Hard- und Software-Engineering HS

– Merkblatt zur Klausur –

22. 1. 2007

1. Themen:

- Aufbau und Einsatz von Mikrocontrollern
- universelle E-A-Ports
- zweckgebundene E-A-Ports
- externe Porterweiterung
- periphere Funktionseinheiten in Mikrocontrollern
- Programmiermodelle
- Grundlagen des Multitasking
- Mehrprozessorkonfigurationen
- Entwerfen mit programmierbaren Logikschaltkreisen
- Assemblerprogrammierung am Beispiel Atmel AVR
- Betrieb von LCD-Displays mit Busschnittstelle

2. Informationsquellen:

Skript-Material
S. Internetseite.

Zur Programmierung des Atmel AVR:

- Einführung in die Mikrocontroller-Programmierung am Beispiel Atmel AVR
- Die Original-Befehlsbeschreibung der Fa. Atmel (8 Bit AVR Instruction Set). Im Grunde genügt die Kurzübersicht der Seiten 10 bis 15 (Instruction Set Summary)
- Beliebige Lehrbücher zur AVR-Programmierung
- einschlägiges Material aus dem Internet

Alternative/Ergänzung
Beliebige Lehr- und Tabellenbücher zu den genannten Themen.

3. Die Klausuraufgaben umfassen:

- Wissensfragen zum Entwickeln mit Mikocontrollern und programmierbare Logik sowie zu Problemen der Software (Programmorganisation, Multitasking usw.),
- Entwicklung elementarer Schaltungslösungen auf Grundlage von Mikrocontrollern und CPLDs,
- elementare Asssemblerprogrammierung am Beispiel Atmel AVR (Schreiben kleiner Programmstücke).

4. Übungsaufgaben

Hinweis: Weitere Übungsgelegenheiten finden Sie in den bisherigen Klausuren (Internet).

1. Es ist zyklisch (mit Takt CLK) ein Impulsmuster gemäß Abb. 1 zu erzeugen. Entwerfen Sie eine entsprechende Schaltung zur Implementierung in einem programmierbaren Logikschaltkreis.

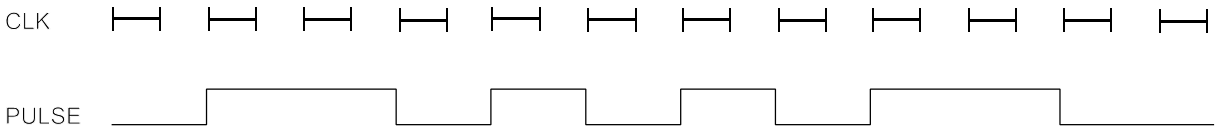

Abb. 1

2. Erläutern Sie kurz (mit Skizze), wie ein einfaches Schieberegister-Interface (für parallele Ein- und Ausgabe) aufgebaut ist. Welchen Vorteil hat dieses Prinzip? Nennen Sie wenigstens zwei Schieberegister-Interfaces, die als Industriestandard anzusehen sind. Skizzieren Sie eine Schaltungslösung (eine Bitposition genügt), die sich zur Implementierung mit CPLDs eignet. Wieviele CPLD-Zellen brauchen Sie für eine Bitposition?

3. Erläutern Sie kurz, worin sich GALs und CPLDs voneinander unterscheiden.

4. Nennen Sie wenigstens zwei Prinzipien der Parameterübergabe an Unterprogramme. Veranschaulichen Sie diese Prinzipien anhand kurzer Programmstücke (AVR-Assemblerprogrammierung). Das Unterprogramm soll SPECIAL heißen. Es sollen drei Parameter P1, P2, P3 (jeweils ein Byte) übergeben werden. P1 ist ein Direktwert, P2 steht in Register r12 und P3 steht im SRAM unter der symbolischen Adresse EX_P.

5. Es geht um einen Testprogrammablauf. Über Port A eines AVR-Mikrocontrollers ist ein Datenbyte auszugeben und zur Kontrolle wieder zurückzulesen. Das auszugebende Datenbyte steht in r16. Der Kontrollwert soll nach r20 gebracht werden. Geben Sie eine geeignete Befehlsfolge an. Worauf ist hierbei besonders zu achten?

