```
/***********************************
 * Filename      : UE01_strgout.c   *
 * Created on    : Nov 17, 2018     *
 * Author        : Christian Zahner *
 ***********************************/


#include "UART0.h"
#include "support_common.h"  // include peripheral declarations and more;
#include "uart_support.h"    // universal asynchronous receiver transmitter,
                             // (d.h. die serielle Schnittstelle)
#include "terminal_wrapper.h"
#include <stdio.h>

#include "UE01_Strgout.h"


  void strgout(){

    char strg[]="Stringlystring\r\n";

    asm{
                    bra start
        start:

                    lea strg, a2     // strg in Adressregister a2
                                     // lea load effectiv adress
        loop:
                    move.b (a2)+,d3  // a2 byteweise in Datenregister d2
                                     // (a2)+ post incrementiert Addresse byteweise
                    beq loop_end     // Prüft ob Zeiche = 0 wenn ja dann Sprung
                                     // auf Marke loop_end
                    move.b  d3,-(sp) // Inhalt von d2 auf Stack
                                     // -(sp) pre decrement reservier von Speicher
                                     // für Char aus d2

                    jsr TERM_Write   // jsr Jump Subroutine springt in Unterprogramm
                    adda.l #1,sp     // Stack Speicherplatzfreigeben
                    bra loop         // Branch (Rücksprung) auf loop Marke


        loop_end:
                    //jsr TERM_Read    // Warten auf Tastendruck

    }
    TERM_WriteLn();
    TERM_WriteLn();
}

  void strgoutrev(){

    char strgr[]=" Stringlystring";
    printf(" String: %s\r\n",strgr);
    asm{
                    bra start
        start:
                    lea strgr, a2 // strg in Adressregister a2
                    clr.l d3      // Register d1 löschen

        loop_cnt:
                    add.l #1, d3  // increment counter in d3 um 1
                    tst.b (a2)+   // test auf a2 null und
                                  // (a2)+ Byteweise verschieben
                    bne loop_cnt  // branch not equal
```

```
                    sub.l #1, d3  // Anzahl Zeichen im String ohne 0-Byte
                    sub.l #1, a2  // Zeiger um 1 zurückgesetzt
                                  //jetzt auf \0 am String Ende

                    bra loop_end  // Sprung ans Schleifenende
                                  // => auch leere Strings werden richtig behandelt
                                  // => Schleife muss von (AnzahlZeichen - 1)
                                  // downto 0 laufen

        loop_out:
                    move.b -(a2),-(sp) // -(a2) da wir ja nicht
                                       // das Ende \0 Zeichen ausgeben wollen
                    jsr TERM_Write     // Zeichen ausgeben
                    adda.l #1, sp      // Stack bereinigen

        loop_end:

                    sub.l #1,d3
                    bge loop_out  //branch greater equal loop_cnt


    }
    TERM_WriteLn();
}

  void strgoutrevalt(){


        char stralt[]="Stringlystring";
        printf(" String: %s\r\n",stralt);
        asm{
                        bra start
            start:
                        lea stralt, a2        // stralt in Adressregister a2
                                              // für loop_cnt

                        clr.l d3              // Register d3 löschen

            loop_cnt:
                        add.l #1, d3          // increment counter in d3 um 1
                        tst.b (a2)+           // test auf a2 null und
                                              // (a2)+ Byteweise verschieben
                        bne loop_cnt          // branch not equal

                        sub.l #1, d3          // Anzahl Zeichen im String ohne 0-Byte
                        lea stralt, a2        // zurücksetzen der Adresse
            loop_out:
                        move.b (a2,d3),-(sp) // Char auf Stack mittles Adressversatz
                                              // d3 counter als byteweiser Adressversatz
                        jsr TERM_Write        // Jump to Subroutine
                        adda.l #1, sp         // Speicher auf Stack freigeben
                        subq.l #1, d3         // Counter erniedrigen


                        tst.b d3              // Test ob Counter 0


                        blt loop_end          // Falls Counter kleiner 0 Springe Ende

                        bra loop_out          //branch loop_out

            loop_end:                         //Marke zum beenden der Schleife
```

```
	}
  TERM_WriteLn();
}

void strgoutRevWithoutCount(){
	char strgr[]=" Stringlystring";
	printf(" String: %s\r\n",strgr);
	asm{

			lea strgr, a2 // strg in Adressregister a2
			move.l a2, a3 // kopie für Strgr
			clr.l d3      // Register d3 löschen

	    loop_cnt:

			tst.b (a2)+   // test auf a2 null und
			              // (a2)+ Byteweise verschieben
			bne loop_cnt  // branch not equal

			sub.l #1, a2  // Zeiger um 1 zurückgesetzt (auf \0 String Ende)

			bra loop_end  // Sprung ans Schleifenende
			              // => auch leere Strings werden richtig behandelt
			              // => Schleife muss von (AnzahlZeichen - 1)
			              //downto 0 laufen

	    loop_out:
			move.b -(a2),-(sp) // -(a2) da wir ja nicht
			                   // das Ende \0 Zeichen ausgeben wollen
			jsr TERM_Write     // Zeichen ausgeben
			adda.l #1, sp      // Stack bereinigen

	    loop_end:
			cmpa.l a2,a3
			bne loop_out  // branch not equal loop_out
			              // falls gleich beende

	}
	TERM_WriteLn();

}
```