Freescale Semiconductor
Application Note

Document Number: AN4995
Rev 0, 01/2015

MPC5775K Twiddle Factor Generator User Guide

by: Curt Hillier and Maik Brett

Contents

1 Introduction

Twiddle factors are complex number constants used when recursively combining results from smaller discrete Fourier T ransforms in the Fast Fourier Transform (FFT) calculation process. The term 'twiddle factor' was first seen in publication in 1966 in the paper "Fast Fourier Transforms – for fun and profit," written by W.M. Gentleman and G. Sande[1]. Since FFTs have been in use since the 1960s, there are a number of papers and algorithms in existence explaining twiddle factor calculation. In this application note, we discuss the structure and use of a twiddle factor generator Matlab script and produce outputs in a format useful for programming the MPC5775K MCU on-chip FFT accelerators.

The twiddle factor generator described in this application note depends on the user specifying a filename, FFT size, and start address. It then generates twiddle factors and saves the factors in two formats:

- A C style header file with hex data format for use in customers' application software
- To a text file with decimal real and imaginary format.

The following components make up the Twiddle Factor Generator:

- Matlab file "twgen.m" for use with 2014 release of Matlab and later. Accepts user inputs for filename, FFT size, and start address. Generates twiddle factors and saves to output files.

- Text Output file (<filename>.twd). Contains twiddle factors in real and imaginary text format:

  <address> <Real part> <Imaginary part>where address is the physical Twiddle memory address, real part is the real portion of the twiddle entry, and imaginary part is the imaginary portion of the twiddle entry.

- Header Output file (<filename>.h). Contains output of the Matlab script. Format is in an unsigned array of 32-bit values compatible with C programs.

2 Using the software

Follow the below steps to use the software:

1. Load the ‘twgen.m’ file into Matlab
2. Click on the Go button to start the software
3. In the following dialog box, enter the size and start address of the FFT. For example, enter size = 512 and Start Address = 0x4000.

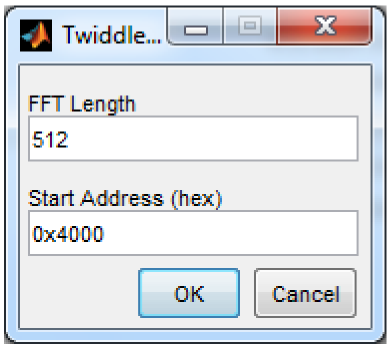

Figure 1. Size and start address

4. Next, the Matlab script will ask you to create an output *.twd file. Type in the name of the output file as shown below, then click on the Save button.

