LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 1

BEISPIEL 1 – IDE

Starten Sie die Keil µVision2-IDE und erstellen Sie anhand der zur Verfügung gestellten Anleitung ein Beispielprojekt. Führen Sie die darin beschriebenen Arbeitsschritte durch und verfolgen Sie die Pegel wechsel der Pins von Port 1 im Einzelschrittmodus des Debuggers.

BEISPIEL 2 – TASCHENRECHNER

Programmieren Sie einen Taschenrechner, der fortlaufend zwei 4-Bit-Werte logisch miteinander verknüpft und das Ergebnis der Rechenoperation ausgibt. Die Eingabewerte liegen dabei an Port 0 an, wobei das High Byte (die oberen 4 Bit) den ersten Operanden definiert, und das Low Byte (die unteren 4 Bit) den zweiten. Das Ergebnis wird auf Port 1 ausgegeben, die Wahl der Rechenoperation erfolgt über Port 2 (0 = addieren, 1 = subtrahieren, 2 = UND (AND), 3 = ODER (OR), 4 = Exklusives Oder (XOR)). Nachfolgend finden Sie ein Beispiel, in dem die beiden Zahlen 7 und 3 zu 10 addiert werden:

Port 0	
0111	0011
Operand: 7 (High Byte)	Operand: 3 (Low Byte)

Port 1
0000 1010
Ergebnis: 10

Port 2
0000 0000
Addition

BEISPIEL 3 – INTERRUPT-HANDLING

Modifizieren Sie Beispiel 2 derart, dass die Berechnung des Ergebnisses nicht laufend erfolgt, sondern lediglich dann, wenn eine fallende Flanke an Port 3, Pin 2 anliegt. Nutzen Sie zur Reaktion auf dieses Ereignis den externen Interrupt 0 und führen Sie die Ergebnisberechnung in der Interrupt-Service-Routine (ISR) durch. Setzen Sie zur Aktivierung des Interrupts die entsprechenden Bits in den speziellen Funktionsregistern (SFR) TCON und IE, deren Semantik Sie im 8051-Hardware-Manual in Kapitel 2.11 bzw. im AT89C51SND1C-Manual im Abschnitt „Interrupt System“ im finden.

BEISPIEL 4 – ASSEMBLER-CODE-ANALYSE

Analysieren Sie den vom Compiler generierten Assemblercode in der Disassembler-Ansicht des Debuggers und ermitteln Sie die Größe, die der gesamte Code einnimmt, sowie die Ausführungsdauer der einzelnen Assembleranweisungen bei einem gegebenen Standardtakt von 12 MHz. Nutzen Sie dazu die Informationen in den Kapiteln 1.3 bis 1.9 des Manuals.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 2

BEISPIEL 5 – HEX-ASCII-UMWANDLUNG

Schreiben Sie ein Programm, das den an Port 0 anliegenden Wert als Hexadezimalzahl in ASCII auf den beiden Ports 1 und 2 ausgibt. Port 1 stellt dabei das High Byte dar, während Port 2 das Low Byte darstellt. Es ist empfehlenswert, zuerst die beiden Hexadezimalziffern (4 Bit) zu extrahieren und sie anschließend mit einer Funktion in die entsprechenden ASCII-Zeichen umzuwandeln. Es ist dabei zu beachten, dass die ASCII-Zeichen von ‘0‘ bis ‘9‘ den Werten 0x30 bis 0x39 entsprechen, während die ASCII-Zeichen von ‘A‘ bis ‘F‘ den Werten 0x41 bis 0x46 entsprechen (vgl. ASCII-Tabelle). Nachfolgendes Beispiel stellt die gewünschte Ausgabe bei Anliegen des Werts 0x3C an Port 0 dar:

Port 0	
0x3	0xC
High Byte	Low Byte

Port 1
0x33
‘3‘ (ASCII)

Port 2
0x43
‘C‘ (ASCII)

BEISPIEL 6 – SERIELLE DATENAUSGABE (VEREINFACHT)

