```
byte WaveArrayRise[256];  //Save space for CW Rise amplitudes in a table as bytes
byte WaveArrayFall[256];  //Save space for CW Fall amplitudes in a table as bytes
int i;  //Temporary variable as an integer
long int x; //Delay variable as an integer
float Ang;  //Angle in degrees
float Rad;  //Angle in radians
float xp; //Sigmoid exponent
float y;  //Sigmoid function
int RiTm; //Rise times
int FaTm; //Fall times
int FT = A0;
int RT = A1;
int FS = A2;
int RS = A3;
int SPD = A4;
long int CWSpeed;
int Duration;

int AAA = 0;

int FallTime;
int FallMs;
int RiseTime;
int RiseMs;
int FallShape;
int RiseShape;
int Speed;
int RFRSkip = 0;
int RFFSkip = 0;

void setup()
{
  Serial.begin(9600);

pinMode(13, OUTPUT);
DDRD = 0xff;  //Set up port D for output

FallTime = analogRead(FT);  //Input selected fall time (1023=8ms; 682=6ms; 340=3ms; 0=1ms)
RiseTime = analogRead(RT);  //Input selected rise time (1023=8ms; 682=6ms; 340=3ms; 0=1ms)
FallShape = analogRead(FS); //Input selected fall shape (1023=Square; 819=RC; 614=Ramp; 409=RaisedCosine; 204=Sigmoid; 0=Spare)
RiseShape = analogRead(RS); //Input selected rise shape (1023=Square; 819=RC; 614=Ramp; 409=RaisedCosine; 204=Sigmoid; 0=Spare)
Speed = analogRead(SPD);  //Input keying speed (1023=8wpm; 682=15wpm; 340=30wpm; 0=60wpm)
FallTimeChoice(); //Select a fall time
RiseTimeChoice(); //Select a rise time
FallShapeChoice(); //Select a fall shape
RiseShapeChoice(); //Select a rise shape
HighLowDurationChoice();  // select high and low duration based on keying speed


noInterrupts(); //Turn off interrupts so waveforms will be pure

}




//*************************
void loop()
{
  if(RFRSkip == 0) //Skip the rise routine if a square shape  for rise
  {
    Rise();
  }
  HoldUp();
  if(RFFSkip == 0)  //Skip the fall routine if a square shape  for fall
  {
    Fall();
  }
  HoldDown();
}


//*************************
void Rise()
{
  for(i = 0; i <= 255; i++)
  {
    PORTD = WaveArrayRise[i];
    MyDelay(RiTm);
```

```
  }
}



void Fall()
{
  for(i = 0; i <= 255; i++)
  {
    PORTD = WaveArrayFall[i];
    MyDelay(FaTm);
  }
}

void MyDelay(long int DelayTime)
{
  for(int x = 0; x <= DelayTime; x++)
  {
    digitalWrite(13, LOW);
  }
}

void HoldUp()
{
  PORTD = 255;
  MyDelay(CWSpeed + 0);
}

void HoldDown()
{
  PORTD = 0;
  MyDelay(CWSpeed + 0);
}



void RaisedCosRise()
{
  for(i = 0; i <= 255; i++)
  {
    Ang = 180 + (i * 0.703125);
    Rad = Ang * .0174533;
    WaveArrayRise[i] = int (255 *(cos(Rad) + 1) / 2);
  }
  if(RiseMs == 1.6) {RiTm = 1;}
  if(RiseMs == 3) {RiTm = 3;}
  if(RiseMs == 5) {RiTm = 7;}
  if(RiseMs == 8) {RiTm = 14;}
}

void RaisedCosFall()
{
  for(i = 0; i <= 255; i++)
  {
    Ang = i * 0.703125;
    Rad = Ang * .0174533;
    WaveArrayFall[i] = int (255 *((cos(Rad) + 1) / 2));
  }
  if(FallMs == 1.6) {FaTm = 1;}
  if(FallMs == 3) {FaTm = 3;}
  if(FallMs == 5) {FaTm = 7;}
  if(FallMs == 8) {FaTm = 14;}
}

void LinRise()
{
  for(i = 0; i <= 255; i++)
  {
    WaveArrayRise[i] = i;
  }
  if(RiseMs == 1.6) {RiTm = 0;}
  if(RiseMs == 3) {RiTm = 2;}
  if(RiseMs == 5) {RiTm = 5;}
  if(RiseMs == 8) {RiTm = 10;}
}

void LinFall()
{
  for(i = 0; i <= 255; i++)
  {
```

```
    WaveArrayFall[i] = 255 - i;
  }
  if(FallMs == 1.6) {FaTm = 0;}
  if(FallMs == 3) {FaTm = 2;}
  if(FallMs == 5) {FaTm = 5;}
  if(FallMs == 8) {FaTm = 10;}
}

void SigmoidRise()
{
  for(i = 0; i <= 255; i++)
  {
    xp = (i * 0.039) - 5;
    y = 1/(1 + exp(-xp));
    WaveArrayRise[i] = int (255 * y);
  }
  if(RiseMs == 1.6) {RiTm = 2;}
  if(RiseMs == 3) {RiTm = 6;}
  if(RiseMs == 5) {RiTm = 10;}
  if(RiseMs == 8) {RiTm = 18;}
}

void SigmoidFall()
{
  for(i = 0; i <= 255; i++)
   {
    xp = (i * 0.039) - 5;
    y = 1/(1 + exp(-xp));
