September 16, 2009

Supplementary Information for:
Quantitative measurement of zinc secretion from pancreatic islets with high temporal resolution using droplet-based microfluidics

Christopher J. Easley[1, †], Jonathan V. Rocheleau[1,‡], W. Steven Head[1], David W. Piston[1,2,*]
[1] Vanderbilt University, Molecular Physiology and Biophysics, Vanderbilt University Medical Center, 702 Light Hall, Nashville, TN 37232, USA

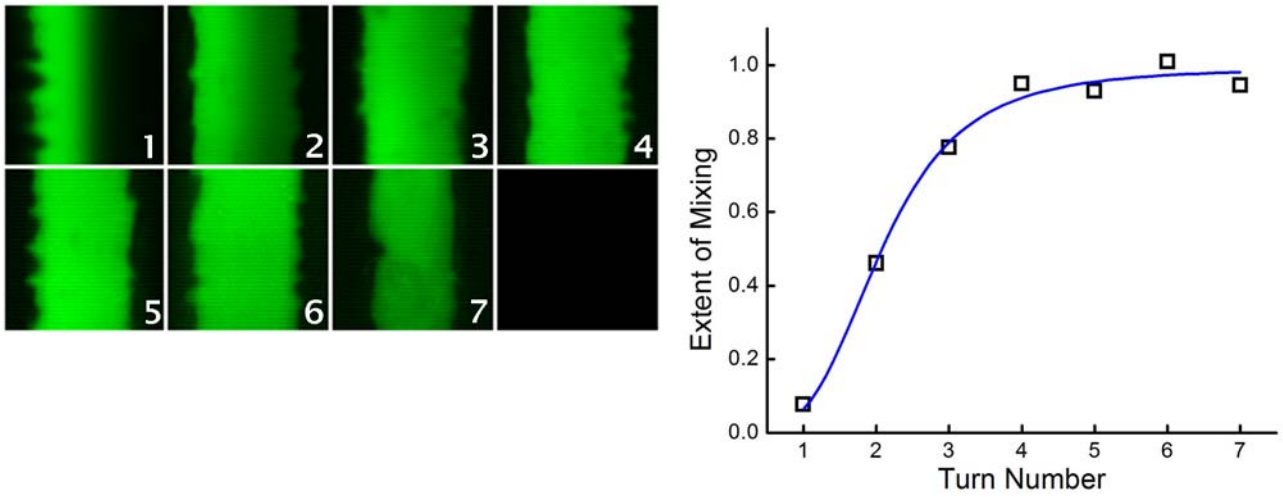

Figure S1. Diffusive mixing during calibration. To confirm that diffusive mixing was complete at the droplet formation region during on-chip calibrations (**Fig. 3C** of manuscript), each turn of the aqueous channel (R_{aq}) was imaged during calibration flow conditions at 37 °C. (**A**) Diffusive mixing was essentially complete by turn 4 of 7 in the aqueous channel. (**B**) Image analysis confirmed this result.

