;PROJECT :ShiftoutDualLowLevel

;PURPOSE :Code-Level Series: C (High) > Register (Mid) > Assembly (Low)

;COURSE :ICS4U-E

;AUTHOR :C. D'Arcy

;DATE :2025 02 16

;MCU :\* (we used the ATtiny85)

;STATUS :Working

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

;REFERENCE:http://darcy.rsgc.on.ca/ACES/TEI4M/Fritzing/ShiftoutDualMBV4.png

#include <avr/io.h>

.global setup ;identify externally accessible function

.global loop ;identify externally accessible function

;--------------C-Style Defines--------------

#define UTIL r16 // name registers if they have a purpose

;------------------Equates------------------

.equ DDR, DDRB-0x20 ;good practice to name all resources

.equ PORT, PORTB-0x20 ;improves readability

.equ CLOCK, PB2 ;

.equ DATA, PB1 ;

.equ LATCH, PB0 ;

.equ ORDER, 1 ;0:LSBFIRST, 1:MSBFIRST

.equ ITERS, 8 ;for the loops...

.text ;place data in Program Memory

value:

.word **0xFADE** ;little Endian order

alt:

.byte 0xFF,0x0F ;alternative

setup:

; ldi UTIL, 1<<CLOCK | 1<<DATA | 1<<LATCH ;

; out DDR, r16 ;

sbi DDR,CLOCK ;

sbi DDR,LATCH ;

sbi DDR,DATA ;

ret

loop:

ldi XL,lo8(value) ;pointer to start of PROGMEM data to be shifted

ldi XH,hi8(value) ;

ldi YL,lo8(alt) ;pointer to end of PROGMEM data to be shifted

ldi YH,hi8(alt) ;

movw Z,X ;position Z to the start of the data

again:

lpm r24,Z+ ;acquire the first data byte

ldi UTIL,ORDER ;consider the bit order...

sbrc UTIL,0 ;...if clear (LSBFIRST), we're good to go

rcall reverseBits ;...if set (MSBFIRST), reverse the bits

cbi PORT,LATCH ;PORTB &= ~LATCH;

rcall shiftOut ;shiftOut(DATA, CLOCK, LSBFIRST, value);

sbi PORT,LATCH ;PORTB |= LATCH;

cp ZL,YL ;have we reached the end of data to be shifted?

brne again ;if not, repeat shiftout for the next byte

rjmp .-2 ;hold and confirm

ret

;---------------------------------------------------------------------------

;**See Design By Contract:** https://en.wikipedia.org/wiki/Design\_by\_contract

;Precondition: CLOCK, DATA and ITERS defined. Data byte to be shifted is in r24

;Postcondition: Data byte is shifted out LSBFIRST

;Side Effects: r16, r24 changed

;---------------------------------------------------------------------------

shiftOut:

ldi r16,ITERS ;number of bit shifts

nextBit: ;start of loop...

cbi PORT,CLOCK ;PORTB &= ~CLOCK;

cbi PORT,DATA ;assume the LSB is clear so clear the DATA pin

sbrc r24,0 ;if LSB is clear, nothing to do

sbi PORT,DATA ;LSB is set so set the DATA pin

sbi PORT,CLOCK ;PORTB |= CLOCK;

lsr r24 ;shift data right one bit

dec r16 ;one less iteration

brne nextBit ;repeat if not finished

ret

;---------------------------------------------------------------------------

;**See Design By Contract:** https://en.wikipedia.org/wiki/Design\_by\_contract

;Precondition: ITERS defined. Data byte to be reversed (source) is in r24

;Postcondition: Contents of r24 reversed

;Side Effects: r16, r24, r25 changed

;---------------------------------------------------------------------------

reverseBits:

ldi r16,ITERS ;number of bit shifts (8)

clr r25 ;target <- 0

next: ;start of loop...

lsl r24 ;shift the byte left, C <- MSB Flag, LSB <- 0

ror r25 ;rotate target, MSB <- C Flag, C <- LSB

dec r16 ;one less iteration

brne next ;if not done, repeat

mov r24,r25 ;source <- target

ret

