Prüfung: µ-Computertechnik - Bachelor
Termin: Montag, 16.07.2012; 09:30-10:30
Prüfer: Prof. Walter
Hilfsmittel: beliebig, keine Kommunikationsmittel, kein Servo ;-)

Name: _____

Vorname: _____

Studiengang: _____

Labor: _____

Bitte überprüfen Sie, ob alle Protokolle des Labors in den Webseiten sind, inkl. Protokoll der Präsentation
bitte keine rote Farbe verwenden

(nicht ausfüllen)!

Aufgabe	mögl. Punkte	erreichte Punkte
1	10	
2	12	
3	18	
4	10	
Gesamt	50	
	Note	

Bearbeiten Sie die Aufgaben nur, falls Sie keine gesundheitlichen Beschwerden haben.

Viel Erfolg!

Bemerkungen: Bitte erstellen Sie die Lösungen auf der eigenen Festplatte im Ordner MCSS12_NAME! Am Ende der Klausur belassen Sie ihren Rechner am Platz und verlassen den Raum.
Zum Kopieren ihrer Lösung werden Sie jeweils mit Namen im Anschluss an die Klausur aufgerufen. Bitte senden Sie ihre Lösung zusätzlich an:
waju0001@web.de Betreff: MCSS12_Name
Schreiben Sie in jeden Programmkopf ihren Namen! Bei nicht vorhandenem Namen wird die Lösung NICHT gewertet.

WICHTIG!
Alle Programme sind für die VC_2-Hardware zu schreiben. Der Sysclk liegt bei 3 MHz.
Speichern Sie jeweils die dazugehörige **Configuration Wizard Datei!!**
Nur dokumentierte Software!

1. 8051-Programmierung (10 Punkte)

Der Mitarbeiter „Ewengzweng“ wird aus der Firma entlassen. Er hinterlässt folgendes Programm:

```
$NOMOD51                    ;der Modus fuer 8051 wird abgeschaltet
$debug
$nolist                    ;es wird kein Listing fuer reg51 erstellt
$include(REG51.inc)        ;die 8051-spezifischen Daten
$list                      ;es wird ein Listing erstellt
$title (INDIREKTE_ADRESSIERUNG.A51)
;------------------------------------------------------------------------
-
;Programmbeschreibung
;------------------------------------------------------------------------
;Programm:
; Indirekte Adressierung
; T1--> Speicherbereich von 30h bis 7FH mit A5h beschreiben
; T2--> Speicherbereich von 30h bis 7FH mit 5Ah beschreiben
;
;Erstellt am:
;Programmiert: Ewengzweng
;
;Verwendete Einspruenge: keine
;
;Verwendete Unterprogramme: keine
;
;
;Verwendete Register und Variable:
;Registerbank(0)
;R2
;
;Kommentar:
;
;Aenderungen: Prof. J. Walter
;Geaendert am: 16.07.2012
;
;
;------------------------------------------------------------------------
-
;Initialisierungsteil fuer allgemeine Konstanten
;------------------------------------------------------------------------
-

CSEG AT 0H              ; Legt absolute Codesegmentadresse auf 0h
jmp INIT

;------------------------------------------------------------------------
-
;Interrupt-Vektoren
```

```
;-------------------------------------------------------------------------
-
;ORG

;-------------------------------------------------------------------------
-
;Initialisierungsteil fuer On-Chip Peripherie
;-------------------------------------------------------------------------
-
ORG 100H                ; Programmstart bei 100H
INIT:

;-------------------------------------------------------------------------
-
;Programmschleife
;-------------------------------------------------------------------------
-
ABFRAGE:
jnb P1.1,SCHREIBE_A5        ; T1--> A5h von 30H bis 7F
jnb P1.2,SCHREIBE_5A        ; T2--> 5Ah von 30h bis 7F
jmp ABFRAGE                 ;zur ABFRAGE

SCHREIBE_A5:
mov A,#0A5h                 ; Schreibe A5 in Akku
mov R0,#30h                 ; Zeiger = Adresse
mov R2,#50h                 , Abbruchbedingung: nach 50h anhalten
SCHLEIFE_1:
mov @R0,A                   ; Schreibe den Inhalt von A in die Adresse
                            , die in R0 steht
inc R0                      ; inc R0
djnz R2,SCHLEIFE_1          ; Ueberpruefe ob bereits
                            ; 50h geschrieben wurden

SCHREIBE_5A:

mov A,#05Ah                 ; Schreibe A5 in Akku
mov R0,#30h                 ; Zeiger = Adresse
mov R2,#50h                 , Abbruchbedingung: nach 50h anhalten

SCHLEIFE_2:
mov @R0,A                   ; Schreibe den Inhalt von A in die
                            ; Adresse die in R0 steht
inc R0                      ; inc R0
djnz R2,SCHLEIFE_2          ; Ueberpruefe ob bereits
                            ; 50h geschrieben wurden
jmp ABFRAGE                 ; zur ABFRAGE
end
```