Erstellen Sie ein Programm, das auf Port 1, Pin 1 den an Port 0 anliegenden Hexadezimalwert laufend seriell ausgibt, wobei jedes Daten-Byte von einem Startbit (0, vor den Daten) und einem Stoppbit (ebenfalls 0, nach den Daten) umschlossen wird. Das Timing muss in diesem Beispiel nicht beachtet werden – es genügt, die Daten bitweise (das niederwertigste Bit zuerst) am angegebenen Pin auszugeben und die Ausgabe im Einzelschrittmodus im Debugger nachzuvollziehen. Nachfolgende Abbildung illustriert den gewünschten Output an Port 1, Pin 1, wenn an Port 0 der Wert 0x43 anliegt:

Startbit	**Daten**	**Stoppbit**
0	1100 0010	0

Zeit

BEISPIEL 7 – ASCII-HEX-UMWANDLUNG

Analog zu Beispiel 5 soll die Umwandlung zweier ASCII-Zeichen, die Hexadezimalziffern darstellen, in den entsprechenden Hexadezimalwert programmiert werden. Die beiden ASCII-Zeichen liegen dabei an den Ports 1 (High Byte) bzw. 2 (Low Byte) an, das Ergebnis wird auf Port 0 ausgegeben.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 3

BEISPIEL 8 – BITMUSTER

Es ist ein Programm zu erstellen, das auf den Ports 1 und 2 in 100-ms-Schritten (abgestimmt über Timer 0) nachfolgendes Muster periodisch ausgibt. Die Anzahl der High Bytes erhöht sich auf beiden Ports von „innen“ in jedem Schritt bis zum einschließlich achten um 1 und nimmt danach wieder ab.

Zeit	Port 1	Port 2
0ms	0000 0001	1000 0000
100ms	0000 0011	1100 0000
200ms	0000 0111	1110 0000
300ms	0000 1111	1111 0000
…	…	…

Zeit	Port 1	Port 2
…	…	…
1100ms	0000 1111	1111 0000
1200ms	0000 0111	1110 0000
1300ms	0000 0011	1100 0000
0/1400ms	*0000 0001*	*1000 0000*

BEISPIEL 9 – PULSWEITENMODULATION

Es ist ein Programm zu erstellen, welches am Port 1.1 ein pulsweitenmoduliertes Signal mit einer Frequenz von 500 µs ausgibt. Es ist der Timer 0 als Timerinterrupt zu verwenden. Das PWM-Verhältnis ist in der unsigned-char-Variable PWM_Relation zu setzen. Für diese Variable gelten folgende Bestimmungen:

PWM_Relation	High-Time	Low-Time
$\text{PWM\_Relation} < 30$	$0\ \mu s$	$500\ \mu s$
$30 \leq \text{PWM\_Relation} \leq 225$	$\frac{\text{PWM\_Relation}}{255} \cdot 500\ \mu s$	$500\ \mu s - \text{HighTime}$
$\text{PWM\_Relation} > 225$	$500\ \mu s$	$0\ \mu s$

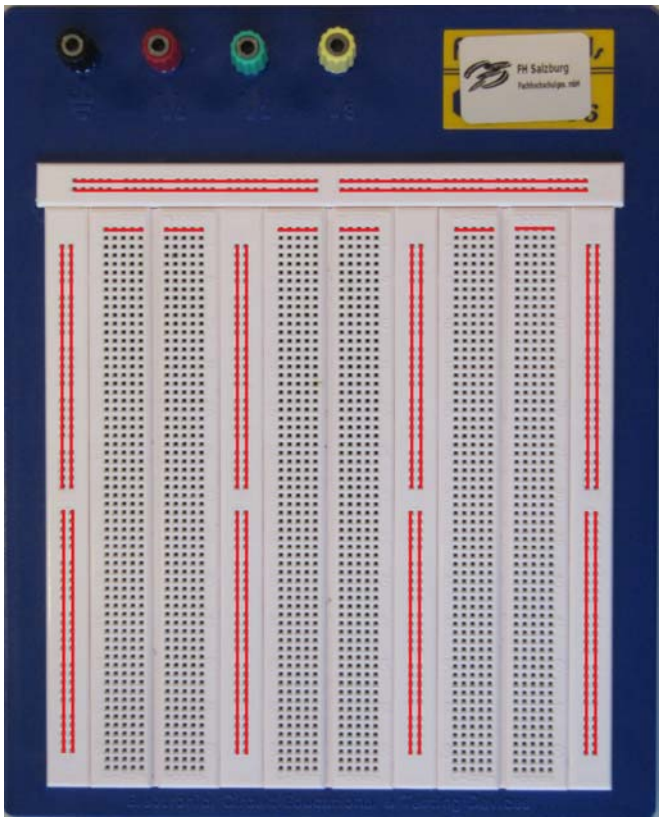
LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 4

