// PROJECT :SerialCommString (SerialCommOneString.S)

// PURPOSE :Assembly level UART transmission of a null-terminated string

// COURSE :ICS4U-E

// AUTHOR :C. D'Arcy

// DATE :2025 03 25

// MCU :328P

// STATUS :Working

// REFERENCE:http://darcy.rsgc.on.ca/ACES/Datasheets/ATMEGA328P.pdf

#include "avr/io.h"

#define BAUDRATE 9600 //Desired baud rate

//#define BAUD\_PRESCALER (((F\_CPU / (BAUDRATE \* 16))) - 1)

#define BAUD\_PRESCALER 103 //precomputed from formula above

#define data 'A' //ASCII value translates to 'A'

string:

 .asciz "RSGC ACES" //null-terminated string

.global setup

setup:

 rcall Init\_UART //configure the UART for transmission

 ldi ZL, lo8(string) //position Z as pointer to string in program memory

 ldi ZH, hi8(string)

nextChar:

 lpm r24,Z+ //load r24 register with the next character

 cpi r24,0 //are we at the end of the string?

 breq bye //if so, we're done

 rcall TXCHAR //transmit the contents of r24

 rcall delay1ms //pause...

 rjmp nextChar //keep going...

bye:

 ret //end of setup

// Precondition: Character data (1 byte) to be transmitted is in R24

// Postcondition: Contents of R24 are transmitted via the UART

TXCHAR:

 sts UDR0, r24 //load UDR0 with r24 to send the data

 ret

delay1ms:

; Assembly code auto-generated

; by utility from Bret Mulvey

; Delay 16 000 cycles

; 1ms at 16.0 MHz

 ldi r18, 21

 ldi r19, 199

L1: dec r19

 brne L1

 dec r18

 brne L1

 ret

Init\_UART:

 ldi r16, BAUD\_PRESCALER >> 8 //Prepare to do the high bit first

//ldi r16, hi8(BAUD\_PRESCALER) //this is an alternative to above

 sts UBRR0H, r16 //UBRR0H/L registers control the baud

 ldi r16, BAUD\_PRESCALER //Now prepare to do the low bit

//ldi r16, lo8(BAUD\_PRESCALER) //this is an alternative to above

 sts UBRR0L, r16 //UBRR0H/L registers control the baud

 ldi r16, 1 << TXEN0 //enable transmission

//ldi r17, 1 << RXEN0 //enable receiving

 sts UCSR0B, r16 //Set to transmit only in this case

 ldi r16, 3 << UCSZ00 //8 bit data size (enough for char)

 sts UCSR0C, r16 //Data is 8 bit data size

 ret

.global loop

loop:

 rjmp loop //Avoid returning to the C driver