6. Parallele Bussysteme kann man auf vielfältige Weise aufbauen. In der Praxis sind zwei Auslegungen weit verbreitet, die man als Intel-Ausführung und als Motorola-Ausführung bezeichnen könnte. Erläutern Sie kurz. worin die kennzeichnenden Merkmale beider Ausführungen bestehen. Nennen Sie jeweils einen typischen Anwendungsfall.

7. Die Register r2 bis r5 enthalten zwei 16 Bits lange Binärzahlen X und Y. Schreiben Sie einen Programmablauf (AVR Assembler) für die Subtraktion X-Y. Hierbei soll das Ergebnis in den Registern r2 und r3 zu stehen kommen.

r2	LO_X
r3	HI_X
r4	LO_Y
r5	HI_Y

8. Wir beziehen uns auf Aufgabe 7. Jetzt ist aber die Subtraktion X-Y so auszuführen, daß das Ergebnis in den Registern r18 und r19 zu stehen kommt. Die Inhalte der Register r2 bis r5 sollen unverändert erhalten bleiben.

9. Schreiben Sie einen Programmablauf, der über Port C, Bitpositionen 1 und 0 das in Abb. 2 gezeigte Bitmuster ausgibt.

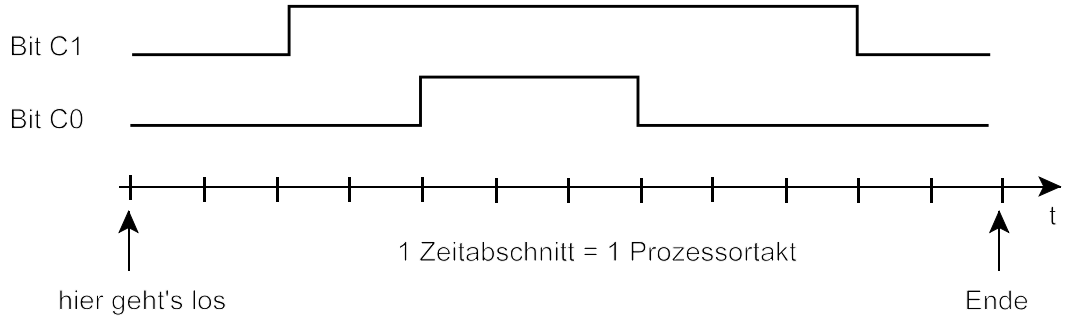

Abb. 2

10. Schreiben Sie eine Interruptroutine zur Bedienung des Analog-Digital-Wandlers. Die Datenregister sind einzulesen. Aus den 10 Bits ist ein 8-Bit-Wert zu bilden (Abb. 3) und über Port C auszugeben. Achten Sie darauf, daß das unterbrochene Programm nicht beeinträchtigt wird.

ADC Data Register ADCL und ADCH:

Register	7	6	5	4	3	2	1	0
ADCH	-	-	-	-	-	.	ADC9	ADC8
ADCL	ADC7							ADC0

Das gewünschte Ergebnis:

