```
/***********************************
 * Filename      : UE07A_bitrev.c   *
 * Created on    : Nov 29, 2018     *
 * Author        : Christian Zahner *
 ***********************************/

#pragma compact_abi

#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 "Intro.h"
#include <stdio.h>

#include "UE07A_Bitrev.h"

char zer[] = "0";
char on[]  = "1";
char sep[] = " ";

unsigned long bitrevUp(unsigned long quelle, unsigned long ziel){


  asm{
        bra         start


    BITREV2:
        adda        #-12, sp    // Platz für Register auf Stack machen
        movem.l     d1-d3,(sp)  // Register sichern

        move.l      d0,d1       // Datensichern

        clr.l       d0          // Datenregister mit 0 initialisieren
        clr.l       d3          // Beginn am "0ten" Bit mit Bitset bis 31. Bit

        moveq.l     #31, d2     // Anzahl ist von welchen Bit aus kopiert wird
                                // bzw. die Position von 31 bis 0
        bra         looptst

    looptst:
        tst.b       d2          // testen d2 0?
        blt         end
        btst.l      d2, d1      // Ist Bit an Position 1 oder 0 gesetzt
        bgt         bitset

        addi.l      #1, d3      // naechste Bit Stelle die gesetzt werden soll
        subi.l      #1, d2      // naechste Bit Position die getestet wird

        bra         looptst
    bitset:
        bset.l      d3, d0      // Bit setzten in Dataregister
        subi.l      #1, d2      // naechste Bit Position die geteste wird
        addi.l      #1, d3      // naechste Bit Stelle die gesetzt werden soll
        bra         looptst

    end:
        movem.l     (sp), d1-d3 // Register restaurieren
        adda        #12, sp     // Stack bereinigen

        rts
```

```
    start:
        move.l      quelle,d0   // Inhalte Quelle in Datenregister
        bsr         BITREV2
        move.l      d0,ziel
    }
}

void printbit(unsigned long zahl){

  asm{
      bra           start       // Branch Start erstmal zur Marke start

    PRNTBIT:
      adda      #-12,sp         // Platz für Register auf Stack machen
      movem.l   d3-d5,(sp)      // Register sichern

      clr.l     d5              // verschiedenen Datenregister 0 initialisieren
      clr.l     d3
      clr.l     d4

      move.l    zahl, d4        // Bitmuster in Speicherschreiben
      move.l    #32, d3         // Start vom 32. Bit
      move.l    #4,  d5         // Counter für Leerzeichen

      bra       outptloop       // Springe zur Marke out

    outptloop:
      subi.l    #1, d3          // wieder an Stelle 31 beginnend
      btst.l    d3, d4          // Bittesten ist es 1 oder 0
      bgt       one             // sollte das Bit 1 sein wird
                                // Zerobit in CCR nicht gesetzt
    zero:
      pea       zer             // Adresse char zero auf Stack pushen
      jsr       TERM_WriteString
      adda      #4, SP          // Stack bereinigen

      tst.b     d3              // d3 0?
      beq       end             // wenn ja Springe zur Marke end

      subi.l    #1, d5          // Leerzeichen Counter erniedirgen
      tst.b     d5              // d5 0?
      beq       plchld          // Wenn ja wurden 4 Zeichen ausgegeben
                                // Dann wird ein Leerzeichen gedruckt

      bra       outptloop       // Schleife wiederholen, nächstes Zeichen

    one:
      pea       on              // Adresse char one auf Stack
      jsr       TERM_WriteString
      adda      #4, SP          // Stack bereinigen

      tst.b     d3              // d3 0?
      beq       end             // wenn ja Springe zur Marke end

      subi.l    #1, d5          // Leerzeichen Counter erniedirgen
      tst.b     d5              // d5 0?
      beq       plchld          // Wenn ja wurden 4 Zeichen ausgegeben
                                // Dann wird ein Leerzeichen gedruckt

      bra       outptloop

    plchld:
      pea       sep             // Adresse des Seperators auf Stack pushen
      jsr       TERM_WriteString
```

```
      adda      #4, SP          // Wie immer Stack bereinigen

      move.l    #4, d5          // Counter für Leerzeichen wieder auf 4 setzen
      bra       outptloop       // Schleife wiederholen, nächsten 4 Zeichen

    end:
      movem.l   (sp),d3-d5      // Register restaurieren
      adda      #12,sp          // Stack bereinigen

      rts

    start:

     bsr        PRNTBIT         // Branch Subroutine PRNTBIT
     jsr        TERM_WriteLn    // neue Zeile

    }
}
```