a) Was macht das Programm? **(2 Punkte)**
b) Ergänzen Sie die Dokumentation des Programmes im Code. **(3 Punkte)**
c) Vervollständigen Sie das Programm im Code. **(5 Punkte)**

Programme „Sekundentakt - CAM_1“ (12+18 Punkte)

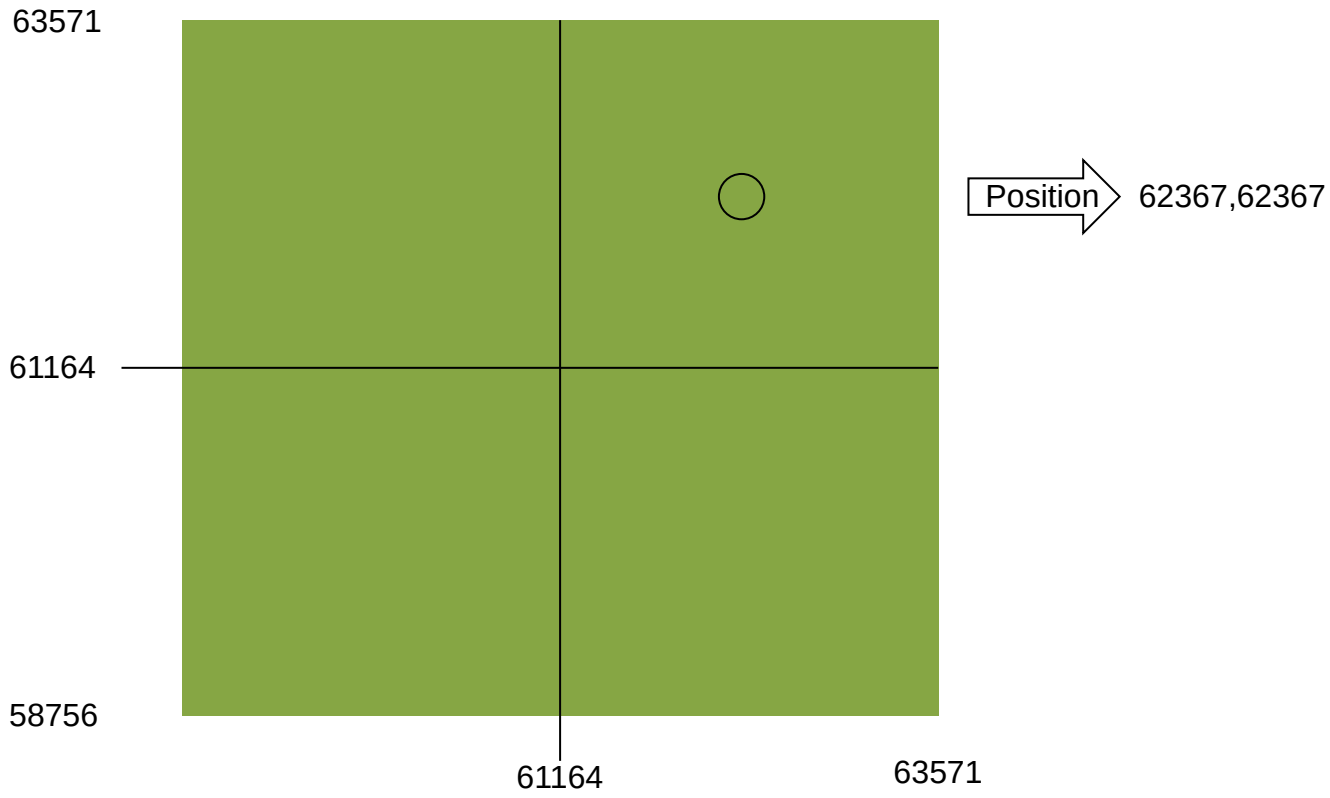

Bild: Beispiel Positionsausgabe

Eine sehr gute Programmiererin entwirft auf einem Touch-Pad das Programm: CAMAPP. Durch Aufsetzen eines Fingers auf das Touch-Pad werden die Koordinaten in die externen Speicherzellen des VC_2 ab Adresse 0 geschrieben. Ihr Freund studiert Mechatronik und steuert mit diesen Daten zwei Servos an, die eine Kamera steuern.

Für einen ersten Test erzeugt er mit Timer 0 einen **Sekundentakt**. Dann schreibt er ein zweites Programm: **CAM_1**. Nach jeder Sekunde werden die neuen Positionswerte vom externen Speicher übernommen und damit die Servomotoren gesteuert.

a) Tragen Sie die Werte: Adresse, Daten in die nachfolgende Tabelle für den Kreis (62367) in Hex ein. ***(4 Punkte)***
b) Erzeugen Sie das Programm **Sekundentakt**. ***(8 Punkte)***
c) Erweitern Sie das Programm Sekundentakt auf das Programm: **CAM_1** ***(18 Punkte)***

