V1.4 RMS

Department Informations- und Elektrotechnik
Labor für Digitale Informationstechnik
Praktikum Mikroprozessortechnik

Labor 4	Steuerung eines Microcontrollers mit dem PC über das UART-Protokoll	
Praktiumsgruppe-Labortisch:	**Abgabedatum:** Abgabe erfolgt spätestens eine Woche nach Durchführung	
Versuchstag:		**Versuchsteilnehmer:**
Dozent:		
Bewertung/Kommentar:		

Hinweise zur Ausarbeitung

In die schriftliche Ausarbeitung und das Protokoll gehört unter anderem:

- Eine kurze Beschreibung der Aufgabenstellung in eigenen Worten.
- Eine Beschreibung des Lösungsansatzes, des Algorithmus oder der Messmethode.
- Bei komplexen Programmteilen ein kommentierter Programmablaufplan oder ein Struktogramm.
- Listings der C-Programme im Anhang mit Autor, Version und Datum als Kommentar.
- Bei Messungen eine Beschreibung des Messaufbaus und der zu berücksichtigenden Effekte einschließlich relevanter Geräteeinstellungen.
- Ein Blockschaltbild und eine Übersicht der Schnittstellen und Ports
- Eine kurze Diskussion der Versuchs- und Messergebnisse, inklusive Abweichungen und Fehlern.
- Alle Übernahmen, Werte und Bilder sind zitiert und ein Quellenverzeichnis am Ende angelegt. In der Elektrotechnik benutzt man dazu den IEEE-Stil. Anleitungen finden Sie im Internet.

1. Ziele

Zunächst wird die Verbindung über die UART zum PC hergestellt. Es werden Zeichen übertragen und das Erkennen der Zeichen anhand des Signals mit dem Oszilloskop geübt.

Auf dem PC wird ein Terminalprogramm benutzt. Die Texteingaben im Terminalprogramm werden an den Mikrocontroller übertragen.

Anschließend werden vom Terminalprogramm aus einige Steuerbefehle an den Mikrocontroller geschickt. Diese werden dort ausgewertet, um einzelne LED‘s ein- oder auszuschalten. Die Abbildung 1 zeigt die wesentlichen beteiligten Komponenten.

