```
/*
 * caolibrary.asm
 * This is a collection of assembly routines for the CAO1 course at
 * Via University College.
 * Version 2
 */

; ------- Port constants (for 16 MHz & 115.2 Kbaud) -------
; UBRR0 = (F_CPU/(16*BAUD))-1 = (16e6/(8*115200))-1 = 16.36 ~ 16
#define UBRR0_VALUE 16

/*************************
 *     Name: cao_usart0_init
 *  This routine initializes the USART0 (Universal Synchronous
 * and Asynchronous serial Receiver and Transmitter) serial port.
 *     Input  : None
 *     Output : None
 *************************/
cao_usart0_init:
    push r31

    ; High speed (U2X0=1), 115200 baud @16MHz
    ldi r31, high(UBRR0_VALUE)
    sts UBRR0H, r31
    ldi r31, low(UBRR0_VALUE)
    sts UBRR0L, r31

    ; UCSR0A: U2X0=1 (fast), other stuff default
    ldi r31, (1 << U2X0)
    sts UCSR0A, r31

    ; UCSR0B: Enable TX (TXEN0=1, RX is turned off)
    ldi r31, (1<<TXEN0)
    sts UCSR0B, r31

    ; UCSR0C: Asynchronous, No parity, 1 stopbit, 8-bit data (UCSZ01:0=11)
    ldi r31, (1<<UCSZ01) | (1<<UCSZ00)
    sts UCSR0C, r31

    pop r31
    ret

#undef UBRR0_VALUE

; Allocate 10 bytes buffer in SRAM to generate output message
#define BUFSIZE_PRINT_R31 10
.dseg
_buffer_pr31: .BYTE BUFSIZE_PRINT_R31

.cseg

/*************************
 *     Name: cao_print_r31
 *  This routine converts the number in r31 to an ascii string and
 *  prints the resulton the serial line (USART0).
 *     Input  : The number to be printed must be in r31
 *     Output : None
 *************************/
```

```
cao_print_r31:
    push r26  ; Save registers
    push r27  ; r26:r27 = register X
    push r29  ; Work register
    push r30    ; Work register
    push r31    ; Incoming value to print

    ldi XH, HIGH(_buffer_pr31) ; Make X register point to start of buffer
    ldi XL, LOW(_buffer_pr31)
    ldi r29,0                     ; We use r29 as a flag (boolean)

    cpi r31,100                          ; Skip i r31 < 100
    brlo _check10s_pr31
    ldi r30,'0'                          ; Set r30 = '0' = 0x30
_loop100s_pr31:                          ; At this point we know r31 >= 100
    inc r30
    subi r31,100                  ; Subtract 100 from r31
    cpi r31, 100
    brsh _loop100s_pr31           ; Try again if r31 is still >= 100
    st X+,r30                     ; Write '1' or '2' to buffer
    ldi r29,1                     ; Force output of all following digits

_check10s_pr31:
    ldi r30,'0'                          ; Set r30 = '0' = 0x30
    cpi r31,10                           ; Skip if r31 < 10
    brlo _no10s_pr31
_loop10s_pr31:                           ; At this point we know r31 >= 10
    inc r30
    subi r31,10                          ; Subtract 10 from r31
    cpi r31,10
    brsh _loop10s_pr31            ; Try again if r31 is still >= 10
    ldi r29,1                     ; Force output
_no10s_pr31:
    tst r29                              ; Check if we should output a placeholder '0'
    breq _write1s_pr31
    st X+,r30                     ; Write '0'..'9' to buffer

_write1s_pr31:
    ldi r30,'0'
    add r30,r31                          ; At this point we know r31 < 10
    st X+,r30                     ; Write '0'..'9' to buffer

    ldi r30,'\r'                  ; Write CR (carriage return) to buffer
    st X+,r30
    ldi r30,'\n'                  ; Write LF (newline) to buffer
    st X+,r30
    ldi r30,0                     ; Write 0 (string terminator
    st X+,r30

    ldi XH, HIGH(_buffer_pr31)    ; Make X register point to start of buffer again
    ldi XL, LOW(_buffer_pr31)
    call cao_print_data_x   ; Print the number

_done_pr31:
    pop r31          ; Restore registers
    pop r30
    pop r29
    pop r27
    pop r26
    ret                     ; Return to caller

#undef BUFSIZE_PRINT_R31

/*************************
 *     Name: cao_print_data_x
 *  This routine prints a (0-terminated) string from data space
 *     to the serial line (USART0)
 *     Input  : The x register [r27:r26] must point to a 0-terminated string in data
space
 *     Output : None
 *************************/
cao_print_data_x:
    push r26                ; x low
    push r27                ; x high
    push r29
    push r30

_send_loop_pdx:
    ld r30, X+              ; r30 = next character from data memory
    tst r30                 ; 0 indicates string termination
    breq _done_pdx

_wait_udre_pdx:
    lds  r29, UCSR0A
    sbrs r29, UDRE0         ; Wait until transmit buffer (UDR0) is empty
    rjmp _wait_udre_pdx
    sts  UDR0, r30          ; Send character
    rjmp _send_loop_pdx

_done_pdx:
    pop r30          ; Restore registers
    pop r29
    pop r27
    pop r26
    ret                     ; Return to caller

 /*************************
 *     Name: cao_print_code_z
 *  This routine prints a (0-terminated) string from code space
 *     to the serial line (USART0)
 *     Input  : The z register [r31:r30] must point to a 0-terminated string in code
space
 *     Output : None
 *  NOTE      : Because code memory uses word addressing, 1 word = 16 bits/2 bytes,
 *               you must load the z register like this (msg is the address of the
 *            message to print):
 *                     ldi ZH, HIGH(msg*2) ; <--- Notice the *2 !
 *                     ldi ZL, LOW(msg*2)  ; <---
 *************************/
cao_print_code_z:
    push r28
    push r29
    push r30                ; z low
    push r31                ; z high

_send_loop_pcz:
    lpm r28, Z+             ; r28 = next character from code/flash memory
    tst r28                 ; 0 indicates string termination
    breq _done_pcz

_wait_udre_pcz:
    lds  r29, UCSR0A
    sbrs r29, UDRE0               ; Wait until transmit buffer (UDR0) is empty
    rjmp _wait_udre_pcz
    sts  UDR0, r28          ; Send character
    rjmp _send_loop_pcz

_done_pcz:
    pop r31          ; Restore registers
    pop r30
    pop r29
    pop r28
    ret                     ; Return to caller

; The delay loop below takes 6 + N x (131074) cycles ~ 8.2 ms,
; where N = the value loaded into register r16 when the routine
; is called
cao_delay_r16:
    push r16
    push r17
    push r18

_loop1_dr16:
    ldi r17, 255
_loop2_dr16:
        ldi r18, 255
_loop3_dr16:
            dec r18
            brne _loop3_dr16
        dec r17
        brne _loop2_dr16
    dec r16
    brne _loop1_dr16

    pop r18
    pop r17
    pop r16
    ret
```