Adresse ext. Datenspeicher	Wert in Hex
0000	9F
0001	F3
0002	9F
0003	F3

2b Sekundentakt

Sehr einfach lässt dich dieser aus dem Programm HZ_250 erzeugen. Es werden mit dem Zählregister R2 250 Interrupts gezaehlt.

```
$NOMOD51                ;der Modus fuer 8051 wird abgeschaltet
$debug
$nolist                 ;es wird kein Listing fuer reg535 erstellt
$include(C8051F340.inc) ;die C8051F340-spezifischen Daten
$list                   ;es wird ein Listing erstellt
$title (Sekundentakt.A51)

;------------------------------------------------------------------------
;Programmbeschreibung
;------------------------------------------------------------------------
;Programm:
;;Mit T1 startet das Programm
;An Port 3.4 liegt LED 3
;LED 3 wird 250 mal pro Sekunde umgeschaltet. --> R2 zaehlt 250 -->
umschalten
;Sysclk ist mit 3 MHz festgelegt
;Timer 0 im Autoreload-Modus mit Sysclk / 48 --> 62500 Hz / 250 = 250 Hz --
;> 4ms
;Reload-Wert 256-250=6
;
;
;
;Erstellt am: Sonntag, 17. Juni 2012 12:35:57
;Programmiert: Prof. Juergen Walter
;
;Verwendete Einspruenge: keine
;
;Verwendete Unterprogramme: keine
;
;
;Verwendete Register und Variable:
;Registerbank(0)
;R2
;
;Kommentar:
;
;Aenderungen:
;Geaendert am: Sonntag, 17. Juli 2011 12:35:57
;
;
;------------------------------------------------------------------------
;Initialisierungsteil fuer allgemeine Konstanten
;------------------------------------------------------------------------


CSEG AT 0H              ;Legt absolute Codesegmentadresse auf 0h
call Init_Device        ;Aufruf zur Initialisierung der Controller
Funktionen
jmp INIT
;
;------------------------------------------------------------------------
;Interrupt-Vektoren
;------------------------------------------------------------------------
ORG 000BH
call ISR_T0       ;Interrupt Service Routine Timer 0
```

```
reti

;-------------------------------------------------------------------------
;Initialisierungsteil fuer On-Chip Peripherie
;-------------------------------------------------------------------------
ORG 100H                ;Programmstart bei 100H
INIT:
mov R2,#250                     ;250Hz --> 1 Hz
;-------------------------------------------------------------------------
;Programmschleife
;-------------------------------------------------------------------------

ABFRAGE:
jnb P1.1,START                  ;Start des Programmes
jmp ABFRAGE

START:
setb TR0                        ;Timer laeuft
jmp ABFRAGE

ISR_T0:
djnz R2,SEK                     ;250 mal?
mov R2,#250                     ;Register neu laden
cpl P3.2                        ;umschalten
