function [header, time, xyz, light, button, prop\_val] = binread(filename, varargin)

% BINREAD Reads GENEActive .bin files

%

% [hdr, time, xyz, light, but] = binread(fname)

% [hdr, time, xyz, light, but, prop\_val] = read(fname, 'key1', 'key2',...)

%

% Where

%

% FNAME is the file name

%

% HDR is a Mx1 cell array containing M header pages (each of them a struct)

%

% TIME is an Nx1 vector of measurement times. The times are expressed as

% serial date numbers (see help datenum)

%

% XYZ is a Nx3 matrix of calibrated accelerometer measurements. The columns

% correspond to the x, y and z axes

%

% LIGHT is a Nx1 vector of calibrated light measurements

%

% BUT is a Nx1 vector of button status values (1 on / 0 off)

%

% 'key1', 'key2' are names of page properties that should be extracted (and

% interpolated) from each data page. For instance

%

%

% (c) German Gomez-Herrero

% german.gomezherrero@ieee.org

% Some constants

DATA\_PAGE\_NAME = 'Recorded Data';

NB\_HEADER\_PAGES = 7;

NB\_DATA\_PAGES = 100;

CALIBRATION\_PAGE\_NAME = 'Calibration Data';

TIME\_NAME = 'Page Time';

TIME\_FORMAT = 'yyyy-mm-dd HH:MM:SS:FFF';

DATA\_PROPS = {'Battery voltage', 'Temperature'};

INTERPOLATE\_PROPS = true;

MEASUREMENT\_FREQ\_NAME = 'Measurement Frequency';

if nargin < 2,

 data\_props = DATA\_PROPS;

else

 data\_props = varargin;

end

fid = fopen(filename, 'r');

% Skip any blank line at the beginning of file

C = textscan(fid, '%[^\n]',1);

while isempty(C{1}),

 C = textscan(fid, '%[^\n]', 1);

end

% Read header pages

header = cell(NB\_HEADER\_PAGES, 1);

header\_page\_count = 1;

page\_name = C{1}{1};

while ~strcmpi(page\_name, DATA\_PAGE\_NAME),

 C = textscan(fid, '%[^\n:\*]: %[^\n]');

 header{header\_page\_count} = cell2struct(C{2}, ...

 strrep(C{1}(1:numel(C{2})), ' ', '\_'), 1);

 header{header\_page\_count}.Page\_Name = page\_name;

 if strcmpi(page\_name, CALIBRATION\_PAGE\_NAME),

 x\_gain = str2double(header{header\_page\_count}.x\_gain);

 y\_gain = str2double(header{header\_page\_count}.y\_gain);

 z\_gain = str2double(header{header\_page\_count}.z\_gain);

 x\_offset = str2double(header{header\_page\_count}.x\_offset);

 y\_offset = str2double(header{header\_page\_count}.y\_offset);

 z\_offset = str2double(header{header\_page\_count}.z\_offset);

 volts = str2double(header{header\_page\_count}.Volts);

 lux = str2double(header{header\_page\_count}.Lux);

 end

 if numel(C{2})<numel(C{1}),

 page\_name = C{1}{end};

 header\_page\_count = header\_page\_count + 1;

 else

 % We have reached the end of the file

 xyz = [];

 light = [];

 button = [];

 prop\_val = [];

 return;

 end

end

header(header\_page\_count+1:end) = [];

if isfield(header{end},'Number\_of\_Pages'),

 nb\_pages\_in\_header = true;

 nb\_pages = str2double(header{end}.Number\_of\_Pages);

else

 nb\_pages\_in\_header = false;

 nb\_pages = NB\_DATA\_PAGES;

end

% Read the data pages

data\_page\_count = 1;

page\_name = DATA\_PAGE\_NAME;

xyz = nan(300\*nb\_pages, 3);

light = nan(300\*nb\_pages, 1);

button = nan(300\*nb\_pages, 1);

prop\_val = nan(nb\_pages, length(data\_props));

time = nan(nb\_pages, 1);

freq = nan(nb\_pages, 1);

while strcmpi(page\_name, DATA\_PAGE\_NAME),

 C = textscan(fid, '%[^\n:\*]: %[^\n]');

 if numel(C{1}) ~= numel(C{2})+1,

 error('Invalid format in %dth data page', data\_page\_count);

 end

 % Get the numeric properties of that the user wants to get

 [prop\_idx, prop\_loc] = ismember(C{1}(1:end-1), data\_props);

 [prop\_loc, idx] = sort(prop\_loc(prop\_idx));

 prop\_idx = find(prop\_idx);

 prop\_idx = prop\_idx(idx);

 prop\_val(data\_page\_count, prop\_loc) = str2double(C{2}(prop\_idx));

 % Get the measurement time

 time(data\_page\_count) = datenum(C{2}(ismember(C{1}(1:end-1), TIME\_NAME)), ...

