LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 1

BEISPIEL 1 – IDE

Starten Sie die Keil µVision4-IDE und erstellen Sie anhand der zur Verfügung gestellten Anleitung ein Beispielprojekt. Führen Sie die darin beschriebenen Arbeitsschritte durch und verfolgen Sie die Pegelwechsel der Pins von Port 1 im Einzelschrittmodus des Debuggers.

AUFGABE I – ASSEMBLER-CODE-ANALYSE

Analysieren Sie den vom Compiler generierten Assemblercode von Beispiel 1 in der Disassembler-Ansicht des Debuggers und ermitteln Sie die Größe, die der gesamte Code einnimmt, sowie die Ausführungsdauer und Größe der einzelnen Assembleranweisungen bei einem gegebenen Standardtakt von 12 MHz. Nutzen Sie dazu die Informationen in den Kapiteln 1.3 bis 1.9 des zur Verfügung gestellten Manuals sowie die Befehlssatzreferenz des 8051-Mikrocontrollers.

AUFGABE II – PINBELEGUNG DES AT89C52-MIKROCONTROLLERS

Machen Sie sich anhand des Datenblattes des AT89C52-Mikrocontrollers mit dessen Pinbelegung vertraut und bestimmen Sie die für einen manuellen Versuchsaufbau relevanten Pins sowie die zusätzlich notwendigen Bauelemente zur Spannungsversorgung und Taktgenerierung.

Legende	
BEISPIEL 1	Während der Laborübung fertigzustellendes Übungsbeispiel mit arabischer Nummerierung
AUFGABE I	Bis zur nächsten Laborübung fertigzustellendes Übungsbeispiel mit römischer Nummerierung
AUFGABE A	Bis zur Projektabgabe fertigzustellendes Übungsbeispiel mit alphabetischer Nummerierung

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 2

BEISPIEL 2 – STECKBRETT

Beschalten Sie den Mikrokontroller vom Typ Atmel AT89C52 auf einem Steckbrett. Der Mikrokontroller ist bereits vorprogrammiert und toggelt Port 1 im 500ms-Takt, was durch eine LED-Beschaltung am Port 1, Pin 0 sichtbar dargestellt werden soll. Die LED soll dabei von der Spannungsversorgung (5V) und nicht vom Pin getrieben werden. Die Anschlussbelegung für den Mikrokontroller ist im Datenblatt aus Beispiel II zu finden, die Durchkontaktierung des Steckbrettes im E-Learning-Kurs.

BEISPIEL 3 – INTERRUPT-HANDLING AM STECKBRETT

Schreiben Sie ein Programm für den Mikrocontroller AT89C52, das Port 1, Pin 0 toggelt, wenn eine fallende Flanke an Port 3, Pin 2 anliegt. Nutzen Sie zur Reaktion auf dieses Ereignis den externen Interrupt 0 und schreiben Sie eine geeignete Interrupt-Service-Routine (ISR). 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. Modifizieren Sie anschließend den Aufbau aus Beispiel 2 derart, dass Sie das programmierte Beispiel testen können. Nutzen Sie zur Programmierung des Mikrocontrollers den „GALEP Programmer“.

AUFGABE III – INTERRUPT-HANDLING MIT ZÄHLER

Modifizieren Sie das Programm aus Beispiel 3 derart, dass es beim Auftreten eines externen Interrupts einen Zähler inkrementiert anstatt einen Pin zu toggeln. Erstellen Sie dazu ein neues Projekt für einen AT89C51SND1-Mikrocontroller und verwenden Sie dieses als Basis für Ihr Laborprojekt.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 3

BEISPIEL 4 – BITMUSTER

Schreiben Sie ein Programm, 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 (vgl. Tabelle unten). Machen Sie sich zur Realisierung der Zeitsteuerung mit der Funktionsweise und den unterschiedlichen Modi von Timer 0 anhand des Datenblattes vertraut und wählen Sie einen geeigneten Modus. Versuchen Sie durch Assemblercodeanalyse eine mikrosekundengenaue Ausgabe zu erreichen und verifizieren Sie die Genauigkeit durch Simulation.