SEK:
ret



;-------------------------------------------------------------------------
; Peripheral specific initialization functions,
; Called from the Init_Device label
;-------------------------------------------------------------------------
PCA_Init:
    anl  PCA0MD,    #0BFh
    mov  PCA0MD,    #000h
    ret

Timer_Init:
    mov  TMOD,      #002h
    mov  CKCON,     #002h
    mov  TH0,       #006h
    ret

Port_IO_Init:
    ; P0.0  -  Skipped,     Push-Pull,  Digital
    ; P0.1  -  Skipped,     Push-Pull,  Digital
    ; P0.2  -  Skipped,     Push-Pull,  Digital
    ; P0.3  -  Skipped,     Push-Pull,  Digital
    ; P0.4  -  TX0 (UART0), Push-Pull,  Digital
    ; P0.5  -  RX0 (UART0), Push-Pull,  Digital
    ; P0.6  -  Skipped,     Push-Pull,  Digital
    ; P0.7  -  Skipped,     Push-Pull,  Digital

    ; P1.0  -  CEX0  (PCA), Push-Pull,  Digital
    ; P1.1  -  CEX1  (PCA), Push-Pull,  Digital
    ; P1.2  -  CEX2  (PCA), Push-Pull,  Digital
    ; P1.3  -  CEX3  (PCA), Push-Pull,  Digital
    ; P1.4  -  CEX4  (PCA), Push-Pull,  Digital
    ; P1.5  -  Skipped,     Push-Pull,  Digital
```

```
    ; P1.6  -  Skipped,     Push-Pull,  Digital
    ; P1.7  -  Skipped,     Push-Pull,  Digital

    ; P2.0  -  Skipped,     Push-Pull,  Digital
    ; P2.1  -  Skipped,     Push-Pull,  Digital
    ; P2.2  -  Skipped,     Push-Pull,  Digital
    ; P2.3  -  Skipped,     Push-Pull,  Digital
    ; P2.4  -  Skipped,     Push-Pull,  Digital
    ; P2.5  -  Skipped,     Push-Pull,  Digital
    ; P2.6  -  Skipped,     Push-Pull,  Digital
    ; P2.7  -  Skipped,     Push-Pull,  Digital

    ; P3.0  -  Skipped,     Push-Pull,  Digital
    ; P3.1  -  Skipped,     Push-Pull,  Digital
    ; P3.2  -  Skipped,     Push-Pull,  Digital
    ; P3.3  -  Skipped,     Push-Pull,  Digital
    ; P3.4  -  Unassigned,  Push-Pull,  Digital
    ; P3.5  -  Unassigned,  Push-Pull,  Digital
    ; P3.6  -  Unassigned,  Push-Pull,  Digital
    ; P3.7  -  Unassigned,  Push-Pull,  Digital

    mov  P0MDOUT,   #0FFh
    mov  P1MDOUT,   #0FFh
    mov  P2MDOUT,   #0FFh
    mov  P3MDOUT,   #0FFh
    mov  P0SKIP,    #0CFh
    mov  P1SKIP,    #0E0h
    mov  P2SKIP,    #0FFh
    mov  P3SKIP,    #00Fh
    mov  XBR0,      #001h
    mov  XBR1,      #045h
    ret

Oscillator_Init:
    mov  OSCICN,    #081h
    ret

Interrupts_Init:
    mov  IT01CF,    #010h
    mov  IE,        #082h
    ret

; Initialization function for device,
; Call Init_Device from your main program
Init_Device:
    lcall PCA_Init
    lcall Timer_Init
    lcall Port_IO_Init
    lcall Oscillator_Init
    lcall Interrupts_Init
    ret

end
```