 TIME\_FORMAT);

 % Get the measurement frequency

 freq(data\_page\_count) = str2double(C{2}(ismember(C{1}(1:end-1), ...

 MEASUREMENT\_FREQ\_NAME)));

 % Get the measurements

 meas\_idx = (data\_page\_count-1)\*300+1:(data\_page\_count\*300);

 [xyz(meas\_idx,:), light(meas\_idx), button(meas\_idx)] = hex2xyz(C{1}{end});

 page\_name = textscan(fid, '%[^\n]',1);

 if ~isempty(page\_name{1}),

 page\_name = page\_name{1};

 data\_page\_count = data\_page\_count + 1;

 else

 page\_name = '';

 end

end

if ~isempty(page\_name),

 warning('binread:unknownPageName', 'Unknown page name %s', page\_name);

end

if nb\_pages\_in\_header && data\_page\_count ~= nb\_pages,

 warning('binread:unknownPageName', ...

 'Only %d data pages were found although %d pages are annotated in the header', ...

 data\_page\_count, nb\_pages);

end

% Interpolate the time

if any(diff(freq)),

 error('Not implemented yet');

else

 secs = 300/freq(1);

 msecs = round((secs-floor(secs))\*1e3);

 secs = floor(secs);

 time\_end = addtodate(addtodate(time(1), secs, 'second'), ...

 msecs, 'millisecond');

 offset = linspace(0, time\_end-time(1), 300);

 time\_interp = repmat(time(:), 1, 300) + repmat(offset, numel(time), 1);

 time\_interp = time\_interp';

 time\_interp = time\_interp(:);

end

% Intepolate the selected page properties

if INTERPOLATE\_PROPS

 prop\_val\_interp = nan(numel(time\_interp), size(prop\_val, 2));

 for i = 1:size(prop\_val, 2)

 prop\_val\_interp(:, i) = interp1(time, prop\_val(:,i), time\_interp, 'spline');

 end

 prop\_val = prop\_val\_interp;

end

time = time\_interp;

% Calibrate the data

xyz = (xyz\*100 - repmat([x\_offset, y\_offset, z\_offset], ...

 data\_page\_count\*300, 1))./repmat([x\_gain, y\_gain, z\_gain], ...

 data\_page\_count\*300, 1);

light = floor(light\*lux/volts);

end

function [xyz, light, button] = hex2xyz(hstr)

% Hexadecimal to decimal conversion of data values

n\_bytes = floor(numel(hstr)/2);

n\_meas = n\_bytes/6;

hstr = reshape(hstr(1:n\_bytes\*2), 2, n\_bytes)';

bin\_values = dec2bin(hex2dec(hstr))';

bin\_values = reshape(bin\_values, 1, n\_bytes\*8);

idx = repmat((1:48:48\*n\_meas)', 1, 12) + repmat(0:11, n\_meas, 1);

x = tc2dec(bin\_values(idx),12);

y = tc2dec(bin\_values(idx+12),12);

z = tc2dec(bin\_values(idx+24),12);

idx = repmat((37:48:48\*n\_meas)', 1, 10) + repmat(0:9, n\_meas, 1);

light = bin2dec(bin\_values(idx));

button = bin\_values((47:48:48\*n\_meas)')=='1';

f = bin\_values((48:48:48\*n\_meas)')=='1';

if any(f),

 error('The (f) field is not zero!');

end

xyz = [x(:),y(:),z(:)];

button = button(:);

light = light(:);

end

function value = tc2dec(bin,N)

% Two-complement to decimal conversion

val = bin2dec(bin);

y = sign(2^(N-1)-val).\*(2^(N-1)-abs(2^(N-1)-val));

value = y;

condition = (y==0 & val~=0);

value(condition) = -val(condition);

end