BEISPIEL 10 – TRIAC

Erstellen Sie ein Programm, das einen Triac zur Phasenanschnittsteuerung (siehe Vorlesung) zeitgenau einschaltet. Es wird dabei davon ausgegangen, dass der Nulldurchgang der Netzspannung durch eine fallende Flanke an Port 3, Pin 2 (externer Interrupt 0) signalisiert wird, wonach mit einer einstellbaren Verzögerung (verwenden Sie dazu Timer 0) der Triac zur Phasenanschnittsteuerung gezündet werden muss, was durch einen Zündimpuls (High-Pegel) an Port 1, Pin 0 für die Dauer von ca. 10µs (kein Timer notwendig) geschieht. Beginnen Sie mit einer fixen Verzögerung von (genau!) 5ms und erweitern Sie nach Verifikation der Funktionsfähigkeit das Programm so, dass die Verzögerung in ms von Port 2 eingelesen werden kann. Als Frequenz der Netzspannung werden 50Hz angenommen, woraus sich die maximale Anschnittdauer errechnen lässt, die die maximal mögliche Verzögerung limitiert. Berücksichtigen Sie dieses Limit in Ihrem Programm.

BEISPIEL 11 – STECKBRETT

Für diese Übung soll ein Mikrokontroller vom Typ Atmel AT89C52 auf einem Steckbrett richtig beschalten werden. Der Mikrokontroller ist bereits vorprogrammiert und toggelt Port 1 im 500ms-Takt. Die Anschlussbelegung für den Atmel AT89C52-Mikrokontroller ist im Datenblatt-AT89C52.pdf im eLearning-Kurs zu finden. Zudem soll das Toggeln von Port 1 mit einer LED Beschaltung am Pin 1.0 sichtbar dargestellt werden. Die LED soll dabei von der Spannungsversorgung (5V) und nicht vom Pin getrieben werden. Nachfolgend ist die Durchkontaktierung des Steckbrettes dargestellt:



BEISPIEL 12 – GETAKTETES LATCH

Der Status von Pin 2.0 wird über Pin 0.0 definiert. Dieser soll nur durchgeschaltet werden, wenn auf dem externen Interrupt 0 eine fallende Flanke auftritt. Der externe Interrupt wird durch das Toggeln am Port 1 ausgelöst (physikalische Verbindung). Zur Visualisierung wird eine LED mit Vorwiderstand am Pin 2.0 gegen 5V geschalten. Für die bessere Nachvollziehbarkeit des Durchschaltvorganges soll das Timer-Intervall von 500ms auf eine Sekunde angehoben werden. Die Programmierung des Mikrokontrollers erfolgt mit dem „GALEP Programmer“.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 5

BEISPIEL 13 – INBETRIEBNAHME ENTWICKLUNGSBOARD

Nehmen Sie das MKS-EVAL51SND1-Entwicklungsboard anhand der zur Verfügung gestellten Anleitung in Betrieb und verifizieren Sie dessen Funktionsfähigkeit anhand der Anleitung erstellten Hello-World-Blink-Programms.

BEISPIEL 14 – ANPASSUNG UND TRANSFER BEISPIEL 9

Passen Sie das in Beispiel 9 erstellte Programm derart an, dass es auf dem Entwicklungsboard lauffähig ist, d.h. an Stelle von Port 1 den in Beispiel 13 beschriebenen Port zur Ausgabe verwendet, da nur an diesem LEDs zur Visualisierung der jeweiligen Signalpegels vorhanden ist. Verifizieren Sie die Funktionsfähigkeit des Programms für verschiedene Werte von PWM_Relation und passen Sie das PWM-Intervall so an, dass die Visualisierung besser erfassbar ist (Sekunden-Bereich).

BEISPIEL 15 – RINGPUFFER

Implementieren Sie einen Ringpuffer, der 20 Elemente (Byte), fassen kann, indem Sie ein Array entsprechender Größe anlegen und Funktionen zum Hinzufügen sowie zum Entnehmen einzelner Elemente bereitstellen. Fügen Sie der Reihe nach die nachfolgend aufgezählten Elemente dem Ringpuffer hinzu und lesen Sie nach jedem Aufzählungspunkt ein Element aus. Kommentieren Sie die zurückgelieferten Elemente sowie den Inhalt des Arrays zu jedem Zeitpunkt.