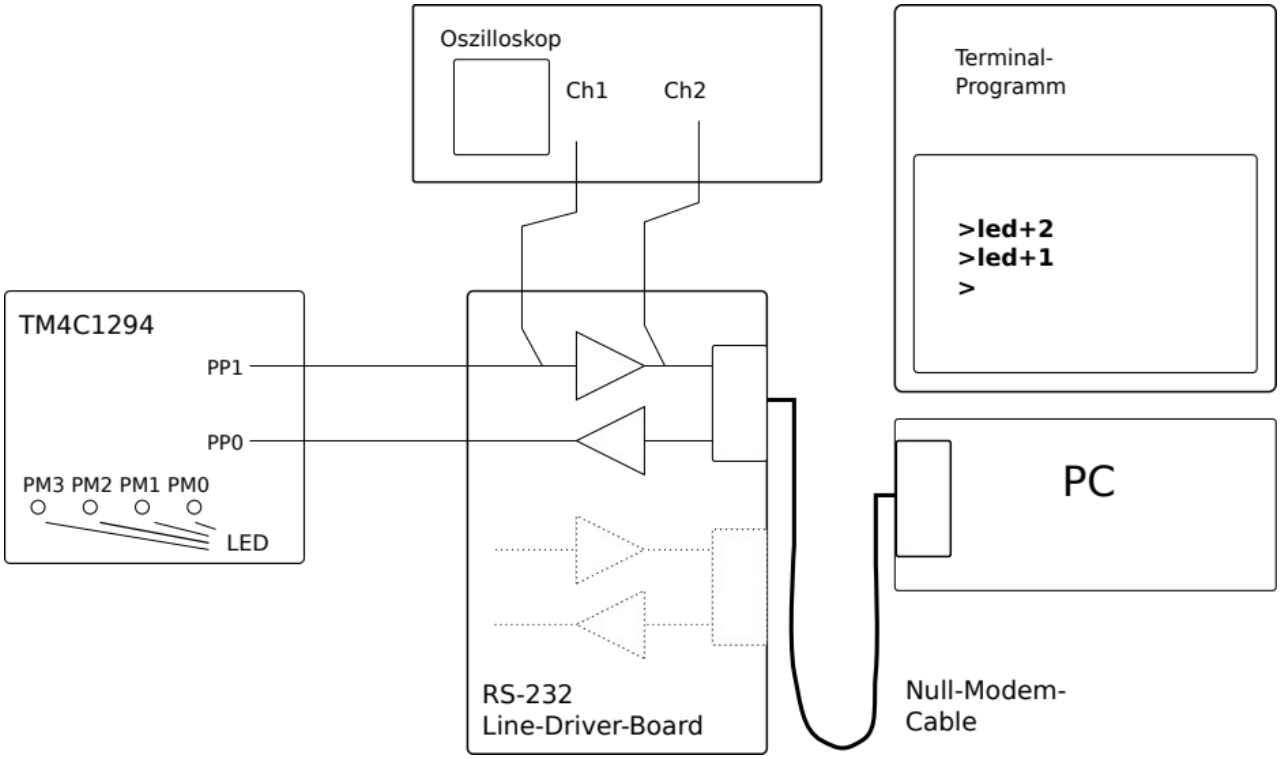

Abbildung 1: Darstellung des Versuchsaufbaus

2.1. Recherche als Vorbereitung

Lesen Sie die Registerkonfiguration der UART und die Beispiele in den Vorlesungsunterlagen durch. Suchen Sie im Datenblatt [1] die untengenannten Register heraus und notieren Sie sich die relevanten Bits und die erforderlichen Einträge und Informationen:
Zusätzlich sind die **Tabelle 10-2** auf der **Seite 744** und die **Registerbeschreibung** auf Seite **788** erforderlich.

Registernamen (SW / HW)	Bedeutung	Seite im Datenblatt
SYSCTL_RCGCUART_R / RCGCUART	UART Run Mode Clock Gating Control	388
UARTx_CTL_R / UARTCTL	UART Control	1188
UARTx_IBRD_R / UARTIBRD	UART Integer Baud-Rate Divisor	1184
UARTx_FBRD_R / UARTFBRD	UART Fractional Baud-Rate Divisor	1185
UARTx_LCRH_R / UARTLCRH	UART Line Control	1186
UARTx_FR_R / UARTFR	UART Flag	1180
UARTx_DR_R / UARTDR	UART Data	1175
GPIO_PORTx_AHB_AFSEL_R / GPIOAFSEL	GPIO Alternate Function Select	770
GPIO_PORTx_AHB_PCTL_R	GPIO Port Control	770

2.2. Terminalprogramm konfigurieren

Damit der PC die seriellen Schnittstellen ansprechen kann, muss ein Terminal-Programm verwendet werden.

Im Labor wird das bekannte Programm „Putty“ benutzt, das für die meisten Betriebssysteme verfügbar ist. Abbildung 2 zeigt die zunächst erforderlichen Einstellungen.

Sie können Einstellungen unter einem wählbaren Namen speichern und wiederherstellen.