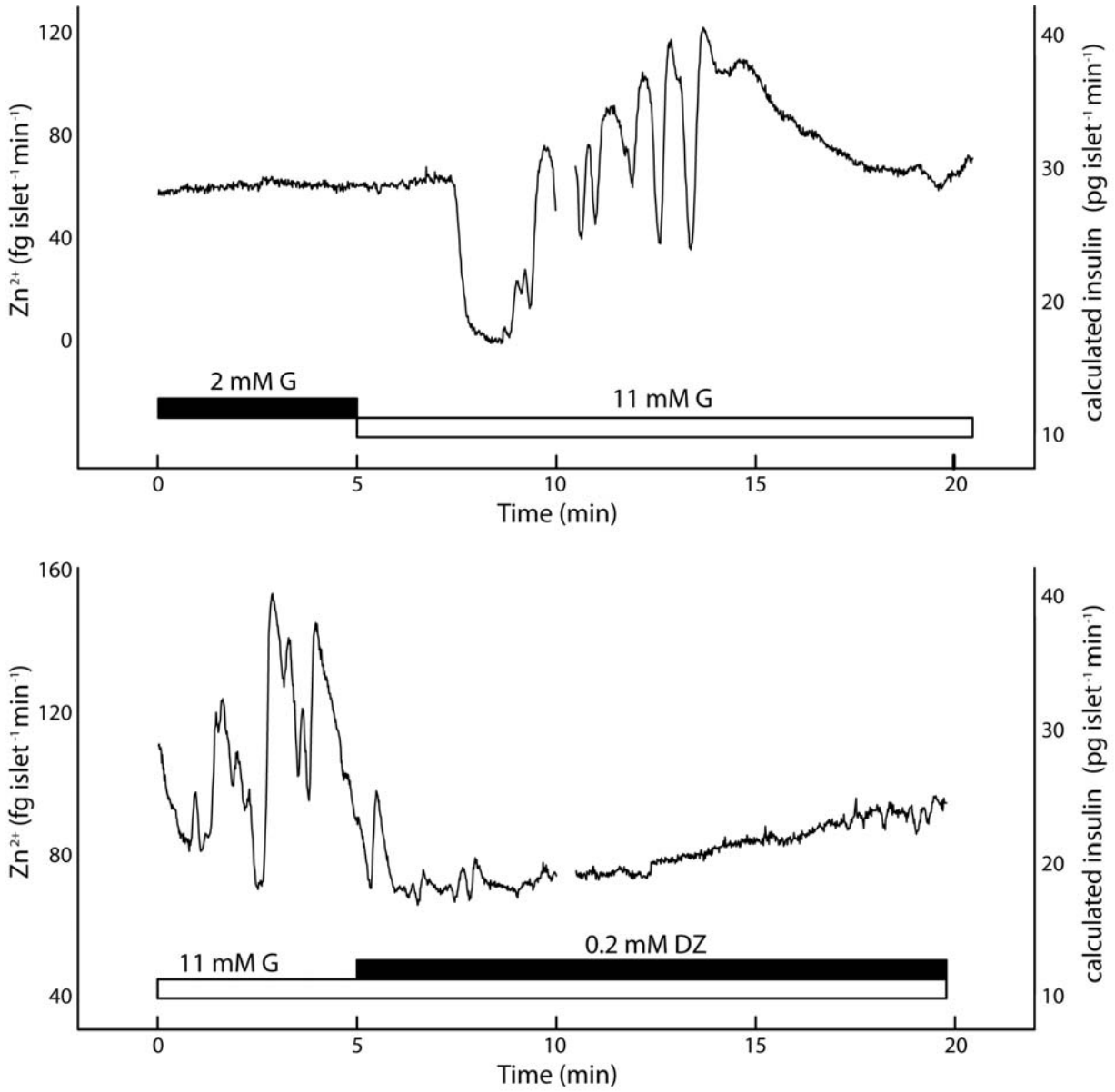

Figure S2. Additional zinc secretion measurements from single islets. (**A**) A third starved islet (referred to here as islet Z5) was treated with 11 mM glucose (G) after 5 min of sampling and measurement, with similar responses observed (no diazoxide treatment, compared to **Fig. 4** in text). (**B**) A third unstarved islet (referred to as islet Z6) was treated with diazoxide (DZ), showing a similar loss of secretory activity compared to **Fig. 6A**.