    WaveArrayFall[i] = int(255 - (255 * y));
   }
  if(FallMs == 1.6) {FaTm = 2;}
  if(FallMs == 3) {FaTm = 6;}
  if(FallMs == 5) {FaTm = 10;}
  if(FallMs == 8) {FaTm = 18;}
}




void SquareRise()
{
  RFRSkip = 1;
}


void SquareFall()
{
  RFFSkip = 1;
}

void RCExponentialRise()
{
  for(i = 0; i <= 255; i++)
  {
    WaveArrayRise[i] = 255 * (1 - exp(-0.022 * i));
  }
  if(RiseMs == 1.6) {RiTm = 2;}
  if(RiseMs == 3) {RiTm = 6;}
  if(RiseMs == 5) {RiTm = 11;}
  if(RiseMs == 8) {RiTm = 20;}

}

void RCExponentialFall()
{
  for(i = 0; i <= 255; i++)
  {
    WaveArrayFall[i] = 255 * exp(-0.022 * i);
  }
  if(FallMs == 1.6) {FaTm = 2;}
  if(FallMs == 3) {FaTm = 6;}
  if(FallMs == 5) {FaTm = 14;}
  if(FallMs == 8) {FaTm = 20;}
}

void RiseShapeChoice()   //Determine the rising shape
{
  if(RiseShape > 983)
  {
    SquareRise();
```

```
  }
  else if((RiseShape < 860) && (RiseShape > 780))
  {
    RCExponentialRise();
  }
  else if((RiseShape < 655) && (RiseShape > 575))
  {
    LinRise();  //Ramp up
  }
  else if((RiseShape < 450) && (RiseShape > 370))
  {
    RaisedCosRise();  //Raised cosine up
  }
  else if((RiseShape < 245) && (RiseShape > 164))
  {
    SigmoidRise();  //Sigmoid up
  }
  else if(RiseShape < 125)
  {
    //Spare
  }
}

void FallShapeChoice()   //Determine the falling shape
{
  if(FallShape > 983)
  {
    SquareFall();
  }
  else if((FallShape < 860) && (FallShape > 780))
  {
    RCExponentialFall();
  }
  else if((FallShape < 655) && (FallShape > 575))
  {
    LinFall();  //Ramp down
  }
  else if((FallShape < 450) && (FallShape > 370))
  {
    RaisedCosFall();//Raised cosine down12750
  }
  else if((FallShape < 245) && (FallShape > 164))
  {
    SigmoidFall();  //Sigmoid down
  }
  else if(FallShape < 125)
  {
    //Spare
  }
}



void HighLowDurationChoice() //Determine the high/low duration from the speed
//Speed is adjusted on a square wave.  Fine adjustment is done for Sigmoid,
//Raised Cosine, and Linear waveforms for 40 WPM with a 5 ms rise and fall time.
//Speed is measured from the start of RF to the start of RF on the next cycle.
{
  if(Speed > 983) {CWSpeed = 16500;} //20 wpm 60ms dit

  else if((Speed < 722) && (Speed > 642)) {CWSpeed = 10000;} //30 wpm 40ms dit


  else if((Speed < 381) && (Speed > 301) && (RiseShape < 265) && (RiseShape > 154)) {CWSpeed = 6000;}   //40 wpm 30ms dit for Sigmoid shape

   else if((Speed < 381) && (Speed > 301) && (RiseShape < 450) && (RiseShape > 370)) {CWSpeed = 6800;}   //40 wpm 30ms dit for Raised Cosine
shape

  else if((Speed < 381) && (Speed > 301) && (RiseShape < 675) && (RiseShape > 575)) {CWSpeed = 7250;}   //40 wpm 30ms dit for Linear shape

   else if((Speed < 381) && (Speed > 301) && (RiseShape < 860) && (RiseShape > 780)) {CWSpeed = 5500;}   //40 wpm 30ms dit for RC Exponential
shape

   else if((Speed < 381) && (Speed > 301) && (RiseShape > 983)) {CWSpeed = 9570;}   //40 wpm 30ms dit for RC Square Wave shape

  else if(Speed < 261) {CWSpeed = 5000;}  //50 wpm 24ms dit}
  }


void RiseTimeChoice() //Determine rise time choice
```

```
{
  if(RiseTime > 983)
  {
    RiseMs = 8;  //8 ms rise time
  }
  else if((RiseTime < 722) && (RiseTime > 642))
  {
    RiseMs = 5;  //5 ms rise time
  }
  else if((RiseTime < 381) && (RiseTime > 301))
  {
    RiseMs = 3;  //3 ms rise time
  }
  else if(RiseTime < 261)
  {
    RiseMs = 1;  //1 ms rise time
  }
}

void FallTimeChoice() //Determine fall time choice
{
  if(FallTime > 983)
  {
    FallMs = 8;  //8 ms fall time
  }
  else if((FallTime < 722) && (FallTime > 642))
  {
    FallMs = 5;  //5 ms fall time
  }
  else if((FallTime < 381) && (FallTime > 301))
  {
    FallMs = 3;  //3 ms fall time
  }
  else if(FallTime < 261)
  {
    FallMs = 1;  //1 ms fall time
  }
}
```