Strobe Sequence Generator 1

This file generates a pattern which alternates between two strobe frequencies at a given rate (uses functions from the Signal Processing Toolbox). For this example the brighness of the central LED and the on-brightness of the ring LEDs is kept constant.

Parameters

First we define the parameters used in the strobe generation:

```
durationSeconds = 10;
strobeHzA = 4;
strobeHzB = 20;
abSwitchRateHz = 0.5;
centralBrightness = 10;
ringBrightness = 255;
```

strobeHzA - The first strobe rate (set here as 4Hz).

strobeHzB - The second strobe rate (set here as 20Hz).

abSwitchRateHz - The rate at which the output completes a cycle of displaying strobe rate A and then displaying strobe rate B (set here as 0.5Hz, switching every 1 second).

Generating the Sequence

The strobe device plays back samples at a fixed rate of 2ksps (2000 samples per second), and so we need to generate a sample for every 2000th of a second for the entire duration.

```
frameDurationS = (1/2000); % Time duration of each frame
sampleTimes = (0:frameDurationS:durationSeconds - frameDurationS)'; %
Generate a list sample timestamps
```

sampleTimes will now contain the timestamps for each of the frames:

[0, 0.0005, 0.001, 0.0015, 0.002, ...]

From these timestamps we can generate square waves for the two strobe rates, and also for the switching control.

```
strobeA = (1 + square(sampleTimes * 2 * pi * strobeHzA)) ./ 2; % Generate
strobe waveform for frequency A (and shift to be 0-1-0-1)
strobeB = (1 + square(sampleTimes * 2 * pi * strobeHzB)) ./ 2; % Generate
strobe waveform for frequency B (and shift to be 0-1-0-1)
abSwitch = (1 + square(sampleTimes * 2 * pi * abSwitchRateHz)) ./ 2; %
Generate square waveform for output switching (A = 1, B = 0) (and shift to
be 0-1-0-1)
```

These square waves are plotted in the graphs below.

We can also combine them to make the final output wave:

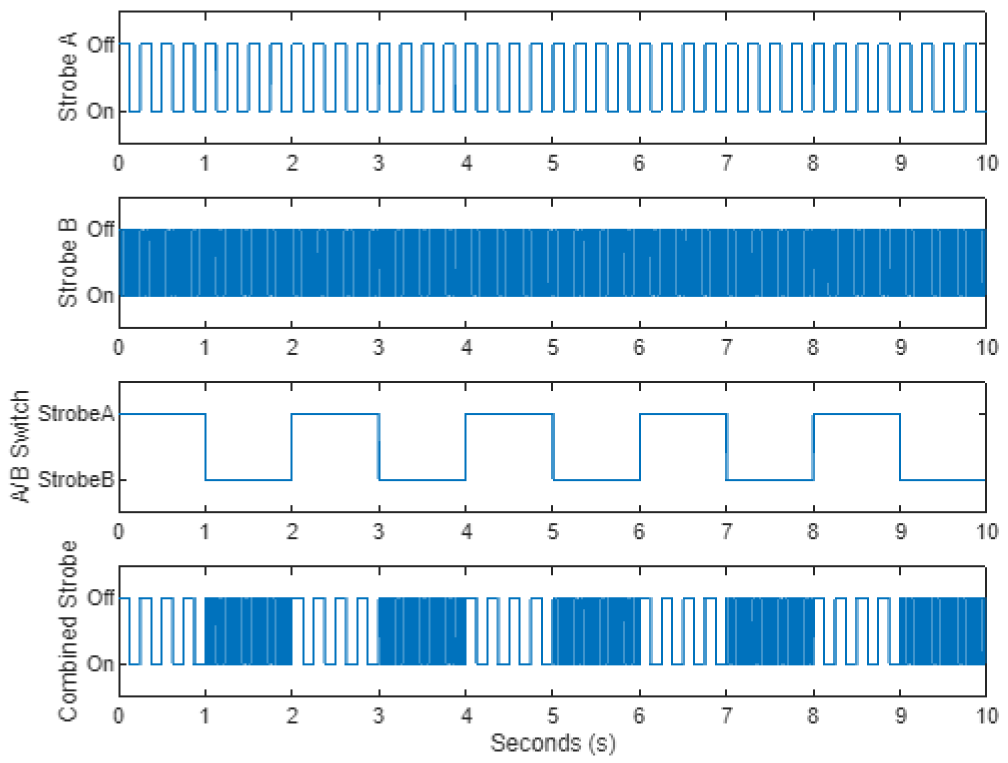
```
combinedStrobe = (abSwitch .* strobeA) + ((1-abSwitch) .* strobeB); % 1 =
LED on, 0 = LED off
```