Matlab Code
Droplock10.m

```
% Version 10 of this code reworked the lock-in method.  The masking
% procedure now uses only an aqueous mask, multiplying data inside the
% droplets by the 'gain' factor (an input now), then dividing all data by
% the same 'gain' factor before averaging by summing and dividing.
%
% This code is designed to load data that has been uploaded as raw files of
% the names 'red' and 'green'.  These files were videos from the LSM 5Live
% that have been pre-loaded as '.raw' files by ImageJ.
%
% IMPORTANT NOTE ON DIMENSIONALITY:  ImageJ saves '.raw' files as a
% continuous data set, with each point representing the pixels of the image
% starting from top-left, scanning left-to-right, and ending on bottom-
% right.  Each frame of a video is then scanned.  For example, a 128x32
% image with 1000 frames would give 128*32=4096 data points before moving
% to the second frame.  Thus, this '.raw' file would have 4,096,000 data
% points.  This scanning effect, as defined by ImageJ, is different than
% the display as defined by Matlab.  Therefore, in this m-file, the matrix
% of the '.raw' file is loaded, then transposed to allow the images to be
% displayed the same as ImageJ, on-screen.
%
% An example of how to pre-load the data is:
%  clear
%  fid = fopen('oil_stack.raw','r');
%  tic; red = fread(fid,'uchar'); toc
%  fid = fopen('aq_stack.raw','r');
%  tic; green = fread(fid,'uchar'); toc
%
% An example command to run this file is:
%  [Saq, Sbg, S] = DropLock10(green, red, 40, 0, 1000);
%
% Also, an example command set to run for multiple folders is:
%  tic;
%  for n = 1:600
%  [Saq, Sbg, S] = DropLock10(green, red, 40, (n-1)*40, 1000);
%  dataset(n,:) = [Saq; Sbg; S];
%  n
%  toc
%  end;
%  save filename.txt dataset -ascii -double -tabs;
%
%
% Inputs:   1) aqueous data ('green')
```

```
%         2) oil data ('red')
%         3) width of ROI in X-axis
%         4) width of ROI in Y-axis
%         5) number of frames to load
%         6) starting frame
%         7) gain of lock-in
%
% Outputs:  1) aqueous fluorescence signal (Saq)
%         2) background fluorescence signal (Sbg)
%         3) Saq - Sbg (S)
%

function[Saq, Sbg, S] = ...
   DropLock10(green, red, dx, dy, frames, start, gain)



% Open droplet data, and process.

%tic
FrameStart=start;
FrameEnd=start+frames-1;
TSize = frames;

XSize = dx;
YSize = dy;
TSize = frames;


% Load data. Transpose matrix due to RAW data file reading direction.
oil = zeros(128,32,TSize);
oil2 = zeros(32,128,TSize);
for t=start+1:start+TSize
for y=1:32
oil(:,y,t-start) = red((t-1)*32*128 + (y-1)*128+1:(t-1)*32*128 + y*128,1);
end
oil2(:,:,t-start)=oil(:,:,t-start).'; % Note the transpose operation --> .'
end
%figure; imshow(oil(:,:,1),[0 255]);
%figure; imshow(oil2(:,:,1),[0 255]);

clear oil


% Redefine X and Y sizes to define 'Data_oil'.
XSize = dy;
```

```
YSize = dx;


Data_oil = zeros(XSize,YSize,TSize);
if rem(XSize,2)>0
   sub=(XSize-3)/2;
   add=(XSize+1)/2;
else
   sub=(XSize-2)/2;
   add=(XSize)/2;
end
Data_oil(:,:,:) = double(oil2(32/2-sub:32/2+add,128-YSize+1:128,...
   1:frames));
clear red
clear oil2

%figure; imshow(Data_oil(:,:,1),[0 255]);

%'loaded oil files'
%'size of Data_oil'
%size(Data_oil)
%toc



% Median filter gives the background level in the oil signal for
% normalization.A 3x3 mean filter (Mxy) is also applied to the image.

Mxy = [1/9,1/9,1/9; 1/9,1/9,1/9; 1/9,1/9,1/9];
medo = median(Data_oil,3);
medo3D = zeros(XSize,YSize,TSize);
normo = zeros(XSize,YSize,TSize);

medo = conv2(medo,Mxy,'same');

for t=1:TSize
   medo3D(:,:,t) = medo;
end

normo = Data_oil./medo3D;
%'size of medo3D'
%size(medo3D)

clear Data_oil

% Sobel edge detection (derivative) kernels are generated (Gx & Gy), and
```

```
% the edge detection is used to produce masks for spatial lock-in.  A 3x3
% mean filter (Mxy) is also applied to the edge images.

Mxy = [1/9,1/9,1/9; 1/9,1/9,1/9; 1/9,1/9,1/9];
Gx = [1,2,1; 0,0,0; -1,-2,-1]; Gy = [1,0,-1; 2,0,-2; 1,0,-1];
Xder = zeros(XSize,YSize);
Yder = zeros(XSize,YSize);
edges = zeros(XSize,YSize,TSize);

for t = 1:TSize
    Xder = conv2(normo(:,:,t),Gx,'same');
    Yder = conv2(normo(:,:,t),Gy,'same');
    edges(:,:,t) = conv2(sqrt(Xder.*Xder + Yder.*Yder),Mxy,'same');
end
%'Edges detected'
%toc

% Create binary images from the edges.
edges = edges./(max(max(max(edges))));
level = graythresh(edges);
edgesbw = zeros(XSize,YSize,TSize);

for t = 1:TSize
    edgesbw(:,:,t) = im2bw(edges(:,:,t), level);
end
%'Binary images created'

% Create oil and aqueous masks.
normo = normo./(max(max(max(normo))));
normobw = zeros(XSize,YSize,TSize);
masko = zeros(XSize,YSize,TSize);
maska = zeros(XSize,YSize,TSize);

level = graythresh(normo);

for t = 1:TSize
    normobw(:,:,t) = im2bw(normo(:,:,t), level);
end

edgesbw = ~edgesbw;

masko = edgesbw.*normobw;
maska = edgesbw.*(~normobw);
zerobar = zeros(2,YSize,TSize);

% Aqueous mask is now changed to lock-in mask (1000's and 1's).
```

```
maska = (maska*(gain-1))+1;


% Chop off top and bottom bars (2 pix) of aq mask due to filtering anomalies.
masko(1:2,:,:) = zerobar;
masko(XSize-1:XSize,:,:) = zerobar;
maska(1:2,:,:) = zerobar;
maska(XSize-1:XSize,:,:) = zerobar;
%'masks generated'
%toc



%figure; imshow(masko(:,:,1),[0 255]);
%figure; imshow(maska(:,:,1),[0 255]);
%size(masko)
%size(maska)
% (to check mask generation)
%pause



% Load aq data. Transpose matrix due to RAW data file reading direction.
aq = zeros(128,32,TSize);
aq2 = zeros(32,128,TSize);
for t=start+1:start+TSize
for y=1:32
aq(:,y,t-start) = green((t-1)*32*128 + (y-1)*128+1:(t-1)*32*128 + y*128,1);
end
aq2(:,:,t-start)=aq(:,:,t-start).'; % Note the transpose operation --> .'
end
%figure; imshow(aq(:,:,1),[0 255]);
%figure; imshow(aq2(:,:,1),[0 255]);

clear aq


% Redefine X and Y sizes to define 'Data_oil'.
XSize = dy;
YSize = dx;


Data_aq = zeros(XSize,YSize,TSize);
Data_aq(:,:,:) = double(aq2(32/2-sub:32/2+add,128-YSize+1:128,...
   1:frames));
clear green
```

```
clear aq2

%figure; imshow(Data_aq(:,:,1),[0 255]);

%'loaded aqueous files'
%'size of Data_aq'
%size(Data_aq)
%toc


% Aqueous data is masked, and sums are calculated.

AQ = zeros(XSize,YSize,TSize);
BG = zeros(XSize,YSize,TSize);

normo = normo./(median(median(median(normo))));

for t = 1:TSize
    AQ(:,:,t) = (Data_aq(:,:,t).*maska(:,:,t))/gain;
    BG(:,:,t) = Data_aq(:,:,t).*masko(:,:,t);
end

%figure; imshow(AQ(:,:,1));
%figure; imshow(BG(:,:,1));
%AQ(1:10,1:10,1)
%BG(1:10,1:10,1)
%pause

%'AQsum'
AQsum = sum(sum(sum(AQ)));
BGsum = sum(sum(sum(BG)));
%'BGsum'


%Mask images are summed to get relative contributions.
%'AQrel'
AQrel = sum(sum(sum((maska-1)/(gain-1))));
%'BGrel'
BGrel = sum(sum(sum(masko)));



Saq = zeros(1); Sbg = zeros(1);

%Divide the sum of the line sums by the number of pixels to get
%intensity per pixel.
```

```
Saq = AQsum/AQrel;
Sbg = BGsum/BGrel;
S = Saq - Sbg;

%toc
```