CAM_1

```
$NOMOD51                ;der Modus fuer 8051 wird abgeschaltet
$debug
$nolist                 ;es wird kein Listing fuer reg535 erstellt
$include(C8051F340.inc) ;die C8051F340-spezifischen Daten
$list                   ;es wird ein Listing erstellt
$title (CAM_1.A51)
;------------------------------------------------------------------------
;Programmbeschreibung
;------------------------------------------------------------------------
;Programm:
;PWM auf Port 1.0 und 1.1
;Jede Sekunde werden die 4 Werte aus dem externen Speicher in die Compare-
Register gelesen.
;An Port 3.4 liegt LED 3
;LED 3 wird 250 mal pro Sekunde umgeschaltet. --> R2 zaehlt 250 -->
umschalten
;Sysclk ist mit 3 MHz festgelegt
;Timer 0 im Autoreload-Modus mit Sysclk / 48 --> 62500 Hz / 250 = 250 Hz --
> 4ms
;Reload-Wert 256-250=6
;
;
;
;Erstellt am: Sonntag, 17. Juli 2011 12:35:57
;Programmiert: Prof. Juergen Walter
;
;Verwendete Einspruenge: keine
;
;Verwendete Unterprogramme: keine
;
;
;Verwendete Register und Variable:
;Registerbank(0)
;R2
;
;Kommentar:
;
;Aenderungen:
;Geaendert am: Sonntag, 17. Juli 2011 12:35:57
;
;
;------------------------------------------------------------------------
;Initialisierungsteil fuer allgemeine Konstanten
;------------------------------------------------------------------------


CSEG AT 0H              ;Legt absolute Codesegmentadresse auf 0h
call Init_Device        ;Aufruf zur Initialisierung der Controller
Funktionen
jmp INIT
;
;------------------------------------------------------------------------
;Interrupt-Vektoren
;------------------------------------------------------------------------
ORG 000BH
call ISR_T0       ;Interrupt Service Routine Timer 0
```

```
reti

;------------------------------------------------------------------------
;Initialisierungsteil fuer On-Chip Peripherie
;------------------------------------------------------------------------
ORG 100H                ;Programmstart bei 100H
INIT:
mov R2,#250                  ;250Hz --> 1 Hz
setb TR0                     ;Timer laeuft
;------------------------------------------------------------------------
;Programmschleife
;------------------------------------------------------------------------

ABFRAGE:
jmp ABFRAGE


ISR_T0:
djnz R2,SEK                  ;250 mal?

//1 Sekunde vorbei
mov R2,#250                  ;Register neu laden
cpl P3.2                     ;umschalten

//Programmierung der PWM
mov DPTR,#0000h              ;Datenpointer auf X:0x0000h
movx A,@DPTR                 ;X-Achse-Position Highbyte auslesen
mov PCA0CPL0,A               ;in X-Servo-PCA-Highbyte schreiben

inc DPTR                     ;Datenpointer auf nächste Stelle (X:0x0001h)
movx A,@DPTR                 ;X-Achse-Position Lowbyte auslesen
mov PCA0CPH0,A               ;in X-Servo-PCA-Lowbyte schreiben

inc DPTR                     ;Datenpointer auf nächste Stelle (X:0x0002h)
movx A,@DPTR                 ;Y-Achse-Position Highbyte auslesen
mov PCA0CPL1,A               ;in Y-Servo-PCA-Highbyte schreiben

inc DPTR                     ;Datenpointer auf nächste Stelle (X:0x0003h)
movx A,@DPTR                 ;Y-Achse-Position Lowbyte auslesen
