Name, Vorname, Matr.-Nr.: . ..... 20. Januar 2014

HAW Hamburg - Prüfungsklausur Computertechnik - WS 2013/14						
Punkte	16	14	20	20	30	100
	16	11	~~15~~ 20	15	25	95

+82P.

1 Serielle Übertragung von zwei Zeichen

Ein UART-Modul des Controllers LM3S9B92 sendet zunächst ein Frame mit den ASCII-Symbol 'g' im RS232-Standard mit dem Protokoll 8E1. Die Bitrate beträgt 4800 Bit/s.

Dann wird die UART umkonfiguriert. Jetzt wird das ASCII-Symbol '9' im Protokoll 7O2 mit 9600 Bit/s gesendet.

15 VON 15 LEISTUNGSPUNKTEN

	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0x0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x2...	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Tabelle 1: ASCII-Symbole (niedere Bits in Spalten, höhere Bits in Zeilen)

a) Zeichnen Sie in Bild 1 und 2 die entsprechenden Signale am Ausgangs-Pin des Controllers und auf der Leitung der RS232-Verbindung.
Bezeichnen Sie dort jedes Bit vollständig mit **Kurznamen** und **logischem Wert**.
Bezeichnen Sie **beide Achsen** so, als wenn ein Oszilloskop benutzt würde (siehe Kästen über den Bildern).

