MIKROPROZESSORPRAKTIKUM

SS2023

Termin 3

C-Programmierung für eingebettete Systeme Pointer, Peripherie, PIO, Interrupt, Timer (WAVE-Mode)

Name, Vorname	Matrikelnummer	Anmerkungen
Datum	Raster (z.B. Mi3x)	Testat/Datum

Legende: V:Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

Lernziele:

Mit den folgenden Versuchen sollen Sie lernen, wie Sie aus der Sprache "C" Peripherie (z. B. Timer) von modernen Mikrocontrollern nutzen.

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner „sftp://stxxyyyy@userv-shell.fbi.h-da.de/share/LabDisk/MI/" den Ordner mpsDUESS2023. Dort finden Sie zu jedem Termin vorgegebene Dateien.

Aufgabe 1:

Das Blinken der LED „L" vom Arduino DUE-Board aus Termin2 soll mit einem symmetrischen Rechtecksignal mit einer Frequenz von 0,5Hz über einen Timer realisiert werden.

ACHTUNG: Achten Sie darauf, dass zu keinem Zeitpunkt ein Dauerhighsignal am zugehörigen Port ausgegeben wird.

Vervollständigen Sie die das gegebene Programm *Termin3Aufgabe1.c* entsprechend.

Könnte der eingesetzte Timer bei dem benötigten symmetrischen Signal auch anders betrieben werden?

Zeigen Sie die Berechnung der benötigten Werte um die Frequenz und das geforderte Pulsweitenverhältnis richtig einzustellen.

..

Aufgabe 2:

Erweitern Sie Ihr Programm so, dass die blinkende LED „L" durch Betätigung der Taste (Siehe Termin2) eingeschaltet und abgeschaltet werden kann.

Welche Möglichkeiten haben Sie gefunden, um die LED „L" ein- bzw. auszuschalten?

..

Für welche Lösung entscheiden Sie sich und warum?

..

Aufgabe 3:

Mit der zur Verfügung stehenden Taste soll die LED „L" bei jedem Betätigen um 10% dunkler werden. Beim Betätigen der Taste und ausgeschalteter LED soll die LED wieder mit 90% eingeschaltet werden.

Dokumentieren Sie, durch Sichten der entsprechenden Register/Speicherstellen, dass die Diode jederzeit mit richtigen Werten angesteuert wird.

Aufgabe 4:

Erstellen Sie zu diesem Termin ein Protokoll mit den Lösungen zu den Aufgaben und Ihren Erkenntnissen. Das Protokoll sollen Sie zum nächsten Termin vorlegen können. Isolieren Sie die Routinen/ Funktionen, welche für die nächsten Termine evtl. noch benötigt werden.

```
// Loesung zu Termin 3
// Aufgabe 1
// von:
// vom:
//
#include "../h/pmc.h"
#include "../h/tc.h"
#include "../h/pio.h"
#include "../h/aic.h"

void taste_irq_handler (void) __attribute__ ((interrupt));

// Interruptserviceroutine für die Tasten SW1 und SW2
void taste_irq_handler (void)
{
 StructPIO* piobaseB  = PIOB_BASE;              // Basisadresse PIO B
 StructAIC* aicbase   = AIC_BASE;               //__

// ab hier entsprechend der Aufgabestellung ergänzen
// ************************************************


        aicbase->AIC_EOICR = piobaseB->PIO_ISR;         //__
}

// Timer3 initialisieren
void Timer3_init( void )
{
        StructPMC* pmcbase = PMC_BASE;          // Basisadresse des PMC
        StructTC* timerbase3  = TCB3_BASE;      // Basisadressse TC Block 1
        StructPIO* piobaseA  = PIOA_BASE;       // Basisadresse PIO B

        pmcbase→PMC_PCER = 0x2200;              // Peripheral Clocks einschalten für PIOA und TC3

        timerbase3->TC_CCR = TC_CLKDIS;         // Disable Clock

 // Initialize the mode of the timer 3
        timerbase3->TC_CMR =
                TC_ACPC_CLEAR_OUTPUT  |         // ACPC         : Register C clear TIOA
                TC_ACPA_SET_OUTPUT    |         // ACPA         : Register A set TIOA
                TC_WAVE               |         // WAVE         : Waveform mode
                TC_CPCTRG             |         // CPCTRG       : Register C compare trigger enable
                TC_CLKS_MCK1024;                // TCCLKS       : MCKI / 1024

 // Initialize the counter:
        timerbase3->TC_RA = 400;    // hier sind noch die richtigen Werte zu ermitteln
        timerbase3->TC_RC = 800;    // Die Pumpe soll mit einem 50Hz Signal betrieben werden

 // Starten des Timer und Ausgabe eines Low-Pegel an die Pumpe
        timerbase3->TC_CCR = TC_CLKEN ;     //__
        timerbase3->TC_CCR = TC_SWTRG ;     //__
        piobaseA->PIO_PER  = (1<<PIOTIOA3) ;  //__
        piobaseA->PIO_OER  = (1<<PIOTIOA3) ;  //__
        piobaseA->PIO_CODR = (1<<PIOTIOA3) ;  //__
}

int main(void)
{
        StructPMC* pmcbase = PMC_BASE;      // Basisadresse des PMC
        StructPIO* piobaseA = PIOA_BASE;    // Basisadresse PIO A
        StructPIO* piobaseB = PIOB_BASE;    // Basisadresse PIO B


// ab hier entsprechend der Aufgabestellung ergänzen
//************************************************



        return 0;
}
```