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*

AUFGABE IV – TACHOSIGNALAUSWERTUNG MITTELS TIMER

Erweitern Sie Ihr Projekt aus Beispiel III derart, dass mittels Timer 1 die Zeitdauer zwischen dem Auftreten von n fallenden Flanken an Port 3, Pin 2 gemessen wird. Der Wert von n kann dabei beliebig gewählt und mit dem bereits implementierten Zähler abgeglichen werden. Testen Sie Ihr Programm in der Keil µVision-IDE und erweitern Sie es anschließend um eine Funktion, die Port 3, Pin 2 als Tachosignal eines Lüfters interpretiert (zwei Rechteckimpulse pro Lüfterumdrehung) und die aktuelle Drehzahl in Umdrehungen pro Sekunde (rps) und Umdrehungen pro Minute (rpm) umrechnet. Eine mikrosekundengenaue Berechnung ist zu diesem Zeitpunkt noch nicht notwendig.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 4

BEISPIEL 5 – PULSWEITENMODULATION

Erstellen Sie ein Programm, welches an Port 4, Pin 0 ein pulsweitenmoduliertes Signal (PWM-Signal) mit einer Frequenz von 100 Hz ausgibt. Das Tastverhältnis soll über eine unsigned-char-Variable zwischen 0 und 100(%) variiert werden können. Nutzen Sie Timer 0 zur Generierung dieses Signals und beachten Sie die Sonderfälle 0 und 100 des Tastverhältnisses. Berechnen Sie, wie hoch die Frequenz des PWM-Signals bei Verwendung Ihres Programms auf einem AT89C51SND1-Mikrocontroller minimal und maximal sein kann, wenn ein Tastverhältnis von 1% und 99% möglich sein soll und diskutieren Sie die Änderung dieses Frequenzbereichs bei der Lockerung der unterstützten Tastverhältnisse (z.B. zwischen 5 und 95%).

BEISPIEL 6 – PWM-LÜFTERSTEUERUNG

Integrieren Sie das Programm aus Beispiel 5 derart in Ihr Projekt, dass das PWM-Signal einen an Port 4, Pin 0 angeschlossenen PWM-Lüfter steuern kann. Beachten Sie dazu die Spezifikation für 4-Pin-Lüfter im E-Learning-Kurs. Wählen Sie eine geeignete Frequenz, die von Ihrem Programm auf einem AT89CD51SND1-Mikrocontroller für alle ganzzahligen, prozentuellen Tastverhältnisse unterstützt werden kann und verwenden Sie dazu Ihre Überlegungen aus Beispiel 5. Stellen Sie weiters Überlegungen zu den Interruptprioritäten der Timer und des externen Interrupts an und passen Sie diese anhand des Datenblattes an die Bedürfnisse Ihres Programmes an.

AUFGABE A – GENAUES TACHOSIGNAL

Erweitern Sie Ihren Projektteil aus Beispiel IV derart, dass die fallenden Flanken des Tachosignals mikrosekundengenau erfasst werden und bestimmen Sie die rpm-Differenz zu Ihrer bisherigen Implementierung.

AUFGABE B – GENAUES PWM-SIGNAL

Erweitern Sie Ihren Projektteil aus Beispiel 6 derart, dass das PWM-Signal mikrosekundengenau generiert wird und ermitteln Sie die neuen minimal und maximal unterstützten Frequenzen Ihres Programmes sowie die deren Differenz zu den bisher unterstützten minimalen und maximalen Frequenzen.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 5

BEISPIEL 7 – TRANSFER DES PROJEKTES AUF DAS EVALUATIONSBOARD

Nehmen Sie das MKS-EVAL51SND1-Evaluationsboard anhand der zur Verfügung gestellten Anleitung in Betrieb und verifizieren Sie dessen Funktionsfähigkeit anhand des in der Anleitung erstellten Hello-World-Blink-Programms. Transferieren Sie Ihr Projekt anschließend mittels Flip auf das Evaluationsboard und verbinden Sie den zur Verfügung gestellten 4-Pin-PWM-Lüfter mit den entsprechenden Pins des Evaluationsboards. Testen Sie die Tachosignalmessung und die PWM-Signalgenerierung für unterschiedliche Tastverhältnisse und überprüfen Sie Ihre Messungen mittels Oszilloskop und/oder Multimeter.

BEISPIEL 8 – LCD-ANSTEUERUNG

Schreiben Sie ein Modul für Ihr Projekt, das das LC-Display des Evaluationsboards via I/O-Mapping initialisieren, löschen und mit Zeichen beschreiben kann. Nutzen Sie dazu die Initialisierungssequenz im zur Verfügung gestellten Datenblatt. Verifizieren Sie die Funktionsfähigkeit Ihres Moduls mittels eines Testprogrammes, das „Hello world“ auf dem LC-Display des Evaluationsboards ausgibt.