'g' = 0x67 = 0110 0111 8E1 => 0 1 1 0 0 1 1 1 0 1 1
St D E Stop

'9' = 0x39 = 0011 1001 7O2 => 0 1 0 0 1 1 1 0 1 1 1
0 Stop

... nächste Seite beachten

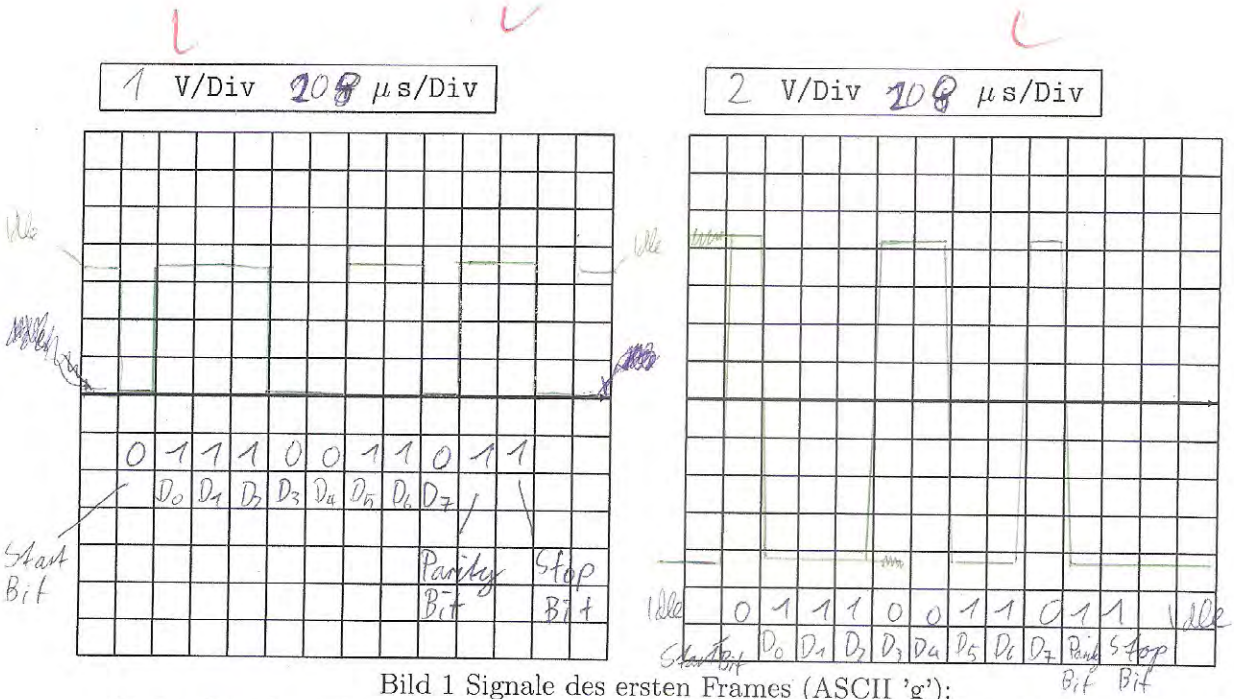


Bild 1 Signale des ersten Frames (ASCII 'g'):
Links: Signal am Ausgangspin des Controllers, Rechts: Signal auf der seriellen Leitung.

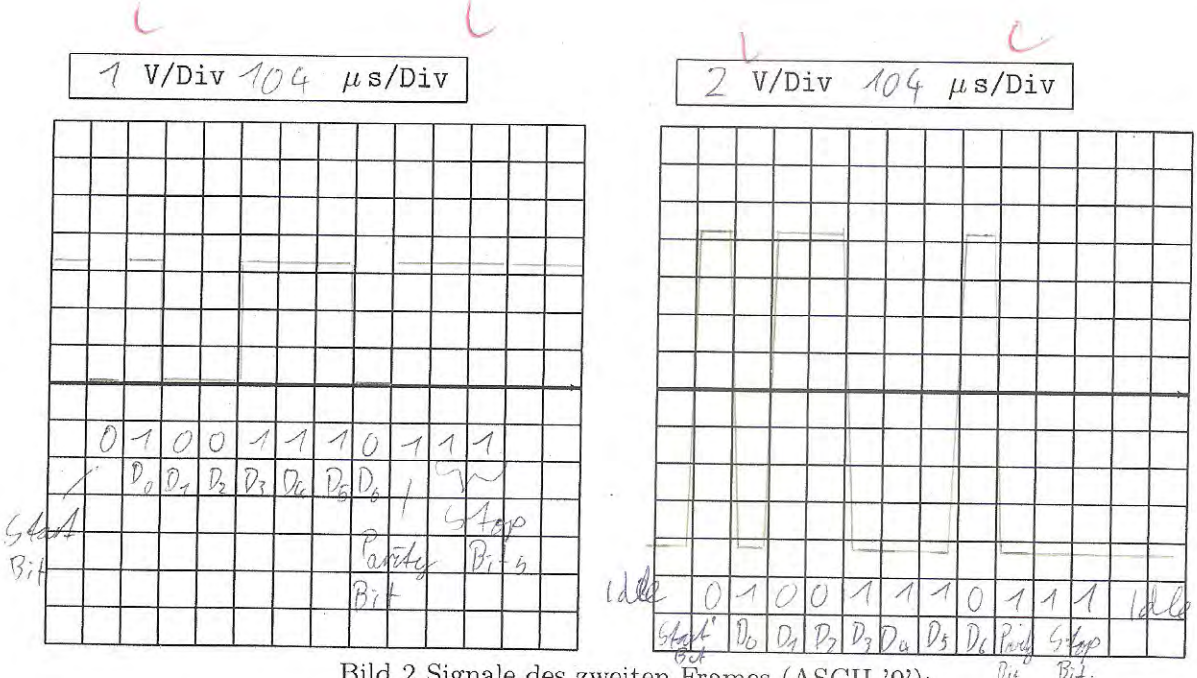


Bild 2 Signale des zweiten Frames (ASCII '9'):
Links: Signal am Ausgangspin des Controllers, Rechts: Signal auf der seriellen Leitung.

2 Architekturen

a) Nennen Sie die Bussysteme sowie die Funktionen in einem Computer mit einer von-Neumann-Architektur. Füllen Sie die Tabelle aus, geben Sie auch an, in welcher Richtung das Bussystem Werte transportiert. Nennen Sie außerdem die Busbreiten und die Kurznamen für mindestens 2 Bussysteme beim ARM-Controller LM3S9B92.

