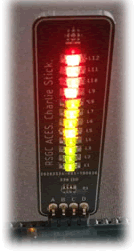
Efficient *code* design goes hand-in-hand with optimum *data* design. In this review, we examine how J. Dolgin’s efficient **Arduino C** design can be reworked into a similar **GPIO** **Register-Level** strategy.

Watch Microchip’s 1:55 minute video summary of the 8-bit AVR Port/Pin manipulation:

<https://www.youtube.com/watch?v=bDPdrWS-YUc>

### 1. Arduino C Version



// PROJECT  :CharlieStick

// PURPOSE  :Create an animation for the ACES CharlieStick

// COURSE   :ICS4U

// AUTHOR   :J. Dolgin

// DATE     :2019 10 25

// MCU      :328P

// STATUS   :Working

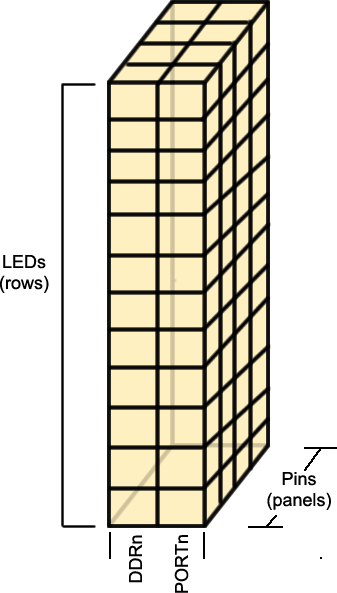
#define PinCon    0     //Column 0: I/O Direction (INPUT/OUTPUT)

#define PinState  1     //Column 1: Signal State (HIGH/LOW)

#define LED\_COUNT 12    //Number of LEDs on the CharlieStick

#define numPin 4     //How many pins are used for the stick

uint8\_t stick[] = {10, 11, 12, 13}; //What pins are being used



uint8\_t configuration[LED\_COUNT][2][numPin] = {

 { { INPUT, INPUT, OUTPUT, OUTPUT }, { LOW, LOW, LOW, HIGH } }, // 1

 { { INPUT, INPUT, OUTPUT, OUTPUT }, { LOW, LOW, HIGH, LOW } }, // 2

 { { INPUT, OUTPUT, OUTPUT, INPUT }, { LOW, LOW, HIGH, LOW } }, // 3

 { { INPUT, OUTPUT, OUTPUT, INPUT }, { LOW, HIGH, LOW, LOW } }, // 4

 { { OUTPUT, OUTPUT, INPUT, INPUT }, { LOW, HIGH, LOW, LOW } }, // 5

 { { OUTPUT, OUTPUT, INPUT, INPUT }, { HIGH, LOW, LOW, LOW } }, // 6

 { { INPUT, OUTPUT, INPUT, OUTPUT }, { LOW, LOW, LOW, HIGH } }, // 7

 { { INPUT, OUTPUT, INPUT, OUTPUT }, { LOW, HIGH, LOW, LOW } }, // 8

 { { OUTPUT, INPUT, OUTPUT, INPUT }, { LOW, LOW, HIGH, LOW } }, // 9

 { { OUTPUT, INPUT, OUTPUT, INPUT }, { HIGH, LOW, LOW, LOW } }, // 10

 { { OUTPUT, INPUT, INPUT, OUTPUT }, { LOW, LOW, LOW, HIGH } }, // 11

 { { OUTPUT, INPUT, INPUT, OUTPUT }, { HIGH, LOW, LOW, LOW } }  // 12

};

void setup() {}

void loop() {

 for (uint8\_t l = 0; l < LED\_COUNT-1; l++) {   //Runs as many times as there are LEDs

   for (uint8\_t i = 0; i < numPin; i++) {      //Runs as many times as the pins on the stick

      digitalWrite(stick[i], configuration[l][PinState][i]);  //Sets the pin state first!

     pinMode(stick[i], configuration[l][PinCon][i]);         //Sets the pin configuration

   }

   delay(100);

 }

 for (uint8\_t l = LED\_COUNT-1; l > 0; l--) {                 //Counts back down

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

      digitalWrite(stick[i], configuration[l][PinState][i]);  //Sets the pin state first!