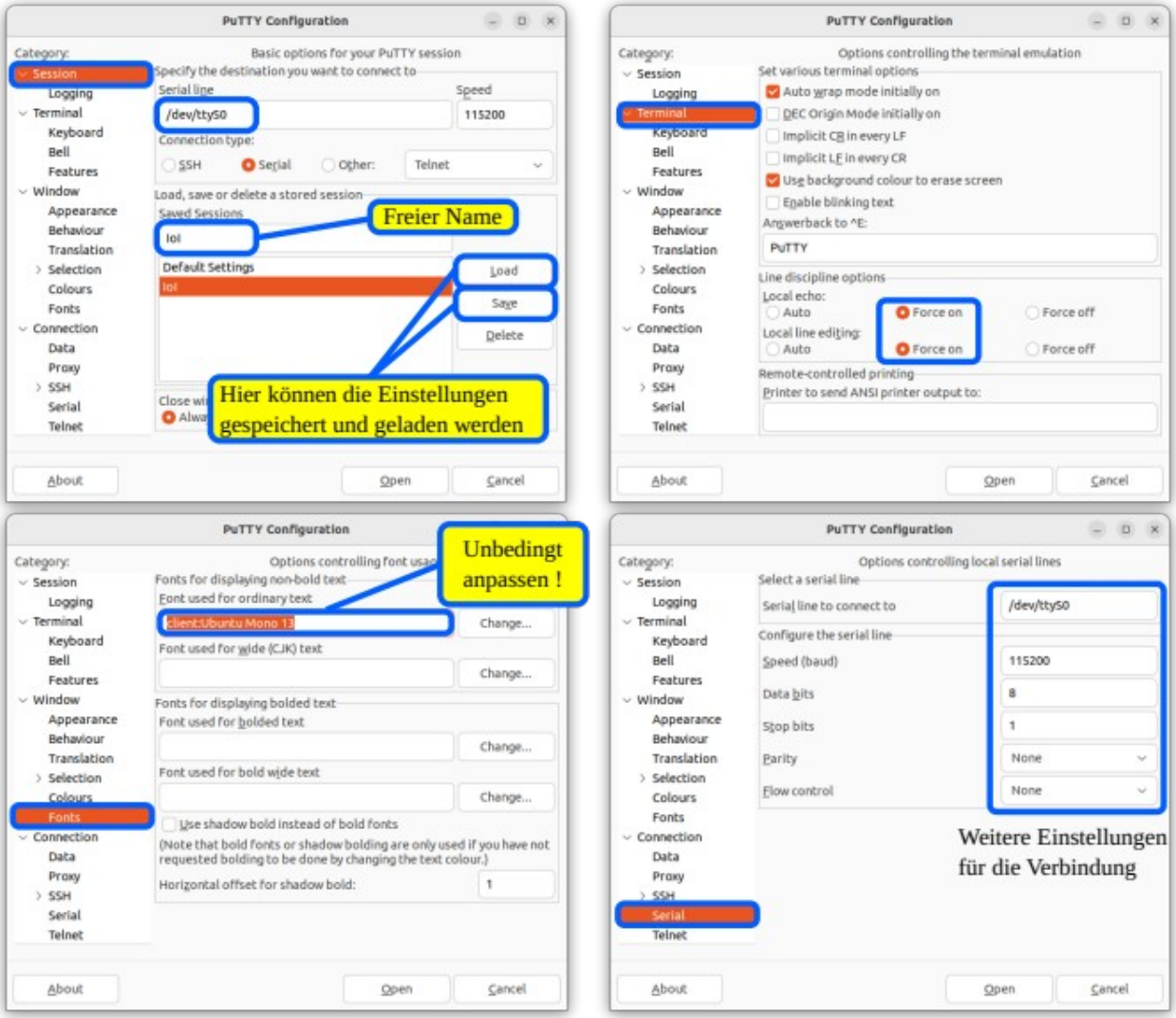

Abbildung 4: Einstellung zum Verhalten des Terminals unter Ubuntu 22.04 und neuer:

Links oben - Schnittstelle **/dev/ttyS0** wählen, eigene Einstellung unter einem Namen speichern