Bussysteme	Funktion	Richtung	Busbreite/Kurznamen
Control Bus	Regelt alle Dinge, die nichts mit Daten oder Adressen zu tun haben ? Um genau	CPU ⇄ Memory ⇄ I/O · z.T. bidirektional	32 Bit
Address Bus	Adressen werden per Broadcast von der CPU gesendet um einen Speicherbereich zu adressieren	I/O ↑ CPU → Memory unidirektional	32 Bit A31...A0
Data Bus	liefert Daten von der CPU zu den Speichermodulen und I/O Ports oder von den Speichermodulen und I/O Ports zur CPU und umgekehrt	CPU ⇄ Memory ⇄ I/O bidirektional	32 Bit D31...D0

6 v. 9

b) Nennen Sie die Unterschiede bei den Bussystemen in Bezug auf die Tabelle in Aufgabe a), wenn eine Harvard-Architektur betrachtet wird.

....

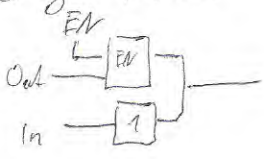
Bei der Harvard Architektur gibt es für den Code Memory ein eigenes Bussystem zusätzlich
Hier liegen nur die Programme.

2 v. 2

c) Erklären Sie, wie ein Bussystem bidirektional genutzt werden kann. Welches Problem ist zu überwinden, weil digitale Logik vorliegt ? Welche Lösung wird benutzt ?

....

Das Problem ist, dass wenn zwei Ports auf der gleichen Busleitung gleichzeitig Senden können die Signale nicht mehr verstanden werden.
Daher setzt man Tri-State Ein- bzw. Ausgänge ein.



3 v. 3

11 v. 14

3 Paritätsgenerator

Es soll ein Paritätsgenerator mit dem Controller LM3S9B92 erstellt werden. Am Port E Pin PE1 ist ein Schalter zum Umschalten zwischen der Parität für 8 oder 7 Bit vorhanden. Am Port E Pin PE0 wird zwischen der geraden und ungeraden Parität umgeschaltet. Es werden acht Schalter an Port D für die Eingabe von 8 bzw. 7 Bit angeschlossen, ggf. wird der Schalter an PD7 nicht beachtet. Am Port J sind 8 LED angeschlossen, welche die Schalterstellung von Port D anzeigen. Als Ergebnis wird Parität wird an Port E Pin PE3 mit einer LED angezeigt. Siehe Bild 3.