- *0x01*
- *0x02, 0x03, 0x04*
- *0x05*, *0x06*, *0x07*, *0x08*, *0x09*, *0x0A*
- 10-mal *0x10*
- *0x0B*
- 5-mal *0x20*
- *0x0C*
- 21-mal *0x30*

BEISPIEL 16 – RINGPUFFER AM ENTWICKLUNGSBOARD

Verwenden Sie die Funktionen zum Zugriff auf den Ringpuffer aus Beispiel 15 zur Erstellung eines Programms, das alle 500ms ein Element auf dem Ringpuffer entnimmt und an jenem Port ausgibt, an dem die LEDs zur Visualisierung des Signalpegels anliegen (vgl. Beispiel 14). Zusätzlich soll alle n ms ein Element dem Ringpuffer hinzugefügt werden, und zwar derart, dass der Wert jedes Elements um 1 höher ist als das des zuletzt hinzugefügten, beginnend bei 0, sich wiederholend nach 15. Untersuchen und kommentieren Sie das Laufzeitverhalten des Programms für n=500, 250, 100 und 1000. Achten Sie dabei auf die exakte, d.h. µs-genaue, Einhaltung der Zeitspannen. Kommentieren Sie zudem die Änderung des Verhaltens bei der halben bzw. der doppelten Größe des Ringpuffers.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 6

BEISPIEL 17 – UART/SERIELLE SCHNITTSTELLE MITTELS TIMER 1

Schreiben Sie ein Programm, das zyklisch „Hello world“, gefolgt von einem Zeilenumbruch, auf der seriellen Schnittstelle mit einer Baud-Rate von 9600 in Form von 8 Bit inkl. Stopp-Bit ausgibt. Setzen Sie dazu die betreffenden Register (SCON, PCON etc., sowie etwaige Timer-Register) entsprechend und realisieren Sie die Taktung der Baud-Rate über Timer 1, d.h. verwenden Sie einen jener seriellen Modi, die auf der Baud-Raten-Generierung über Timer 1 im 8-Bit-Auto-Reload-Modus) basieren. Gehen Sie dabei von jener Oszillatorfrequenz aus, die auf dem MKS-EVAL51SND1-Entwicklungsboard verwendet wird und verifizieren Sie Ihr Programm in der Keil-IDE sowie über den mit einem PC verbundenen seriellen Ausgang des Entwicklungsboards nach dem Aufspielen des Programms auf letzteres.

BEISPIEL 18 – UART MITTELS INTERNEM BAUD-RATEN-GENERATOR

Passen Sie das Programm aus Beispiel 17 so an, dass es zur Baud-Raten-Generierung den internen Baud-Raten-Generator anstatt Timer 1 verwendet. Beachten Sie dabei die zusätzlich zu verwendenden Register laut Manual und verifizieren Sie das Programm wie in Beispiel 17. Je nach Maßgabe der Zeit ist die Implementierung der Ausgabe mittels eines Ringpuffers zu realisieren.

BEISPIEL 19 – SERIELLE KOMMUNIKATION (OPTIONAL)

Verbinden Sie den seriellen Ausgang des Entwicklungsboards mit einem PC und schreiben Sie ein Programm, das vom PC gesendete Daten von der seriellen Schnittstelle zeichenweise als Ziffer interpretiert und den jeweiligen Ziffernwert an den LEDs aus Beispiel 14 in binärer Form anzeigt. Wird ein Zeichen empfangen, das keiner Ziffer entspricht, so sollen alle LEDs leuchten, um einen Fehler zu signalisieren. Verwenden Sie zur seriellen Übertragung 8 Bit inkl. Stopp-Bit (siehe Beispiel 17) und beginnen Sie mit einer Baud-Rate von 9600. Passen Sie diese gegebenenfalls an, um die Ziffern-Visualisierung verifizieren zu können. Je nach Maßgabe der Zeit ist jedes gesendete Byte über die serielle Schnittstelle wie nachfolgend beschrieben zu beantworten und die Antwort über das Windows Hyperterminal zu verifizieren. Jede Ziffer wird dabei unverändert zurückgeschickt, während alle anderen Bytes mit 0xFF bestätigt werden.