Typical applications of an $N$-point radix-2 FFT accept $N x(n)$ input time samples and compute $N X(m)$ frequency-domain samples, where indices $n$ and $m$ both range from zero to $N-1$. However, there are non-standard FFT applications (for example, specialized harmonic analysis, or perhaps using an FFT to implement a bank of filters) where only a subset of the full $X(m)$ results are required.

Consider Figure 1(a) that shows the butterfly operations for an 8-point radix-2 decimation-in-time FFT. Assuming we are only interested in the $X(3)$ and $X(7)$ output samples, rather than compute the entire FFT we perform only the computations indicated by the bold lines in Figure 1(a). In order to compute only $X(3)$ and $X(7)$ we need to know the butterfly twiddle phase angle factors associated with the bold-line computations. Here we show how, and provide a Matlab routine, to compute the twiddle factors of $N$-point radix-2 FFTs.

Notice that the FFT butterflies in Figure 1(a) are single-complex-multiply butterflies. The numbers associated with the butterflies are phase angle factors, 'A', as shown in Figure 1(b). A butterfly's full twiddle factor is shown in Figure 1(c).

**Decimation-in-time FFT Twiddle Factors**

For the decimation-in-time (DIT) FFT using the single-complex multiply butterflies,

- The $N$-point DIT FFT has $\log_2(N)$ stages, numbered $P = 1, 2, \ldots, \log_2(N)$. 

---

**Figure 1:** (a) 8-point decimation-in-time (DIT) FFT signal flow diagram; (b) single-complex multiply DIT butterfly with angle factor $A$; (c) DIT butterfly details.
• Each stage contains \( N/2 \) butterflies.
• Not counting the \(-1\) multiply operations, the \( P \)th stage has \( N/2 \) twiddle factors, numbered \( k = 0, 1, 2, \ldots, N/2-1 \) as indicated by the upward arrows at the bottom of Figure 1(a).

Given those characteristics, the \( k \)th twiddle factor phase angle for the \( P \)th stage is computed using:

\[
k \text{th DIT twiddle factor angle} = \lfloor k \cdot 2^P / N \rfloor \text{bit-rev}
\]

(1)

where \( 0 \leq k \leq N/2-1 \). The \( \lfloor \cdot \rfloor \text{bit-rev} \) operation means the integer part of \( q \).

The \( \lfloor z \rfloor \text{bit-rev} \) function represents the three-step operation of: convert decimal integer \( z \) to a binary number represented by \( \log_2(N)-1 \) binary bits, perform bit reversal on the binary number as discussed in Section 4.5, and convert the bit reversed number back to a decimal integer.

As an example of using Eq.(1), for the second stage \( (P = 2) \) of an \( N = 8 \)-point DIT FFT, the \( k = 3 \) twiddle factor angle is:

\[
3 \text{rd twiddle factor angle} = \lfloor 3 \cdot 2^2 / 8 \rfloor \text{bit-rev}
\]

\[
= \lfloor 1.5 \rfloor \text{bit-rev} = \lfloor 1 \rfloor \text{bit-rev} = 2.
\]

The above \( \lfloor 1 \rfloor \text{bit-rev} \) operation is: take the decimal number 1 and represent it with \( \log_2(N)-1 = 2 \) bits, i.e., as 012. Next, reverse those bits to a binary 102 and convert that binary number to our desired decimal result of 2.

**Decimation-in-frequency FFT Twiddle Factors**

Figure 2(a) shows the butterfly operations for an 16-point radix-2 decimation-in-frequency FFT. As before, notice that the FFT butterflies in Figure 2(a) are single-complex-multiply butterflies. The numbers associated with the butterflies are phase angle factors, 'A', as shown in Figure 2(b). A butterfly's full twiddle factor is shown in Figure 2(c).

For the decimation-in-frequency (DIF) radix-2 FFT using the optimized butterflies,

• The \( N \)-point DIF FFT has \( \log_2(N) \) stages, numbered \( P = 1, 2, \ldots, \log_2(N) \).
• Each stage comprises \( N/2 \) butterflies.
• Not counting the \(-1\) twiddle factors, the \( P \)th stage has \( N/2^P \) unique twiddle factors, numbered \( k = 0, 1, 2, \ldots, N/2^P-1 \) as indicated by the upward arrows at the bottom of Figure 2.