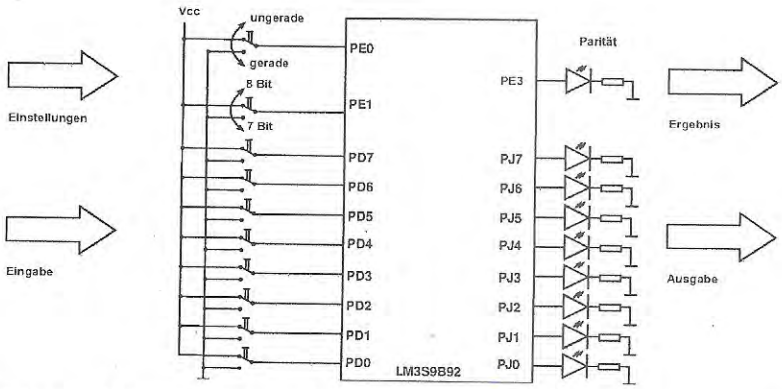


Bild 3: Übersicht über den zu erstellenden Paritätsgenerator

a) Programmieren Sie ein gut kommentiertes Programm, das fortlaufend die Schalter auswertet und die LED-Ausgaben vornimmt.

```
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"
void main(void)
{
int cw=0;
// Quarz Oszillator auf 16 MHz
SYSCTL_RCC_R = ((SYSCTL_RCC_R (0x00000540) & ~0x00002B1);
cw++; // auf stabilen Clock warten
SYSCTL_RCC_R |= 0x00000002; // Internen RC-OSC deaktivieren
// Aktivierung der Clocks
SYSCTL_RCGC2_R |= 0x118  // Clock für Ports D, E und J
cw++;
// Initialisieren der GPIO-Ports
GPIO_PORTJ_DEN_R = 0xFF;   // Port J alle Ports
GPIO_PORTJ_DIR_R = 0xFF;   // als outputs
GPIO_PORTD_DEN_R = 0xFF;   // Port D alle Ports
GPIO_PORTD_DIR_R = 0x00;   // als inputs
GPIO_PORTE_DEN_R = 0x0A;   // Port E Pins 0,1 & 3
GPIO_PORTE_DIR_R = 0x08;   // 0,1 inputs 3 output
// Dauerschleife folgt auf der Rückseite
```

#define Port_D GPIO_PORTD_DATA_R
#define Port_J GPIO_PORTD_DATA_R
#define Port_E GPIO_PORTE_DATA_R
// um den Code zu verkürzen möglich

20 v. 20

20 v. 20

```
// Dauerschleife

while(1)
{
  char c=0, temp=0;                    // Variablen Deklarationen
  int parity=0, odd=0, eight=0;

  odd = (Port_E & 0x01;)               // Paritätsschalter abfragen
  eight = (Port_E & 0x02;)             // 8Bit-Schalter abfragen

  if (eight) c = Port_D;               // Wenn 8 Bit dann alle einlesen
  else  c=(Port_D & 0x7F);             // Wenn 7 Bit nur diese einlesen

  temp = c                             // Aktuellen Wert zwischenspeichern
  if(odd) parity=1;                    // Wenn ungerade parität parity auf 1, da bei 0x00 eine ergänzt werden müsste
  while (temp)                         // Schleife um die Parität festzustellen
  {
     if(parity) parity=0               // parity boolsch gesehen
     else parity=1                     // invertieren
     temp&= (temp-1);                  // temp mit (temp-1) und-verknüpfen
  }

  Port J = c ;                         // LEDs an Port J setzen
  if(parity) Port_E |=0x08;            // Wenn ein Paritätsbit seien sollte die LED an PE3 anschalten
  else  Port_E & ~(0x08);              // sonst ausschalten/aus lassen

}
```

(X)

doch so!! kompliziert aber möglich

(X) Das geht) ~~hier nicht so~~! (Einfacher: Bitschieben, Abfragen, & XOR!)

volle Punkte dafür

=> s. Seite zuvor

4 Adressen und Fragen

An einen ARM-Cortex-Controller werden drei Speichermodule ROM, RAM1 und RAM2 mit den in Bild 4 dargestellten Adressdecodern angeschlossen.

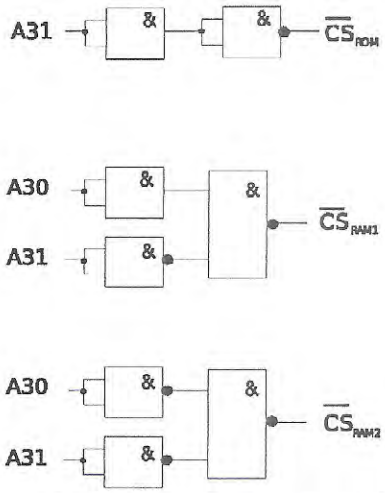


Bild 4: Schaltungen der Adressdecoder der Speichermodule ROM, RAM1 und RAM2

a) Nennen die Anfangs- und die Endadressen der drei Module.