AUFGABE V – DREHZAHLAUSGABE AUF DEM LC-DISPLAY

Erweitern Sie Ihr Projekt derart, dass die über das Tachosignal gemessene Lüfterdrehzahl in rpm auf dem LC-Display angezeigt wird.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 6

BEISPIEL 9 – 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 10 – UART MITTELS INTERNEM BAUD-RATEN-GENERATOR

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.) entsprechend und realisieren Sie die Taktung der Baud-Rate über den internen Baudratengenerator. Beachten Sie dabei die zusätzlich zu verwendenden Register laut Manual und gehen Sie dabei von jener Oszillatorfrequenz aus, die auf dem MKS-EVAL51SND1-Entwicklungsboard verwendet wird. Verifizieren Sie Ihr Programm in der Keil µVision-IDE sowie über den mit einem PC verbundenen seriellen Ausgang des Evaluierungsboards nach dem Aufspielen des Programms und der Software „Hyper Terminal“. Je nach Maßgabe der Zeit ist die Implementierung der Ausgabe mittels eines Ringpuffers zu realisieren.

AUFGABE VI – DREHZAHLAUSGABE VIA UART

Erweitern Sie Ihr Projekt derart, dass die über das Tachosignal gemessene Lüfterdrehzahl in rpm auf der seriellen Schnittstelle ausgegeben wird. Nutzen Sie zur Zwischenspeicherung der Ausgabedaten einen Ringpuffer.

AUFGABE C – LÜFTERSTEUERUNG VIA UART

Erweitern Sie Ihr Projekt derart, dass das PWM-Tastverhältnis zur Lüftersteuerung über die serielle Schnittstelle eingelesen und gesetzt werden kann. Das Eingabeformat ist dabei frei wählbar.

LABORÜBUNGEN MIKROCONTROLLER-PROGRAMMIERUNG – ÜBUNG 7

BEISPIEL 11 – LED-ANSTEUERUNG ÜBER I²C

Nutzen Sie die zur Verfügung gestellte minimale I²C-Bibliothek zum Auslesen und Setzen der Werte der über den I²C-Bus angebundenen LEDs und geben Sie das Bitmuster aus Beispiel 4 auf der oberen und unteren LED-Reihe anstatt auf den Ports 1 und 2 aus. Beachten Sie dabei die Anordnung und I²C-Adressen der LEDs laut Datenblatt (die oberen 4 Bit sind laut Datenblatt vorgegeben, die Bits A1 und A2 der unteren Bits laut Datenblatt sind am Evaluierungsboard 1, A0 ist 0 für den linken und 1 für den rechten LED-Block).

BEISPIEL 12 – A/D-WANDLER-ANSTEUERUNG ÜBER I²C

Nutzen Sie die zur Verfügung gestellte minimale I²C-Bibliothek zum Auslesen der Stellung der Drehpotentiometer auf den über den I²C-Bus angebundenen A/D-Wandlern und nutzen Sie eines der Drehpotentiometer zur Steuerung der Lüftergeschwindigkeit, indem Sie die minimale bzw. maximale Stellung des Drehpotentiometers mit dem minimalen bzw. maximalen PWM-Tastverhältnis verknüpfen und bei einer Stellungsänderung eine entsprechende Tastverhältnisänderung vornehmen.

AUFGABE 13 – INTERNER A/D-WANDLER

Konfigurieren Sie den internen A/D-Wandler über die entsprechenden Register, sodass eine Analogspannung zwischen 0 und +3,3V mit 10 Bit Genauigkeit (8 Bit effektiv) in einen Digitalwert gewandelt wird. Verwenden Sie den eingelesenen Wert zur Steuerung des PWM-Tastverhältnisses analog zu Beispiel 12. Schließen Sie wahlweise eine Spannungsquelle an die Pins des internen A/D-Wandlers an oder verwenden Sie das Potentiometer bzw. den Temperatursensor auf dem zur Verfügung gestellten Zusatzboard.

AUFGABE VII – LED-LAUFLICHT

Erweitern Sie Ihr Projekt derart, dass die untere LED-Reihe des Evaluationsboards ein Lauflicht (ein gesetztes Bit) anzeigt, das in beliebigem, aber für das Auge sichtbarem Tempo von links nach rechts wandert.

AUFGABE VIII – LED-GESCHWINDIGKEITSANZEIGE

Erweitern Sie Ihr Projekt derart, dass die obere LED-Reihe des Evaluationsboards die gewünschte Lüftergeschwindigkeit bzw. das aktuelle PWM-Tastverhältnis in Form eines Fortschrittsbalkens anzeigt. Bei einem Tastverhältnis von 0 soll dabei keine LED leuchten, bei einem Tastverhältnis von 50 die linken 4 LEDs und bei einem Tastverhältnis von 100 alle LEDs leuchten. Eine Binär- oder BCD-Darstellung des Wertes ist dabei nicht gestattet.