:

Im Unterprogramm initTimer:

- Timer mit Takt versorgen
- Timer mit Autoreloadwert für 500ms versorgen
- Prescaler für 1ms Zählperiode einstellen
- Timerzähler CNT mit 0 initialisieren
- Timer starten

Im Hauptprogramm:

initTimeraufrufen

Port PC_0 als Digitalausgang initialisieren

Endlosschleife:

- warten auf $\text{UIF6}=1$
 - UIF6 wieder zurücksetzen
 - PC_0 komplementieren

2. Digitaluhr

Erweitern Sie 1. zu einer Digitaluhr (LCD siehe Anhang)

3. Erweitern Sie die Digitaluhr so, dass mit den Tasten PA_1, PA_6 und PA_10 die Uhr gestellt werden kann. (PA_1: Stunden, PA_6: Minuten, PA_10: Sekunden jeweils mit externem Interrupt)

4. Stoppuhr

- Zeitauflösung: 10ms
- Globale Variable $\text{int } z=0$; zählt die Zeit mit einer Auflösung von 10 ms
- PA_1: Start mit externem Interrupt
- PA_6: Stop mit externem Interrupt
- PA_10: Reset mit externem Interrupt
- Anzeige:
 - CNT auf PortC
 - LCD Format: **s:ms** nur bei gestopptem Timer

Anhang:

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:

The screenshot shows the Mbed IDE interface. At the top, the 'Import' button is highlighted with a red box and a red arrow pointing to it. Below this, the 'Import Wizard' dialog is open, showing the 'Libraries' tab. The search bar contains 'LCD_i2c_GSOE'. A table lists published libraries, with 'LCD_i2c_GSOE' by Jack Hansdampf highlighted. A red box labeled 'Doppelklick' (double-click) points to this entry. Below the table, the 'main.cpp' file is open, showing the code. A red box labeled 'Library mit #include „LCD.h“ einbinden' (Library with #include „LCD.h“ include) points to the line `#include "LCD.h"`. Another red box labeled 'LCD-Display-Objekt mit Namen z.B. mylcd deklarieren und erzeugen' (LCD-Display object with name e.g. mylcd declare and create) points to the line `lcd mylcd;`. A final red box labeled 'Befehle:' (Commands:) contains a list of commands to be used in the code.

Import

Library LCD_i2c_GSOE suchen

Doppelklick

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(Formatstring, Werte);` //Ausgabe. Doku: Internet printf