/\*

Simple Waveform generator with Arduino Due

\* connect two push buttons to the digital pins 2 and 3

with a 10 kilohm pulldown resistor to choose the waveform

to send to the DAC0 and DAC1 channels

\* connect a 10 kilohm potentiometer to A0 to control the

signal frequency

\*/

#include "Waveforms.h"

#define oneHzSample 1000000/maxSamplesNum // sample for the 1Hz signal expressed in microseconds

const int button0 = 2; //button1 = 3;

volatile int wave0 = 0; //wave1 = 0;

int i = 0;

int sample;

void setup() {

Serial.begin(9600);

//analogWriteResolution(12); // set the analog output resolution to 12 bit (4096 levels)

//analogReadResolution(12); // set the analog input resolution to 12 bit

//attachInterrupt(button0, wave0Select, RISING); // Interrupt attached to the button connected to pin 2

//attachInterrupt(button1, wave1Select, RISING); // Interrupt attached to the button connected to pin 3

}

void loop() {

// Read the the potentiometer and map the value between the maximum and the minimum sample available

// 1 Hz is the minimum freq for the complete wave

// 170 Hz is the maximum freq for the complete wave. Measured considering the loop and the analogRead() time

sample = map(analogRead(A0), 0, 4095, 0, oneHzSample);

sample = constrain(sample, 0, oneHzSample);

analogWrite(PWM2, waveformsTable[wave0][i]); // write the selected waveform on DAC0

//analogWrite(2,450.5);

//Serial.println(analogRead(A2));

//analogWrite(A3, waveformsTable[wave1][i]); // write the selected waveform on DAC1

//exit(0);

i++;

if(i == maxSamplesNum) // Reset the counter to repeat the wave

i = 0;

delayMicroseconds(sample); // Hold the sample value for the sample time

}

// function hooked to the interrupt on digital pin 2

void wave0Select() {

wave0++;

if(wave0 == 4)

wave0 = 0;

}

// function hooked to the interrupt on digital pin 3

//void wave1Select() {

//wave1++;

//if(wave1 == 4)

//wave1 = 0;