mov PCA0CPH1,A               ;in Y-Servo-PCA-Lowbyte schreiben

SEK:
ret



;------------------------------------------------------------------------
; Peripheral specific initialization functions,
; Called from the Init_Device label
;------------------------------------------------------------------------
PCA_Init:
    mov  PCA0CN,    #040h
    anl  PCA0MD,    #0BFh
    mov  PCA0MD,    #008h
    mov  PCA0CPM0,  #0C2h
    mov  PCA0CPM1,  #0C2h
    ret

Timer_Init:
    mov  TMOD,      #002h
```

```
    mov  CKCON,     #002h
    ret

Port_IO_Init:
    ; P0.0  -  Skipped,     Push-Pull,  Digital
    ; P0.1  -  Skipped,     Push-Pull,  Digital
    ; P0.2  -  Skipped,     Push-Pull,  Digital
    ; P0.3  -  Skipped,     Push-Pull,  Digital
    ; P0.4  -  TX0 (UART0), Push-Pull,  Digital
    ; P0.5  -  RX0 (UART0), Push-Pull,  Digital
    ; P0.6  -  Skipped,     Push-Pull,  Digital
    ; P0.7  -  Skipped,     Push-Pull,  Digital

    ; P1.0  -  CEX0  (PCA), Push-Pull,  Digital
    ; P1.1  -  CEX1  (PCA), Push-Pull,  Digital
    ; P1.2  -  CEX2  (PCA), Push-Pull,  Digital
    ; P1.3  -  CEX3  (PCA), Push-Pull,  Digital
    ; P1.4  -  CEX4  (PCA), Push-Pull,  Digital
    ; P1.5  -  Skipped,     Push-Pull,  Digital
    ; P1.6  -  Skipped,     Push-Pull,  Digital
    ; P1.7  -  Skipped,     Push-Pull,  Digital

    ; P2.0  -  Skipped,     Push-Pull,  Digital
    ; P2.1  -  Skipped,     Push-Pull,  Digital
    ; P2.2  -  Skipped,     Push-Pull,  Digital
    ; P2.3  -  Skipped,     Push-Pull,  Digital
    ; P2.4  -  Skipped,     Push-Pull,  Digital
    ; P2.5  -  Skipped,     Push-Pull,  Digital
    ; P2.6  -  Skipped,     Push-Pull,  Digital
    ; P2.7  -  Skipped,     Push-Pull,  Digital

    ; P3.0  -  Skipped,     Push-Pull,  Digital
    ; P3.1  -  Skipped,     Push-Pull,  Digital
    ; P3.2  -  Skipped,     Push-Pull,  Digital
    ; P3.3  -  Skipped,     Push-Pull,  Digital
    ; P3.4  -  Unassigned,  Push-Pull,  Digital
    ; P3.5  -  Unassigned,  Push-Pull,  Digital
    ; P3.6  -  Unassigned,  Push-Pull,  Digital
    ; P3.7  -  Unassigned,  Push-Pull,  Digital

    mov  P0MDOUT,   #0FFh
    mov  P1MDOUT,   #0FFh
    mov  P2MDOUT,   #0FFh
    mov  P3MDOUT,   #0FFh
    mov  P0SKIP,    #0CFh
    mov  P1SKIP,    #0E0h
    mov  P2SKIP,    #0FFh
    mov  P3SKIP,    #00Fh
    mov  XBR0,      #001h
    mov  XBR1,      #045h
    ret

Oscillator_Init:
    mov  OSCICN,    #081h
    ret

Interrupts_Init:
    mov  IT01CF,    #010h
    mov  IE,        #082h
    ret
```

```
; Initialization function for device,
; Call Init_Device from your main program
Init_Device:
    lcall PCA_Init
    lcall Timer_Init
    lcall Port_IO_Init
    lcall Oscillator_Init
    lcall Interrupts_Init
    ret

end
```

Eagle Schaltplan „UART“ (10 Punkte)

Die Kommunikation der Servos vom Touch-Pad erfolgt über UART0.

a) Überprüfen Sie im Configuration-Wizard an welchen Pins der Touch-Pad angeschlossen wird.

b) Zeichnen Sie den Schaltplan mit Eagle unter Verwendung der Vorlage für Projekte. Verwenden Sie einen einfachen 6-poligen Pfostenstecker und schliessen Sie alle Signale am Mikrocontroller an.