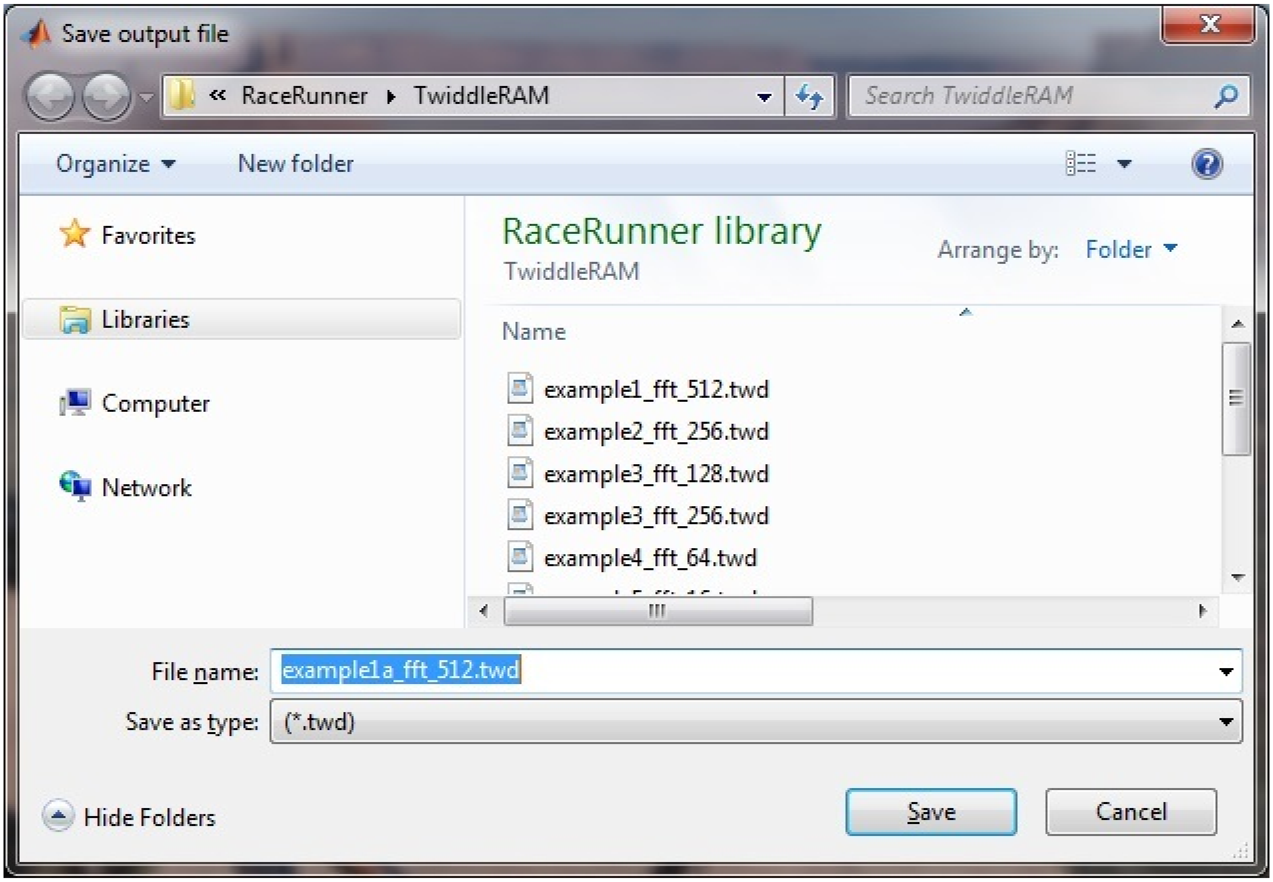

Figure 2. Save output file

3 Example output

The Matlab script will now generate twiddle factors and store them into a C style array of unsigned 32-bit integers into a <filename>.h file. For example, the 512 FFT sized output is shown below:

```
/*Twiddle RAM values for FFT length 512*/
/*for use with 16bit complex PMDA transfer*/
/*tw[k].im, tw[k].re, tw[k+1].im, tw[k+1].re*/
const unsigned long fft_twd512[] = {
  0xfe6e7ffd,   // 1st twiddle factor, 1st entry
  0xfe6e7ffd,   // 1st twiddle factor, 2nd entry
  0xfe6e7ffd,   // 1st twiddle factor, 3rd entry
  0xfe6e7ffd,   // 1st twiddle factor, 4th entry
  0xfe6e7ffd,   // 1st twiddle factor, 5th entry
  0xfe6e7ffd,   // 1st twiddle factor, 6th entry
  0xfe6e7ffd,   // 1st twiddle factor, 7th entry
  0xfe6e7ffd,   // 1st twiddle factor, 8th entry
  0xfcdc7ff5,   // 2nd twiddle factor, 1st entry
  0xfcdc7ff5,   // 2nd twiddle factor, 2nd entry
```

...

In addition, the Matlab script produces a text file, "filename.twd" containing the following information:

```
#Twiddle RAM values for FFT length 512
0x4000: 32765 +i -402
0x4001: 32765 +i -402
0x4002: 32765 +i -402
```

```
0x4003: 32765 +i -402
0x4004: 32765 +i -402
0x4005: 32765 +i -402
0x4006: 32765 +i -402
0x4007: 32765 +i -402
0x4008: 32757 +i -804
0x4009: 32757 +i -804
…
```

4 Twiddle RAM

The twiddle RAM holds constants like coefficients, which are used during some operations. It is also organized in slices of 8 to enable parallel access to 8 coefficients simultaneously and can be initialized with DMA operations.