	Anfangsadresse	Endadresse
ROM	0x80000000	0x FFFF FFFF
RAM1	0x40000000	0x 7FFFFFFF
RAM2	0x00000000	0x 3FFF FFFF

9 v. 9

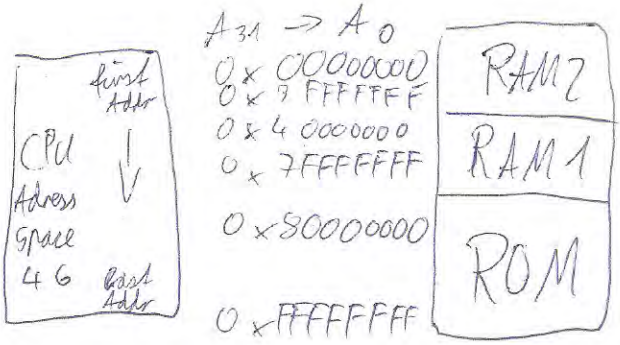
b) Nennen Sie die Kapazität der drei Module.

ROM = 2 GB = 2048 MB

RAM jeweils = 1 GB = 1024 MB

3 v. 3

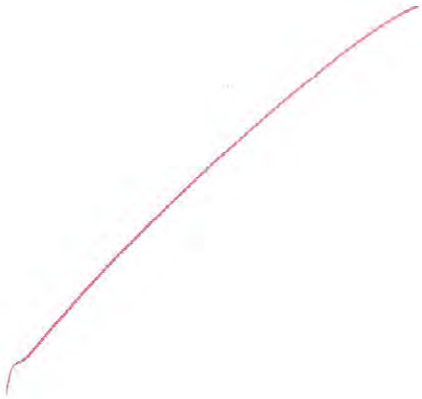
c) Zeichnen Sie die Memory-Map des Controllers mit den drei Modulen, inkl. Adressangaben.



3 v. 3

g) Erklären Sie den Unterschied zwischen einer Interruptservice-Routine (ISR) und einer gewöhnlichen Subroutine.
Kann eine Subroutine von einer Interruptservice-Routine aufgerufen werden ?
Kann eine Service-Routine ablaufen, bevor eine Subroutine beendet ist ?
.....

0
v. 3



g) Nennen Sie die grundsätzlichen Phasen des Ablaufs eines Maschinenbefehls beim ARM-Cortex-Controller.
Worin unterscheiden sich hierbei die Registerbefehle von Speicherbefehlen (Load/Store)?
.....

0
v. 2

15
v. 20

3 Fernsteuerung über UART

a) Ein Controller LM3S9B92 in einer Maschinenfernsteuerung nutzt eine RS232-Verbindung. Sie wird mit Tastern an den Portpins PD0 (EIN-Taster) und PD1 (AUS-Taster) bedient. Die Taster legen bei Betätigung einen LOW-Pegel an den Controller-Pin an. Ein hochohmiger Pull-Up-Widerstand sorgt ansonsten für einen HIGH-Pegel. Bei Tastenbetätigung werden die Kontroll-LEDs an den Pins PD4 (AUS) und PD5 (EIN) entsprechend geschaltet.