//}

----------------------------------------

#ifndef \_Waveforms\_h\_

#define \_Waveforms\_h\_

#define maxWaveform 4

#define maxSamplesNum 120

static int waveformsTable[maxWaveform][maxSamplesNum] = {

// Sin wave

{

0x7ff, 0x86a, 0x8d5, 0x93f, 0x9a9, 0xa11, 0xa78, 0xadd, 0xb40, 0xba1,

0xbff, 0xc5a, 0xcb2, 0xd08, 0xd59, 0xda7, 0xdf1, 0xe36, 0xe77, 0xeb4,

0xeec, 0xf1f, 0xf4d, 0xf77, 0xf9a, 0xfb9, 0xfd2, 0xfe5, 0xff3, 0xffc,

0xfff, 0xffc, 0xff3, 0xfe5, 0xfd2, 0xfb9, 0xf9a, 0xf77, 0xf4d, 0xf1f,

0xeec, 0xeb4, 0xe77, 0xe36, 0xdf1, 0xda7, 0xd59, 0xd08, 0xcb2, 0xc5a,

0xbff, 0xba1, 0xb40, 0xadd, 0xa78, 0xa11, 0x9a9, 0x93f, 0x8d5, 0x86a,

0x7ff, 0x794, 0x729, 0x6bf, 0x655, 0x5ed, 0x586, 0x521, 0x4be, 0x45d,

0x3ff, 0x3a4, 0x34c, 0x2f6, 0x2a5, 0x257, 0x20d, 0x1c8, 0x187, 0x14a,

0x112, 0xdf, 0xb1, 0x87, 0x64, 0x45, 0x2c, 0x19, 0xb, 0x2,

0x0, 0x2, 0xb, 0x19, 0x2c, 0x45, 0x64, 0x87, 0xb1, 0xdf,

0x112, 0x14a, 0x187, 0x1c8, 0x20d, 0x257, 0x2a5, 0x2f6, 0x34c, 0x3a4,

0x3ff, 0x45d, 0x4be, 0x521, 0x586, 0x5ed, 0x655, 0x6bf, 0x729, 0x794

}

,

// Triangular wave

{

0x44, 0x88, 0xcc, 0x110, 0x154, 0x198, 0x1dc, 0x220, 0x264, 0x2a8,

0x2ec, 0x330, 0x374, 0x3b8, 0x3fc, 0x440, 0x484, 0x4c8, 0x50c, 0x550,

0x594, 0x5d8, 0x61c, 0x660, 0x6a4, 0x6e8, 0x72c, 0x770, 0x7b4, 0x7f8,

0x83c, 0x880, 0x8c4, 0x908, 0x94c, 0x990, 0x9d4, 0xa18, 0xa5c, 0xaa0,

0xae4, 0xb28, 0xb6c, 0xbb0, 0xbf4, 0xc38, 0xc7c, 0xcc0, 0xd04, 0xd48,

0xd8c, 0xdd0, 0xe14, 0xe58, 0xe9c, 0xee0, 0xf24, 0xf68, 0xfac, 0xff0,

0xfac, 0xf68, 0xf24, 0xee0, 0xe9c, 0xe58, 0xe14, 0xdd0, 0xd8c, 0xd48,

0xd04, 0xcc0, 0xc7c, 0xc38, 0xbf4, 0xbb0, 0xb6c, 0xb28, 0xae4, 0xaa0,

0xa5c, 0xa18, 0x9d4, 0x990, 0x94c, 0x908, 0x8c4, 0x880, 0x83c, 0x7f8,

0x7b4, 0x770, 0x72c, 0x6e8, 0x6a4, 0x660, 0x61c, 0x5d8, 0x594, 0x550,

0x50c, 0x4c8, 0x484, 0x440, 0x3fc, 0x3b8, 0x374, 0x330, 0x2ec, 0x2a8,

0x264, 0x220, 0x1dc, 0x198, 0x154, 0x110, 0xcc, 0x88, 0x44, 0x0

}

,

// Sawtooth wave

{

0x22, 0x44, 0x66, 0x88, 0xaa, 0xcc, 0xee, 0x110, 0x132, 0x154,

0x176, 0x198, 0x1ba, 0x1dc, 0x1fe, 0x220, 0x242, 0x264, 0x286, 0x2a8,

0x2ca, 0x2ec, 0x30e, 0x330, 0x352, 0x374, 0x396, 0x3b8, 0x3da, 0x3fc,

0x41e, 0x440, 0x462, 0x484, 0x4a6, 0x4c8, 0x4ea, 0x50c, 0x52e, 0x550,

0x572, 0x594, 0x5b6, 0x5d8, 0x5fa, 0x61c, 0x63e, 0x660, 0x682, 0x6a4,

0x6c6, 0x6e8, 0x70a, 0x72c, 0x74e, 0x770, 0x792, 0x7b4, 0x7d6, 0x7f8,

0x81a, 0x83c, 0x85e, 0x880, 0x8a2, 0x8c4, 0x8e6, 0x908, 0x92a, 0x94c,

0x96e, 0x990, 0x9b2, 0x9d4, 0x9f6, 0xa18, 0xa3a, 0xa5c, 0xa7e, 0xaa0,

0xac2, 0xae4, 0xb06, 0xb28, 0xb4a, 0xb6c, 0xb8e, 0xbb0, 0xbd2, 0xbf4,

0xc16, 0xc38, 0xc5a, 0xc7c, 0xc9e, 0xcc0, 0xce2, 0xd04, 0xd26, 0xd48,

0xd6a, 0xd8c, 0xdae, 0xdd0, 0xdf2, 0xe14, 0xe36, 0xe58, 0xe7a, 0xe9c,

0xebe, 0xee0, 0xf02, 0xf24, 0xf46, 0xf68, 0xf8a, 0xfac, 0xfce, 0xff0

}

,

// Square wave

{

0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff,

0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff,

0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff,

0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff,

0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff,

0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff, 0xfff,

0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,

0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,

0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,

0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,

0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,

0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0

}

};

#endif

<http://www.eprojectszone.com/how-to-generate-a-sine-wave-from-arduino-or-atmega-328/>

https://robs-blog.net/2017/06/28/so-you-want-to-make-a-sine-wave-in-arduino/