7	6	5	4	3	2	1	0
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2

Abb. 3

11. Schreiben Sie ein Assemblerprogrammstück, das folgenden Ablauf implementiert (Abb. 4).

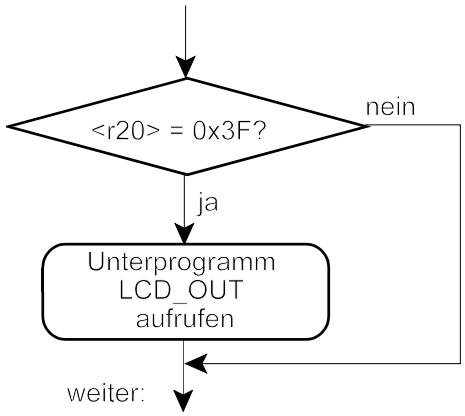

Abb. 4

Wichtige Maschinenbefehle

Befehl	Wirkung
LDI *rd, imm*	Lädt einen Direktwert in ein Register. Beispiel: *LDI r17, 0x22*
MOV *rd, rr*	Transportiert Inhalt des Registers rr in das Register rd. Beispiel: *MOV r2, r22*
LPM	Lädt Byte aus Programmspeicher in Register r0. Adresse in Register Z (r31, r30)
LDS	Lädt ein Byte aus dem SRAM in das Register rd. Beispiel: *LDS r2, BYTE_ADRS*
STS	Transportert den inhalt des Registers rs in den SRAM. Beispiel: *STS BYTE_ADRS, r2*
OUT *port, rr*	Ausgabe eines Registerinhaltes. Beispiel: *OUT portd, r17*
IN *rd, port*	Eingabe in ein Register. Beispiel: *IN r17, pind*
SBI *port, bit*	Setzen Bit in E-A-Port. Beispiel: *SBI porta, 3*
CBI *port, bit*	Löschen Bit in E-A-Port. Beispiel: *CBI porta, 3*
SBIC *port, bit*	Bit in E-A-Port abfragen. Folgebefehl überspringen, wenn = 0. Beispiel: *SBIC porta, 3*
SBIS *port, bit*	Bit in E-A-Port abfragen. Folgebefehl überspringen, wenn = 1. Beispiel: *SBIS porta, 3*
NOP	keine Wirkung (Leerbefehl). Es wird nur eine Taktperiode verbraucht
JMP *label*	Unbedinge Verzweigung. Beispiel: *JMP anfang*
BRNE *label*	Verzweigen, wenn Ergebnis ungleich Null. Beispiel: *BRNE anfang*
BREQ *label*	Verzweigen, wenn Ergebnis gleich Null. Beispiel: *BREQ ende*
CALL *label*	Unterprogrammruf. Beispiel: *CALL warten*
RET	Rückkehr aus Unterprogramm
RETI	Rückkehr aus Unterbrechungsbehandlung
PUSH *rr*	Register rr in Stack retten. Beispiel: *PUSH r20*
POP *rd*	Register rd aus Stack laden. Beispiel: *POP r20*
AND *rd, rs*	Register konjunktiv verknüpfen <rd> := <rd> AND <rr>. Beispiel: *AND r12, r3*
ANDI *rd, imm*	konjunktive Verknüpfung mit Direktwert. Beispiel: *ANDI r12*, 0b11110111
OR *rd, rs*	Register disjunktiv verknüpfen <rd> := <rd> OR <rr>. Beispiel: *AND r12, r3*
ORI *rd, imm*	disjunktive Verknüpfung mit Direktwert. Beispiel: *ORI r12*, 0b11110111
COM *rd*	Bitweise Negation (Einerkomplement). Beispiel: *COM r17*
ADD *rd, rs*	Registerinhalte zueinander addieren. <rd> := <rd> + <rr>. Beispiel: *ADD r12, r4*
ADC *rd, rs*	Addieren mit Eingangsübertrag. <rd> := <rd> + <rr>. + CF. Beispiel: *ADC r13, r5*
ADIW *rd, imm*	Addieren Direktwert zu Wort (Register 24, 26, 28, 30). Beispiel: *ADC r24, 12*
SUBI *rd, imm*	Subtrahieren Direktwert. Beispiel: *SBI r16, 0x04*
SUB *rd, rs*	Registerinhalte voneinander subtrahieren. <rd> := <rd> – <rr>. Beispiel: *SUB r12, r4*
SBC *rd, rs*	Subtrahieren mit Eingangsübertrag. <rd> := <rd> – <rr> – CF. Beispiel: *SBC r13, r5*
SBIW *rd, imm*	Subtrahieren Direktwert von Wort (Register 24, 26, 28, 30). Beispiel*: SBIW r26, 10*
CP *rd, rs*	Registerinhalte vergleichen (Subtraktion <rd> – <rs>). Beispiel: *CP r12, r22*
CPI *rd, imm*	Vergleichen mit Direktwert (Subtraktion <rd> – imm). Beispiel: *CPI r16, 0x33*
ROL *rd*	Linksrotieren über CF (CF => Bit 0, Bit 7 => CF): Beispiel: *ROL r17*
ROR	Rechtsrotieren über CF (CF => Bit 7, Bit 0 => CF). Beispiel: *ROR r17*
LSL	Linksverschieben. Bit 7 nach CF, Bit 0 wird mit 0 gefüllt. Beispiel: *LSL r17*
LSR	Rechtsverschieben. Bit 0 nach CF, Bit 7 wird mit 0 gefüllt. Beispiel: *LSR r17*

Achtung: Direktwertverknüpfungen nur mit Registern r16...r31. Weitere Einzelheiten s. Befehlsbeschreibung.