Given those characteristics, the \( k \)th unique twiddle factor phase angle for the \( P \)th stage is computed using:

\[
k \text{th DIF twiddle factor angle} = k \cdot 2^P / 2
\]

(2)

where \( 0 \leq k \leq N/2^P-1 \). For example, for the second stage \( (P = 2) \) of an \( N = 8 \)-point DIF FFT, the unique twiddle factor angles are:

\[
\begin{align*}
  k = 0, & \quad \text{angle} = 0 \cdot 2^2 / 2 = 0 \cdot 4 / 2 = 0 \\
  k = 1, & \quad \text{angle} = 1 \cdot 2^2 / 2 = 1 \cdot 4 / 2 = 2.
\end{align*}
\]
In Figure 2(a) the final stages's single twiddle factor of zero means multiply by unity, i.e., no operation.

Closing Remarks:
Once you have the following Matlab code running on your computer, at tonight's dinner table you can proudly announce, "Family, ... may I have your attention?" [Wait for your family to stop making noise.] "You will be happy to know that I now have the ability to compute the twiddle factors of decimation-in-frequency fast Fourier transforms." Next, enjoy the loving smiles on the faces of your proud family.

Matlab Code:
Below is the Matlab code to find radix-2 FFT butterfly twiddle factors. The code computes the 'A' phase angle factors that are used in the twiddle factors as shown in Figure 1(c) and Figure 2(c). I suggest you start by running the code for the
8-point DIT FFT in Figure 1(a) and then run the code for the
16-point DIF FFT in Figure 2(a).

```
% Filename: FFT_Twiddles_Find_DSPrelated.m

% Computes 'Decimation in Frequency' or 'Decimation
% in Time' Butterfly twiddle factors, for radix-2 FFTs
% with in-order input indices and scrambled output indices.
% % To use, do two things: (1) define FFT size 'N'; and
% (2) define the desired 'Structure', near line 17-18,
% as 'Dec_in_Time' or 'Dec_in_Freq'.
% % Author: Richard Lyons, November, 2011
%******************************************

clear, clc

% Define input parameters
N = 8; % FFT size (Must be an integer power of 2)
Structure = 'Dec_in_Time'; % Choose Dec-in-time butterflies
%Structure = 'Dec_in_Freq'; % Choose Dec-in-frequency butterflies

% Start of processing
Num_Stages = log2(N); % Number of stages
StageStart = 1; % First stage to compute
StageStop = Num_Stages; % Last stage to compute
ButterStart = 1; % First butterfly to compute
ButterStop = N/2; % Last butterfly to compute
Pointer = 0; % Init 'results' row pointer
for Stage_Num = StageStart:StageStop
    if Structure == 'Dec_in_Time'
        for Butter_Num = ButterStart:ButterStop
            Twid = floor((2^Stage_Num*(Butter_Num-1))/N);
            % Compute bit reversal of Twid
            Twid_Bit_Rev = 0;
            for I = Num_Stages-2:-1:0
                if Twid>=2^I
                    Twid_Bit_Rev = Twid_Bit_Rev + 2^(Num_Stages-I-2);
                    Twid = Twid -2^I;
                end
                % End bit reversal 'I' loop
            end
            A1 = Twid_Bit_Rev; % Angle A1
            A2 = Twid_Bit_Rev + N/2; % Angle A2
            Pointer = Pointer +1;
            Results(Pointer,:) = [Stage_Num,Butter_Num,A1,A2];
        end
    else
        for Twiddle_Num = 1:N/2^Stage_Num
            Twid = (2^Stage_Num*(Twiddle_Num-1))/2; % Compute integer
            Pointer = Pointer +1;
            Results(Pointer,:) = [Stage_Num,Twiddle_Num,Twid];
        end
    end % End 'if'
end % End Stage_Num loop

Results(:,1:3), disp(' Stage#  Twid#   A'), disp(' ')
```