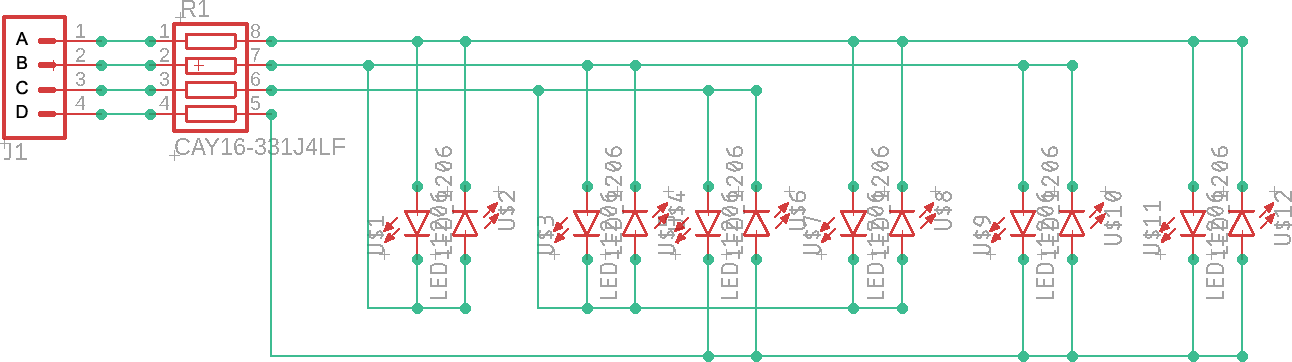
     pinMode(stick[i], configuration[l][PinCon][i]);         //Sets the pin configuration

   }

   delay(100);

 }

}



### Tasks

Reference the CharlieStick EAGLE schematic above for the tasks you are about to undertake. With your CharlieStick in the same position as the previous sketch, supply logic level (0 or 1) for EACH of the empty register bits to accomplish the requested task.

a) Turn on only **LED1**.



b) Turn on only **LED4**.



c) Turn on only **LED12**.



### 2. Register-Level Version

Copy the code shell below and let’s use it to optimize JD’s high-level Arduino C code from page 1.

// PROJECT  :SimonStickRegisterLevelUNO

// PURPOSE  :Register-level demonstration of ACES’ Charlieplexed SimonStick

// DEVICE   :UNO + SimonStick

// AUTHOR   :J. Dolgin, C. D'Arcy

// DATE     :2020 01 30

// MCU      :328P

// COURSE   :ICS4U

// STATUS   :Not Working

#define PORT  PORT?

#define DDR   DDR?

#define pinA 1<<?

#define pinB ?

#define pinC ?

#define pinD ?

uint8\_t pinMask = pinA | pinB | pinC | pinD;    //compose a convenient mask

struct LED { //TIGHT binding of related GPIO registers

 uint8\_t port;

 uint8\_t ddr;

};

LED stick [] = {

 {pinA, pinA | pinB}, //this should get only LED1 to turn on

?,?,…,?

 };

uint8\_t ledCount = sizeof(stick) / sizeof(LED);

void setup() {}

void loop() {

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

?

 }

 for (uint8\_t i = ledCount-1; i > 0; i--) {

?

 }

}

### 3. Your Turn! DDP: CharlieStick with TMP36

Capitalize on your understanding of the previous discussion to have your CharlieStick respond (in bar mode) to the user pinching the TMP36 sensor. (*have it range from room temperature to the maximum healthy human*)

[http://darcy.rsgc.on.ca/ACES/TEI4M/DolginDevPlatform/index.html#SimonStick](http://darcy.rsgc.on.ca/ACES/TEI4M/DolginDevPlatform/index.html%23SimonStick)

You are asked to complete this for homework and bring your prototype to our next class for a demonstration.