Here when abSwitchRateHz is 1, we use the output of strobeA (multiplied by 1) and the output of strobeB is off (multiplied by 0).

When abSwitchRateHz is 0, this equation is reversed.

The four waveforms can be seen in the below plot:

```
figure;
tiledlayout(4,1)
nexttile
plot(sampleTimes, strobeA)
ylabel("Strobe A")
ylim([-0.5,1.5])
yticks([0,1]);
yticklabels(["On", "Off"])
nexttile
plot(sampleTimes, strobeB)
ylabel("Strobe B")
ylim([-0.5,1.5])
yticks([0,1]);
yticklabels(["On", "Off"])
nexttile
plot(sampleTimes, abSwitch)
ylim([-0.5,1.5])
yticks([0,1]);
ylabel('A/B Switch')
yticklabels(["StrobeB", "StrobeA"])
nexttile
plot(sampleTimes, combinedStrobe)
ylabel("Combined Strobe")
ylim([-0.5,1.5])
yticks([0,1]);
yticklabels(["On", "Off"])
xlabel("Seconds (s)")
```



Data Packet Generation

This combined strobe output now contains the on/off state (as a 1 or 0) for the ring LEDs for each of the 2ksps of the sequence.

Using this we can now generate the full data packet for each of the samples, each sample is defined as follows:

<Ring LED on/off bitmap>,<Central brightness>,<North brightness>,<East brightness>,<South brightness>,<West brightness>

Where each value is a 8-bit unsigned integer (0-255).

We will start with the Ring LED ON/OFF bitmap, this value has 8 bits and each bit represents the on/off state of a single ring LED.

We have previously generated the on/off state for each sample and since we are controlling all LEDs with the same signal we need to repeat that value for all 8 LEDs:

```
ledONOFFsamples = repmat(combinedStrobe, 1, 8);
```

Next we need to convert this array of 8 bits (1's or 0's) for each sample into a single 8-bit unsigned integer using this small helper function:

```
ledONOFFBitmap = binary8ToUint8(ledONOFFsamples); % Use the strobe signal to
turn on and off the ring LED states
```

(this function is defined at the end of this file)

Since we are controlling all the leds with the same input signal the ledONBitmap should be a single column now contain the following:

[255;

...;

0;

...;

255;

...] and so on.

We need to append to the right of each of these values the brightness values for each channel, these values are all the same so we generate a single row with the following command:

```
dacChannelValues = [centralBrightness, ringBrightness, ringBrightness,
ringBrightness, ringBrightness];
```

And then repeat that row for every single sample:

```
dacChannelValuesPerSample = repmat(dacChannelValues, [length(sampleTimes),
1]);
```

and append them to the ledONOFFBitmap values:

```
preparedStrobeData2D = [ledONOFFBitmap, dacChannelValuesPerSample];
```

This should be a 2D matrix where each row contains a single data packet and there is a row for each displayed sample.

In order to transmit this data we need to append the rows into a single 1D sequence using the following command:

```
preparedStrobeData1D = reshape(preparedStrobeData2D',
[size(preparedStrobeData2D, 1) * size(preparedStrobeData2D, 2), 1])';
```

This data can then be used with the DeviceUsageFromFileExample.m

```
function value = binary8ToUint8(bitArray)
    value = sum([2^7 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0] .* bitArray, 2);
    return;
end
```