Rechts oben - weitere Einstellungen zum Verhalten des Terminals

Links unten - Einstellung eines passenden Zeichensatzes

Rechts unten - RS232-Format genau entsprechend der UART-Konfiguration definieren

2.3 Verbindung zwischen Mikrocontroller und PC testen

Im nächsten Schritt soll die Verbindung zwischen Microcontroller und PC getestet werden. Dazu soll ein Mikrocontroller-Programm fortlaufend ein einzelnes Zeichen aussenden.
Kopieren Sie ein bestehendes Projekt in der Entwicklungsumgebung Code Composer Studio und nennen Sie die Kopie des Projektes in einen eigenen Projektnamen um. Fügen Sie dort den Programmcode aus Listing 1 ein. Das Programm führt folgende Aktionen durch:

- Parameter des UART-Protokolls und Ausgang UART6 Tx auf Port P1 konfigurieren
- Es wird der RC-Oszillators auf den Chip mit 16 MHz genutzt.
- Das Zeichen ‚H' fortwährend an PC senden

Übersetzen Sie das Programm und laden Sie es auf den Mikrocontroller. Testen Sie, ob das Zeichen „H" im Terminalprogramm fortlaufend wiederholt erscheint.

```
//=========================================================
// Test program UART6 TX 8/N/1 @ 115200 bit via Port PP1
// LTL 17.5.2020 / mod. V0.2 - V0.4 RMS  1.6.2023
//=========================================================
#include "inc/tm4c1294ncpdt.h"          // Header of the controller type
#include <stdint.h>                     // Header w. types for the register ..

#define IDLETIME 1000                   // waiting time between transmissions
int wt = 0;                             // auxillary variable

void config_port(void){
    // initialize Port P
      SYSCTL_RCGCGPIO_R |= 0x02000;     // switch on clock for Port P
      wt++;                             // delay for stable clock
      // initialize Port P
     GPIO_PORTP_DEN_R |= 0x2;           // enable digital pin function for PP1
     GPIO_PORTP_DIR_R |= 0x2;           // set PP1 to output
     GPIO_PORTP_AFSEL_R |= 0x2;         // switch to alternate pin function PP1
     GPIO_PORTP_PCTL_R |= 0x10;         // select alternate pin function PP1->U6Tx
}

void config_uart(void){
    // initialize UART6
    SYSCTL_RCGCUART_R |= 0x40;          // switch on clock for UART6
    wt++;                               // delay for stable clock
    UART6_CTL_R &= ~0x01;               // disable UART6 for config
    // initialize bitrate of 115200 bit per second
    UART6_IBRD_R = 8;                   // set DIVINT of BRD floor(16 MHz/16*115200 bps)
    UART6_FBRD_R = 44;                  // set DIVFRAC of BRD remaining fraction divider
    UART6_LCRH_R = 0x00000060;          // serial format 8N1
    UART6_CTL_R |= 0x0101;              // UART transmit on and UART enable
}

void idle()      {                      // simple wait for idle state
   int i;
   for (i=IDLETIME;i>0;i--);            // count down loop for waiting
}

void main(void){
    config_port();  // configuration of  Port P
    config_uart();  // configuration of  UART6
    while(1){
        while((UART6_FR_R & 0x20) !=0); // till transmit FIFO not full
        UART6_DR_R = 'H';               // send the character 'H'
        idle();                         // idle time
    }
}
```

Listing 1: Programm zum Test der UART-Verbindung

3. Aufgabe 1 – Drei Zeichen in verschiedenen Formaten senden

Bereiten Sie die gleichzeitige, übersichtliche Aufzeichnung der Signale **vor und nach** dem Line-Driver mit **zwei Kanälen** des Oszilloskops vor. Senden Sie die folgenden Zeichen im jeweiligen Format. Speichern Sie die drei Oszilloskop-Bilder. Notieren Sie in der Bildunterschrift die jeweiligen Registereinstellungen des UART-Moduls. Prüfen Sie den Empfang am Terminal-Programm.

aufgabe 1a- 1c

a) ASCII-Zeichen X mit 9600 bps im Format 7E1 prüfen ob formt eine änderung mit sich bringt
b) ASCII-Zeichen a mit 38.400 bps im Format 8O1
c) Die Binärdaten 0x3B mit 4.800 bps im Format 7N2

4. Aufgabe 2 - „Code knacken“ ... Zeichen aus dem Signal decodieren

Üben Sie in der Gruppe das Erkennen von übetragenen Zeichen anhand von Oszilloskop-Bildern entsprechend Aufgabe 1.