Twiddle factors are calculated based on the following equation:

$$W_N^k = e^{-j2\pi k/N}$$

In the Matlab script, the following code generates the twiddle values:

```
      % Calculate twiddle factors, real and imaginary in floating point format

  for n = 1:fft_len/8

  fl_re = int_scale * cos(-2*pi*n/fft_len);
  fl_im = int_scale * sin(-2*pi*n/fft_len);

  cpx_twd(n) = round(fl_re) +i * round(fl_im);
end;
```

5 Matlab source code

The Matlab source code is listed below:

```
% C FSL 31.03.2014 - M.Brett
% C FSL 19.08.2014 - C.Hillier
% Generates twiddle for use with SPT
% to be used with quadrature extension (1/2 quadrant only stored)
% 19.08.2014 updates include change for unsiged integer 32 storing
%            and changing uigetfile to uiputfile for easing new file
%            creation.

clear;

fft_len_sup = {'16','32','64','128','256','512','1024','2048','4096'};
int_scale = (2^15)-1;

%ask the user to input the FFT length and starting address
while 1

    ans = inputdlg({'FFT Length', 'Start Address (hex)'}, 'Twiddle Gen', 1, {'256',
'0x4000'} );

    fft_len=str2num(ans{1});
    twd_adr = sscanf(ans{2}, '0x%x');
```

```
    fft_len_chk = strcmp(ans{1}, fft_len_sup);

    if sum( fft_len_chk ) > 0
        break;
    end

end

% Open a dialog to create an output file.  The same base filename will be
% used for both the .twd file and the .h file
[fname, fdir] = uiputfile('*.twd', 'Save output file');

fnm_twd = strcat(fdir,fname)
fp_twd = fopen(fnm_twd, 'w');

if (fp_twd < 0)
   errordlg(strcat('Cannot open file',fnm_twd), 'Error');
   return;
end

[fnm_spl, pos] =  regexp(fnm_twd,'\.','split');

fnm_mem = strcat(fnm_spl{1}, '.h')

fp_mem = fopen(fnm_mem, 'w');

if (fp_mem < 0)
   errordlg(strcat('Cannot open file',fnm_mem), 'Error');
   return;
end

% Calculate twiddle factors, real and imaginary in floating point format

for n = 1:fft_len/8

  fl_re = int_scale * cos(-2*pi*n/fft_len);
  fl_im = int_scale * sin(-2*pi*n/fft_len);

  cpx_twd(n) = round(fl_re) +i * round(fl_im);
end;

% convert floating point to 16 bit integer
adr_offs = 0;

int_twd = int32([int16(real(cpx_twd)); int16(imag(cpx_twd))]);

% convert to unsigned 16 bit integers
for n = 1:fft_len/8
   for k = 1:2
        if (int_twd(k, n) < 0)
            tc_twd(k, n) = 2^16 + int_twd(k, n) ;
        else
            tc_twd(k, n) = int_twd(k, n);
        end
    end
end

%Write twiddle factors to the *.twd file
fprintf(fp_twd, '#Twiddle RAM values for FFT length %d\n', fft_len);
for n = 1:fft_len/8

    for k = 1:8
        fprintf(fp_twd, '0x%x: %d +i %d\n', twd_adr + adr_offs, int_twd(1, n), int_twd(2,
n) );
        adr_offs = adr_offs + 1;
    end
end

%Write twiddle factors to the *.h file
fprintf(fp_mem, '/*Twiddle RAM values for FFT length %d*/\n', fft_len);
```

```
fprintf(fp_mem, '/*for use with 16bit complex PMDA transfer*/\n');
fprintf(fp_mem, '/*tw[k].im, tw[k].re, tw[k+1].im, tw[k+1].re*/\n');
%fprintf(fp_mem, 'const unsigned long long fft_twd%d[] = {\n', fft_len);
fprintf(fp_mem, 'const unsigned long fft_twd%d[] = {\n', fft_len);

for n = 1:fft_len/8
%    for k = 1:4
%        fprintf(fp_mem, '  0x%04x%04x%04x%04x', tc_twd(1, n), tc_twd(2, n), tc_twd(1, n),
tc_twd(2, n) );
    for k = 1:8
        fprintf(fp_mem, '  0x%04x%04x', tc_twd(2, n), tc_twd(1, n) );

%        if not(k == 4 && n == fft_len/8 )
        if not(k == 8 && n == fft_len/8 )
            fprintf(fp_mem, ',');
        end;
        fprintf(fp_mem, '\n');
    end
end
fprintf(fp_mem, '};\n');

%Close files
fclose(fp_twd);
fclose(fp_mem);
```

6 References

1. W. M. Gentleman and G. Sande, “Fast Fourier transforms-for fun and profit,” 1966 Fall Joint Computer Conf., AFIPS Proc., vol. 29. Washington, D.C.: Spartan, 1966.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4995
Revision 0, 01/2015