Matlab Code
ZincDrops.m

```
%
% This code is designed to use 'DropLock10.m' in order to analyze zinc
% secretion measurements made with the LSM 5Live using a droplet sampling
% microfluidic device.
%
% The number of files is defined by how many sections the user splits the
% video into with ImageJ.  The fastest method has been empirically
% determined (on a fast computer) to be 3 files of 8000 frames each,
% totaling the 24000 frames collected over 10 minutes (40 Hz).  We also
% suggest a gain of at least 1000.
%
% An example of running this code is below:
%
% fulldata = ZincDrops('i1_11mMG_0', 3, 1000);
%
%
%
% Inputs:   1) 'base filename'
%           2) number of files to load
%           3) gain of droplet lock-in
%
% Outputs:  1) data set for fluorescent detection of zinc secretions
%

function[fulldata] = ...
    ZincDrops(base, files, gain)



clear fulldata


for m = 1:files

status = ['starting file ' num2str(m) ' of ' num2str(files)]

  clear green red

  fid = fopen([base '_red' num2str(m) '.raw'],'r');
  tic; red = fread(fid,'uchar'); toc
  fid = fopen([base '_green' num2str(m) '.raw'],'r');
  tic; green = fread(fid,'uchar'); toc
```

```
  tic;

  for n = 1:600/files
    [Saq, Sbg, S] = DropLock10(green, red, 107, 23, 40, (n-1)*40, gain);
    dataset(n,:,m) = [Saq; Sbg; S];
       if rem(n,50)<1
          n
       else
       end
  end;

clear green red

toc

end


for x = 1:m
  fulldata(1+(x-1)*n:x*n,:)=dataset(:,:,x);
end

save([base '_g' num2str(gain) '.txt'], 'fulldata', '-ascii', '-double', '-tabs');

plot(fulldata(:,3));
```