Die Zeichentabelle für den ASCII–Code befindet sich im Anhang.

Ein Partner / einer Partner ist der Sender, der oder die anderen der Gruppe sind der oder die „Code-Knacker“. Es gelten diese Spielregeln.

1) Der oder die Code-Knacker so lange weg, bis der Sender mit Punkt 2 fertig ist. Sie wissen das gewählte Zeichen **nicht** und können auch das Terminalprogramm **nicht sehen** (Monitor wegdrehen oder kurz ausschalten).
2) Der Sender sendet ein **beliebiges** Zeichen in einer **beliebigen** Protokoll-Konfiguration mit dem jeweils angepassten Beispielprogramm aus Aufgabe 3.1.
3) Der oder die Code-Knacker analysieren **nur das Oszillogramm**. Sie versuchen, das Zeichen und das Protokoll zu erkennen.
4) Dann wechseln die Rollen … spielen Sie einige Runden, bis alle Partner oder Partnerinnen einige Zeichen aus den Signalen erfolgreich decodiert haben.

5. Aufgabe 3 - Empfang einer Zeichenkette vom PC

Kopieren Sie das vorheigegesamte Projekt nochmals und benennen Sie die Kopie um. Modifizieren Sie das Programm wie folgt:
Aktivieren Sie den Sender (Tx) **und** den Empfänger (Rx) des UART-Moduls und verbinden Sie beide mit dem Port P und den Pins PP1 bzw. PP0.

Anforderung 1: Als **Prompt** sollen die drei Zeichen mit jeder neuen Zeile im Terminal gesendet werden:

1. **Zeichen 0x0D = '\r'** (Carriage Return = „Wagen"-Rücklauf an Zeilenanfang)
2. **Zeichen 0x0A = '\n'** (Line Feed = Zeilenumbruch)
3. **Zeichen '>'** (ein Symbol als Eingabeaufforderung)

Anforderung 2: UART6 Rx ist von Ihrem Programm stetig abzufragen.

Anforderung 3: Empfangene Zeichen über UART6 Rx sind im Array vom Typ **char** zu speichern.

Anforderung 4: **MAXSIZE** ist als Größe des Arrays durch ein Makro zu definieren (10-50).

Anforderung 5: Beim Empfang des **MAXSIZE-1** Zeichens oder des Wertes **0x0D** ist der letzte Wert des String / Arrays mit 0x00 (\0) zu beschreiben.

Anforderung 6: Danach der String mit der Funktion **printf()** auf der Konsole auszugeben.

Anforderung 7: Der Ablauf soll nach Ausgabe mit einem neuen **Prompt** starten.

Hilfestellung bei Fehlern

Der Prompt '>' erscheint nicht im Terminal-Fenster?
Dann ist vielleicht das UART-Modul sendeseitig falsch konfiguriert.
Ist der Ausgang Tx aktiviert und mit Port Pin PP1 verbunden?
Landen Sie in der Dauerschleife des Interrupt-Handlers FaultISR() ? Dann haben Sie vielleicht den Takt eines der Peripheriemodule nicht aktiviert.
Printf() im Console-Fenster von CCS liefert keine Ausgabe?
Setzen Sie einen Breakpoint dort, wo das Array aus dem UART-Fifo befüllt wird. Mit jeder Tasteneingabe im Terminal-Fenster müsste ihr Programm dort stoppen und das Zeichen im Array erscheinen. Prüfen Sie das mit dem Debugger.
Das Array wird nicht befüllt?
Dann ist vielleicht das UART-Modul empfangsseitig falsch konfiguriert.
Ist der Eingang Rx aktiviert und mit Port Pin PP0 mit einer Leitung verbunden?
Lesen sie das UART6_FR_R Register korrekt? Bleibt Ihr Programm bei der Abfrage des Rx-FIFOs hängen? Prüfen Sie das mit einem schrittweisen Durchlauf im Debugger.
Das Array wird befüllt, aber nicht an der Console ausgegeben?
Übergeben Sie den String korrekt an die printf()-Funktion? Ist der String mit dem Zeichen \0 terminiert?

