


```

byte typecurrent = 8;
byte typelast;
byte typecounter[4];
byte i;

//variables for PW pot monitoring
float pulseWidth;
int pulseWidthScaled;
int PWCCurrent;
byte PWTolerance = 8;//adjust this to increase/decrease stability of PW measurement

//variables for freq pot monitoring
int frequency;
int freqCurrent;
byte freqTolerance = 2;//adjust this to increase/decrease stability of frequency measurement
unsigned int freqscaled;

byte wave;
long t;
long samplerate;
long period;

//storage variables- I used these to cut down on the math being performed during the interrupt
int sawByte = 0;
byte sawInc;
int triByte = 0;
byte triInc;
int sinNum = 0;
int sinInc;

void setup() {

//set port/pin mode
DDRD = 0xFF;//all outputs
DDRC = 0x00;//all inputs
DDRB = 0xFF;//all outputs
//TIMER INTERRUPT SETUP

cli();//disable interrupts
//timer 1:
TCCR1A = 0;// set entire TCCR1A register to 0
TCCR1B = 0;// same for TCCR1B
//set compare match register- 100khz to start
OCR1A = 159; // = (16 000 000 / 100 000) - 1 = 159
//turn on CTC mode
TCCR1B |= (1 << WGM12);
// Set CS10 bit for 0 prescaler
TCCR1B |= (1 << CS10);
// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);

samplerate = 100000;

PORTB = 0;

```

```
PORTB = 1<<type;
```

```
//initialize variables
```

```
frequency = analogRead(A5);//initialize frequency
```

```
freqscaled = 48*frequency+1;//from 1 to ~50,000\
```

```
period = samplerate/freqscaled;
```

```
pulseWidth = analogRead(A4);//initalize pulse width
```

```
pulseWidthScaled = int(pulseWidth/1023*period);
```

```
triInc = 511/period;
```

```
sawInc = 255/period;
```

```
sinInc = 20000/period;
```

```
sei();//enable interrupts
```

```
}
```

```
void checkFreq() {
```

```
  freqCurrent = analogRead(A5);
```

```
  if (abs(freqCurrent-frequency)>freqTolerance){//if reading from pot exceeds tolerance
```

```
    frequency = freqCurrent;//new frequency- number between 0 and 1024
```

```
    freqscaled = 48*frequency+1;//from 1 to ~50,000
```

```
    period = samplerate/freqscaled;
```

```
    pulseWidthScaled = int(pulseWidth/1023*period);
```

```
    triInc = 511/period;
```

```
    if (triInc==0){
```

```
      triInc = 1;
```

```
    }
```

```
    sawInc = 255/period;
```

```
    if (sawInc==0){
```

```
      sawInc = 1;
```

```
    }
```

```
    sinInc=20000/period;
```

```
  }
```

```
}
```

```
void checkPW() {
```

```
  PWCurrent = analogRead(A4);
```

```
  if (abs(PWCurrent-pulseWidth)>PWTolerance){//if reading from pot exceeds tolerance
```

```
    pulseWidth = PWCurrent;//new pulse width, val between 0 and 1023
```

```
    pulseWidthScaled = int(pulseWidth/1023*period);
```

```
  }
```

```
}
```

```
void checkShape() { //debounce and check states of buttons
```

```
// 4 momentary switches control waveshape
```

```
// -pulse connects to A0
```

```
// -triangle A1
```

```
// -saw A2
```

```
// -sine A3
```

```
  typelast = typecurrent;
```

```
  if (digitalRead(A0)==LOW){
```

```
    typecurrent = 1;
```



```

}
else if (digitalRead(A1)==LOW){
    typecurrent = 2;
}
else if (digitalRead(A2)==LOW){
    typecurrent = 4;
}
else if (digitalRead(A3)==LOW){
    typecurrent = 8;
}

for (i=0; i<4; i++){
    if (i==type){
    }
    else{
        if ((typecurrent & (1 << i)) ^ (typelast & (1 << i))){//current diff than prev and debounce
            if ((typecurrent & (1 << i))){//currently depressed
                type = i;//set wave type
            }
            else {
                typecounter[i] = 12;//else set debounce counter to 12
            }
        }
        else if (((typecurrent & (1 << i)) == (typelast & (1 << i)))) { //if current same as prev and diff than debounce
            if (typecounter[i] > 0 && --typecounter[i] == 0) { //decrease debounce counter and check to see if = 0
                if ((typecurrent & (1 << i))){ //if debounce counter = 0 toggle debounced state
                    type = i;
                }
            }
        }
    }
}
}
}
}
}
}
}

```

```

ISR(TIMER1_COMPA_vect){ //timer 1 interrupt
    //increment t and reset each time it reaches period
    t += 1;
    if (t >= period){
        t = 0;
    }
    switch (type) {
        case 0://pulse
            if (pulseWidthScaled <= t) {
                wave = 255;
            }
        else{
            wave = 0;
        }
        break;
        case 1://triangle
            if((period-t) > t) {
                if (t == 0){

```

```

    triByte = 0;
}
else{
    triByte += triInc;
}
}
else{
    triByte -= triInc;
}
if (triByte>255){
    triByte = 255;
}
else if (triByte<0){
    triByte = 0;
}
wave = triByte;
break;
case 2://saw
if (t=0){
    sawByte=0;
}
else{
    sawByte+=sawInc;
}
wave = sawByte;
break;
case 3://sine
sinNum = t*sinInc;
wave = pgm_read_byte_near(sine20000 + sinNum);
break;
}

PORTD = wave;
}

void loop() {
    checkFreq();
    checkShape();
    checkPW();
    PORTB = 1<<type;
}

```