/\* MorlandShiftBar.c

\* Created: 1/13/2019 3:18:20 PM

\* Author: Chris Darcy

\* Notes: Ensure ShiftBar device NOT in MOSI,MISO,SCK pins

\* (interferes with ICE ISP programming)

\*/

#include <avr/io.h>

//#define F\_CPU 16000000 //N/A

//#include <util/delay.h> //N/A

uint8\_t clock=1<<PD7; //ShiftBar in PORTD avoids // conflict with ISP pins

uint8\_t latch=1<<PD6; //

uint8\_t data= 1<<PD5; //

uint8\_t ground = 1<<PD4; //

uint8\_t power = 1<<PD3; //Not the best to have port pin supply power…but…

#define LSBFIRST 0 //order of bit presentation

#define MSBFIRST 1 //

#define ddr DDRD //place on PORTD to avoid ISP interference

#define port PORTD

#define BLINK\_DELAY\_MS 1000 //Not currently implemented

#define n 0xA5 //alternate binary constant syntax: 0b10100101

#define params 0xF8 //AVR GNU C requires initializers to be constants:

//does not like OR'd syntax as in:

// clock | latch | ground | power :(

//declare function template as per convention

void shiftOut(uint8\_t d, uint8\_t c, uint8\_t dir, uint8\_t value);

int main(void) {

 ddr = params; //set data direction of port bits

 port |= power; //supply power from a port pin (not the best)

 shiftOut(data, clock, LSBFIRST, n);

 while(1); //wait indefinitely

 //\_delay\_ms(BLINK\_DELAY\_MS); //included for syntax reference purposes

}

void shiftOut(uint8\_t d, uint8\_t c, uint8\_t dir, uint8\_t value){

 port &= ~latch; //pull latch low (*syntax only works if defined*

*// as a variable*)

 uint8\_t mask; //must handle either bit order...

 for (uint8\_t i=0; i<8; i++){ //structure to shift in (serially) a full byte

 port &= ~clock; //pull bit clock low

 if (dir) //is it MSBFIRST?

 mask = value & (0x80>>i); //if so, mask from left to right..

 else

 mask = value & (1<<i); //else, from right to left

 if (mask) //place the correct bit on the data pin

 port |= data; //

 else //

 port &= ~data; //

 port |= clock; //clock in this bit

 }

 port |= latch; //set latch HIGH to present registers on

// the output pins

}