6. Aufgabe 4 - Dekodierung eines Steuerbefehls vom PC

Das Programm aus Aufgabe 3 soll erweitert werden, um die empfangene Zeichenkette als Steuerbefehle auf dem Mikrocontroller zu interpretieren.

Anforderung 1: Das gültige Eingabeformat hat folgende Syntax-Struktur:

led <+|-><0|1|2|3>

Die Befehle sind entsprechend der folgenden Tabelle zu interpretieren.

Kommando	Bedeutung
led+0	Einschalten erste LED am Port M
led+1	Einschalten zweite LED am Port M
led+2	Einschalten dritte LED am Port M
led+3	Einschalten vierte LED am Port M
led-0	Ausschalten erste LED am Port M
led-1	Ausschalten zweite LED am Port M
led-2	Ausschalten dritte LED am Port M
led-3	Ausschalten vierte LED am Port M

Anforderung 2: Die LED auf dem Board werden vom Programm auf dem Mikrocontroller nach der Dekodierung der empfangenen Zeichenkette einzeln ein- bzw. ausgeschaltet.

Anforderung 3: Sollte die Befehlssyntax verletzt werden, soll die Befehlsdekodierung abgebrochen und die Eingabe ignoriert werden.

Anforderung 4: Der Ablauf wiederholt sich und beginnt wieder mit einem neuen Prompt ‘>‘ auf einer neuen Zeile am Terminal.

Anforderung 5: Mit **printf()** ist die empfangene Zeichenfolge zusätzlich auszugeben.

7. Optionale Erweiterungen (Freiwillig zum selbstständigen Üben)

- Nutzen sie eine Integer-Variable für denkbare Syntaxfehler bei der Eingabe, sie ist bei fehlerfreier Syntax auf dem Wert 0. Bei Fehlern bekommt sie einen bestimmten Errorcode als Wert. (z.B. 1 = zu wenig Zeichen, 2 = Anfang-Folge „led “ nicht korrekt, 3 = Ziffer nicht korrekt o.ä.)
- Erweitern Sie die Ausgabe mit printf() auf der Konsole um den Errorcode.
- Erlauben Sie beliebig viele Leerzeichen zwischen „led“, „+“, „-“ und den Ziffern.
- Erlauben Sie das gleichzeitige Ein- und Ausschalten mehrerer LED durch kombinierte Befehle.

8. Anhang ASCII-Tabelle

Die folgende Tabelle stellt die Zeichen und Codierung des ASCII-Codes mit 7-Bit dar.
Die Zeile entspricht dem niedrigwertigen vier Bits, die Spalte den höherwertigen drei Bits.
Die ersten beiden Spalten sind Sonder- und Steuerzeichen.
Weitere Details liefert der Aufruf „man ascii“ im Terminal von Linux.

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

9. Quellen

[1] Texas Instruments, TivaTM TM4C1294NCPDT Microcontroller DATA SHEET, DS-TM4C1294NCPDT-15863.2743 / SPMS433B, June 18, 2014
http://www.ti.com/lit/gpn/tm4c1294ncpdt

[2] Texas Instruments, Tiva™ C Series TM4C1294 Connected LaunchPad Evaluation Kit EK-TM4C1294XL, User's Guide, SPMU365Cx, March 2014–Revised October 2016
https://www.ti.com/tool/EK-TM4C1294XL