```
/*
 Code for the 3-Oscillator Arduino Synth
  built into a toy keyboard

 Utilizing the three internal timer modules of the ATMega328P
 and a Microchip MCP23017 GPIO extender to allow for input from
 all the keyboard keys

 pins 15~17 of MCP2017 to GND -> I2C bus address is 0x20

*/

#include "Wire.h"

byte pressedPort = 0x00; //00-> none, 01-> expander B, 02-> expander
A, 04-> D, 08-> B
byte noteMap = 0x00; //bitmap for current note in its respective port

byte gate=0; //this will be 0xFF if a button is pressed, and 0x00 if
not

uint8_t noteIndexOffset = 0;
uint8_t note1IndexOffset = 0;
uint8_t note2IndexOffset = 7;

//all the notes we care about; it's an array of bytes for {OCR0A,
TCCR0B}
// there are enough notes here for us to include an offset for Osc2
// the 3 LSBs of TCCR0B will be zeroed out, and then ORed with the
second bit value of the notes row
byte notes[47][2] = {
{79,0x05}, //G2
{74,0x05}, //G#2
{70,0x05}, //A2
{66,0x05}, //A#2
{62,0x05}, //B2
{59,0x05}, //C3
{223,0x04}, //C#3
{211,0x04}, //D3
{200,0x04}, //D#3
{189,0x04}, //E3
{178,0x04}, //F3
{167,0x04}, //F#3
{158,0x04}, //G3
{150,0x04}, //G#3
{140,0x04}, //A3
{133,0x04}, //A#3
{125,0x04}, //B3
{118,0x04}, //C4
{111,0x04}, //C#4
```

```
{105,0x04}, //D4
{99,0x04}, //D#4
{94,0x04}, //E4
{88,0x04}, //F4
{83,0x04}, //F#4
{79,0x04}, //G4
{74,0x04}, //G#4
{70,0x04}, //A4
{66,0x04}, //A#4
{62,0x04}, //B4
{237,0x03}, //C5
{224,0x03}, //C#5
{211,0x03}, //D5
{200,0x03}, //D#5
{189,0x03}, //E5
{178,0x03}, //F5
{168,0x03}, //F#5
{158,0x03}, //G5
{149,0x03}, //G#5
{141,0x03}, //A5
{133,0x03}, //A#5
{125,0x03}, //B5
{118,0x03}, //C6
{112,0x03}, //C#6
{105,0x03}, //D6
{100,0x03}, //D#6
{94,0x03}, //E6
{89,0x03} //F6
};

uint16_t notes16bit[47][2] = {
{158,0x05}, //G1
{150,0x05}, //G#1
{141,0x05}, //A1
{133,0x05}, //A#1
{125,0x05}, //B1
{119,0x05}, //C2
{445,0x04}, //C#2
{423,0x04}, //D2
{399,0x04}, //D#2
{378,0x04}, //E2
{356,0x04}, //F2
{335,0x04}, //F#2
{316,0x04}, //G2
{301,0x04}, //G#2
{281,0x04}, //A2
{267,0x04}, //A#2
{252,0x04}, //B2
{237,0x04}, //C3
{223,0x04}, //C#3
```

```
{211,0x04}, //D3
{199,0x04}, //D#3
{189,0x04}, //E3
{177,0x04}, //F3
{167,0x04}, //F#3
{157,0x04}, //G3
{149,0x04}, //G#3
{139,0x04}, //A3
{133,0x04}, //A#3
{123,0x04}, //B3
{475,0x03}, //C4
{447,0x03}, //C#4
{423,0x03}, //D4
{399,0x03}, //D#4
{379,0x03}, //E4
{355,0x03}, //F4
{337,0x03}, //F#4
{315,0x03}, //G4
{299,0x03}, //G#4
{281,0x03}, //A4
{267,0x03}, //A#4
{249,0x03}, //B4
{237,0x03}, //C5
{225,0x03}, //C#5
{209,0x03}, //D5
{201,0x03}, //D#5
{187,0x03}, //E5
{179,0x03} //F5
};

void setup()
{
  DDRD = DDRD & 0x03; //preserve 1s in the two LSBs; other pins are
switch inputs from keys
  DDRB = DDRB & 0xfe; //preserve 1s everywhere except the LSB; it's a
key input
  Serial.begin(9600);
  Wire.begin(); // wake up I2C bus

  DDRB |= 0x0c; //pins 10 and 11 are OUTPUTs
  //PORTB |=0x08; //initialize pin 11 to HIGH

  cli();//stop interrupts

//timer0 setup
  TCCR0A = 0;// zero out TCCR0A to begin with
  TCCR0B = 0;// same for TCCR2B
  TCNT0  = 0;//initialize counter value to 0

  //turn on CTC mode by setting WGM01 bit to 1
```

```
  TCCR0A |= (1 << WGM01);

  //set CS00 and CS01 for 64 prescaler
  TCCR0B |= (1 << CS00);
  TCCR0B |= (1 << CS01);

  // enable timer compare interrupt
  TIMSK0 |= (1 << OCIE0A);

//timer2 setup
  TCCR2A = 0;// set entire TCCR2A register to 0
  TCCR2B = 0;// same for TCCR2B
  TCNT2  = 0;//initialize counter value to 0

  // turn on CTC mode
  TCCR2A |= (1 << WGM01);

  // Set CS00 and CS01 bit for 64 prescaler
  TCCR2B |= (1 << CS00);
  TCCR2B |= (1 << CS01);

  // enable timer compare interrupt
  TIMSK2 |= (1 << OCIE2A);

//timer1 (16 bit) setup
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0

  // turn on CTC mode
  TCCR1B |= (1 << WGM12);

  // Set CS10 and CS12 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);

  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  sei();//allow interrupts
}

