/\*

\* MorlandShiftBar.c

\*

\* Created: 4/11/2020 10:56:20 AM

\* Author: Chris Darcy

\* Notes: Upload with SPP. MB3 in UNO PORTD to avoid ISP pins

\*/

#include <avr/io.h>

*uint8\_t* ground = 1<<PD7;

*uint8\_t* *clock* = 1<<PD6;

*uint8\_t* power = 1<<PD5;

*uint8\_t* data = 1<<PD4;

*uint8\_t* latch = 1<<PD3;

*uint8\_t* enable = 1<<PD2;

#define LSBFIRST 0 //order of presentation

#define MSBFIRST 1 //

#define ddr DDRD //place on PORTD, avoids ISP

#define port PORTD

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

#define n 0xA5 //alternate binary constant

//AVR GNU C requires initializers to be constants: does not like OR'd syntax:clock | latch | ground | power :(

#define params 0xFC

//declare function template as 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 ☹

 shiftOut(data, *clock*, LSBFIRST, n);

 while(1); //wait indefinitely

}

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

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

 port &= ~latch; //first, pull latch low

//structure to shift in (serially) a full byte

 for (*uint8\_t* i=0; i<8; i++){

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

 if (dir) //is it MSBFIRST?

 mask = value & (0x80>>i); //then mask from MSB to LSB..

 else

 mask = value & (1<<i); //else, from LSB to MSB…

 if (mask) //mount the correct data

 port |= data;

 else

 port &= ~data;

 port |= *clock*; //clock in this bit

 }

 //lastly, set latch HIGH to present registers on the output

port |= latch;

}