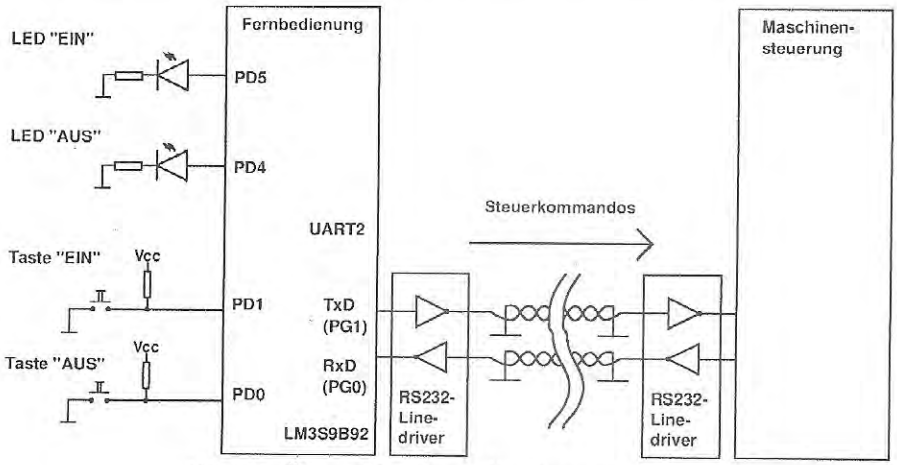


Bild 5: Übersicht über die Fernsteuerung

Die Tasten werden zyklisch von Ihrem Programm nach einer kurzen Wartepause (frei wählbar) abgefragt. Die LED-Ausgabe erfolgt danach. Bei jeder Tasterabfrage wird außerdem die Funktion `serial_transmit (int p)` aufgerufen. Der Parameter p hat den Wert -1, wenn die Maschine ausgeschaltet wird. Er hat den Wert 1, wenn die Maschine eingeschaltet wird. Er hat den Wert 0, wenn die Maschine unverändert weiterlaufen oder stillstehen soll. Dieser Parameter wird immer angewandt, wenn keine Taste oder beide Tasten bei der Abfrage betätigt sind.

Schreiben Sie dafür das Programm mit Kommentaren.

Hinweis: Diese Funktion `serial_transmit (int p)` wird in der Teilaufgabe b) programmiert. Dort wird auch die Funktion `serial_init()` für die Einstellung der UART programmiert. Für die Teilaufgabe a) sind beide Funktionsprototypen vordeklariert, damit Sie diese bereits benutzen können.

```
#define Port_D GPIO_PORTD_DATA_R
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"

void serial_init();
void serial_transmit(int p);

void main(void)
{
  int cw=0
  // Quarz Oszillator auf 16MHz
SYSCTL_RCC_R = (SYSCTL_RCC_R | 0x00000540) & ~0x000002B1);
cw++;
SYSCTL_RCC_R |= 0x00000002;
cw++
// Aktivierung des Clocks
SYSCTL_RCGC2_R |= 0x008; // Clock für Port D
cw++
GPIO_PORTD_DEN_R = 0x33; // Port D Pins 0,1,4,5
GPIO_PORTD_DIR_R = 0x30; // 4,5 als outputs
```

```
// Dauerschleife
while (1)
{
int i=0, p=0;
char c = 0x00;

for (i=0; i<1000; i++);   // Warteschleife
c = Port_D                // Port D einlesen

if (p==0 && (c & 0x03 == 0x01)) p=1;   // wenn die Maschine eingeschaltet wird
else if (p==0 && (c & 0x03 == 0x02)) p=-1;   // wenn die Maschine ausgeschaltet wird
else p=0;                 // ansonsten läuft sie weiter

serial_transmit(p);

if (p==1) PortD |= 0x20;   // LED "EIN"
else if (p==-1) PortD |= 0x10;   // LED "AUS"
else PortD &= ~(0x30);    // Beide LEDs Aus
}
```

15 v. 15

b) Die UART2 sendet bei einer Betätigung des Ein-Tasters das Kommando 'E' 'I' 'N' (drei Zeichen) und nach einer Betätigung des Aus-Tasters 'A' 'U' 'S' (drei Zeichen). Es wird im Format 8O1 mit 9600 Bit/s gesendet.
Schreiben Sie die bereits deklarierten Funktionen `serial_init()` und `serial_transmit(int p)`. Der Funktionsparameter p ist in der Teilaufgabe a) erklärt.

Verschlüsselung der Steuerkommandos:
Falls irgendjemand die Leitung abhört, wollen wir ihm ein kleines Rätsel aufgeben. Dafür werden die Zeichen leicht 'verschlüsselt'.
Das erste Zeichen einer Drei-Zeichen-Sendung, um den Wert einer Variablen X im ASCII-Code weitergerückt. Das zweite Zeichen um X+1 Werte, das dritte Zeichen um X+2 Werte. Die Variable X wird in der Funktion statisch vereinbart und ist anfänglich 1. Jede Drei-Zeichen-Sendung erhöht X um den Wert A. A wird zuvor als X modulo 13 errechnet. Jeder Funktionsaufruf mit dem Parameterwert p erhöht X um 1 ohne Sendung von Zeichen. Wenn X den Wert 253 oder mehr erreicht wird, dann wird X wieder auf 1 gesetzt.