ISR(TIMER0_COMPA_vect){//timer0 interrupt toggles pin 11 if gate
allows
//generates pulse wave
  PORTB ^= (gate&0x08); //PORTB pin 3 --> root note
}

ISR(TIMER1_COMPA_vect){//timer1 interrupt toggles pin 12
  //generates pulse wave
  PORTB ^= (gate&0x10); //PORTB pin 4 --> Octave down
```

```
}

ISR(TIMER2_COMPA_vect){//timer2 interrupt toggles pin 10 if gate
allows
//generates pulse wave
  PORTB ^= (gate&0x04); //PORTB pin 2 --> +7 semitones
}

//inline void checkForNote(){
void loop(){
  // check all ports, in order: aux B, aux A, D, B
  //  |
  //  if note detected or not, update GATE, port, noteMap, OCR0A,
TCCR0B accordingly
  //   early return if we get to current note&port and it's still
pressed
  uint8_t noteIndex;
  for(byte portTest=0x01; portTest<0x10; portTest = portTest<<1){
    byte curPortVals; //holds bits for port under investigation
    switch(portTest){
      case 0x01:
        {
          noteIndex = 0; //start here and iterate
          Wire.beginTransmission(0x20);
          Wire.write(0x13); // set MCP23017 memory pointer to GPIOB
address
          Wire.endTransmission();
          Wire.requestFrom(0x20, 1); // request one byte of data from
MCP20317
          curPortVals=Wire.read(); // store the incoming byte into
curPortVals
        }
        break;
      case 0x02:
        {
          noteIndex = 8; //start here and iterate
          Wire.beginTransmission(0x20);
          Wire.write(0x12); // set MCP23017 memory pointer to GPIOA
address
          Wire.endTransmission();
          Wire.requestFrom(0x20, 1); // request one byte of data from
MCP20317
          curPortVals=Wire.read(); // store the incoming byte into
curPortVals
        }
        break;
      case 0x04:
        {
          noteIndex = 16; //start here and iterate
          curPortVals = (0x3f & ((PIND & 0xfc)>>2)); //store inputs on
```

```
port D into curPortVals
        }
        break;
      case 0x08:
        {
          noteIndex = 22;
          curPortVals = (PINB & 0x01); //only the last port is left;
store port B inputs in curPortVals
          if(curPortVals == 0x00){
            gate=0x00; //if we got here, we're done; no notes are
being pressed anymore
            Serial.print("gate off: ");
            Serial.println(gate, BIN);
            return;
          }
        }
        break;
    }

    //check to see if we're at the currently-pressed note yet
    /*if( ((portTest&pressedPort) != 0x00) && ((noteMap&curPortVals)
== noteMap) && ((noteMap^curPortVals) > noteMap ||
(noteMap^curPortVals) == 0x00)){
      Serial.println("I got here!");
      return; // leave this function; we got to the current note from
below

    }*/
    if(curPortVals == 0x00){ //no note pressed here
      continue; //nothing to see on this port, so let's move on
    }
    //now,
    // 1) we have a note here
    // 2) this port holds the lowest pressed note
    //   loop through each note until we find ours

    for(byte testNoteMap = 0x01; testNoteMap != 0x00;
testNoteMap=testNoteMap<<1){
      if((curPortVals&testNoteMap) == testNoteMap){
        //we found the note! Set everything appropriately, and then
get out of here
        noteMap = testNoteMap;
        pressedPort = portTest;
        //timer 0 settings
        OCR0A = notes[noteIndex+noteIndexOffset][0];
        TCCR0B = ((TCCR0B&0xf8)|notes[noteIndex+noteIndexOffset][1]);

        //timer 1 settings
        OCR1A = notes16bit[noteIndex+note1IndexOffset][0];
        TCCR1B = ((TCCR1B&0xf8)|notes16bit[noteIndex+note1IndexOffset]
```

```
[1]);
        if(TCNT1>notes16bit[noteIndex+note1IndexOffset][0])
          TCNT1 = 0; //reset if the count's too big; this should only
happen at most once in a note press

        //timer 2 settings
        OCR2A = notes[noteIndex+note2IndexOffset][0];
        TCCR2B = ((TCCR2B&0xf8)|notes[noteIndex+note2IndexOffset][1]);

        gate = 0xff;
        return;
      }
      noteIndex++;
    }
  }
//}
//
//
//void loop(){
//  checkForNote();
  //Serial.println(gate);
}
```