+

Habe nicht genau gelesen

```
void serial_init();
{ int cw=0;
  SYSCTL_RCGC1_R |= 0x00000004;   // clock enable für UART2
  cw++;
  SYSCTL_RCGC2_R |= 0x040;        // clock enable für Port G
  cw++
  UART2_CTL_R &= ~0x0301          // UART 2 deaktivieren
  UART2_IBRD_R = 104;             // Bitrate von 9600
  UART2_FBRD_R = 11;              // einstellen
  UART2_LCRH_R = 0x64;            // 8o1 einstellen
  UART2_CTL_R |= 0x0301           // UART 2 aktivieren

  GPIO_PORTG_DEN_R |= 0x03;       // Port G Pins 0,1
  GPIO_PORTG_DIR_R |= 0x02;       // G(1) als output
  GPIO_PORTG_AFSEL_R |= 0x03;     // alternative Funktion
  GPIO_PORTG_PCTL_R |= 0x03;      // U2Rx
```

```
void serial_transmit(int p);      // verschlüsselung mit x ist nicht komplett
{                                 // würde jeweils nur um 0, 1 und 2 verschoben werden
  serial_init();                  nicht wenn x als statische Variable
static int x=0                    verwendet würde!
  while (!UART2_FR_R & 0x80))     // warten bis transmit FIFO leer ist
  if (p = 1)
  {
    UART2_DR_R = (0x45+x); // E senden
    while(!(UART2_FR_R & 0x80));
      UART2_DR_R = (0x49+x+1); // I senden
    while(!(UART2_FR_R & 0x80));
      UART2_DR_R = (0x4E+x+2); // N senden
  }
  else if (p=-1)
  {
    UART2_DR_R = 0x41;  +x
    while(!(UART2_FR_R & 0x80));
    UART2_DR_R = 0x55;  +x+1
    while(!(UART2_FR_R & 0x80));
    UART2_DR_R = 0x53;  +x+2
  }
}
```

fehlt modulo 13! s. S. 4

10 v. 15

25 v. 30

c) **ZUSATZAUFGABE für maximal 10 Zusatzpunkte:** Programmieren Sie eine möglichst einfache C-Funktion für die Verwendung in Software der Maschinensteuerung, welche die Verschlüsselung lt. Aufgabe b) auf der Empfangsseite 'entschlüsselt'. Die Funktion hat drei Parameter x,y,z, die drei nacheinander empfangenen Zeichen entsprechen. Die Funktion gibt -1 zurück, wenn in der Zeichenfolge x,y,z das verschlüsselte Steuerkommando 'AUS' erkannt wurde. Sie gibt 1 zurück, wenn die Zeichenfolge x,y,z dem verschlüsselten Steuerkommando 'EIN' entspricht. Der Wert 0 wird zurückgegeben, wenn in x,y,z kein Steuerkommando enthalten ist.

Hinweis 1: Die Funktion soll ohne Kenntnis der aktuellen Verschiebungswerte (siehe Variable X lt. Aufgabe b) der Verschlüsselung auskommen.
Hinweis 2: Berücksichtigen Sie, dass gelegentlich einige Übertragungen nicht erfolgreich sind (Zeichenverlust), aber bei längerem Tastendruck mehrfache Sendungen direkt hintereinander erfolgen.

```
int decode(char x, char y, char z)
{
int i=0, ergebnis=0;
for (i=0; i<255; i++)
{
 if (x-i==69 && y-(i+1)==73 && z-(i+2)==78) ergebnis=1;
 else if (x-i==65 && y-(i+1)==85 && z-(i+2)==83) ergebnis=-1;
}
return ergebnis;
}
```

Einfacher wäre direkt x-y und y-z vergleichen als usw....

8 v. 10 P. zusätzlich