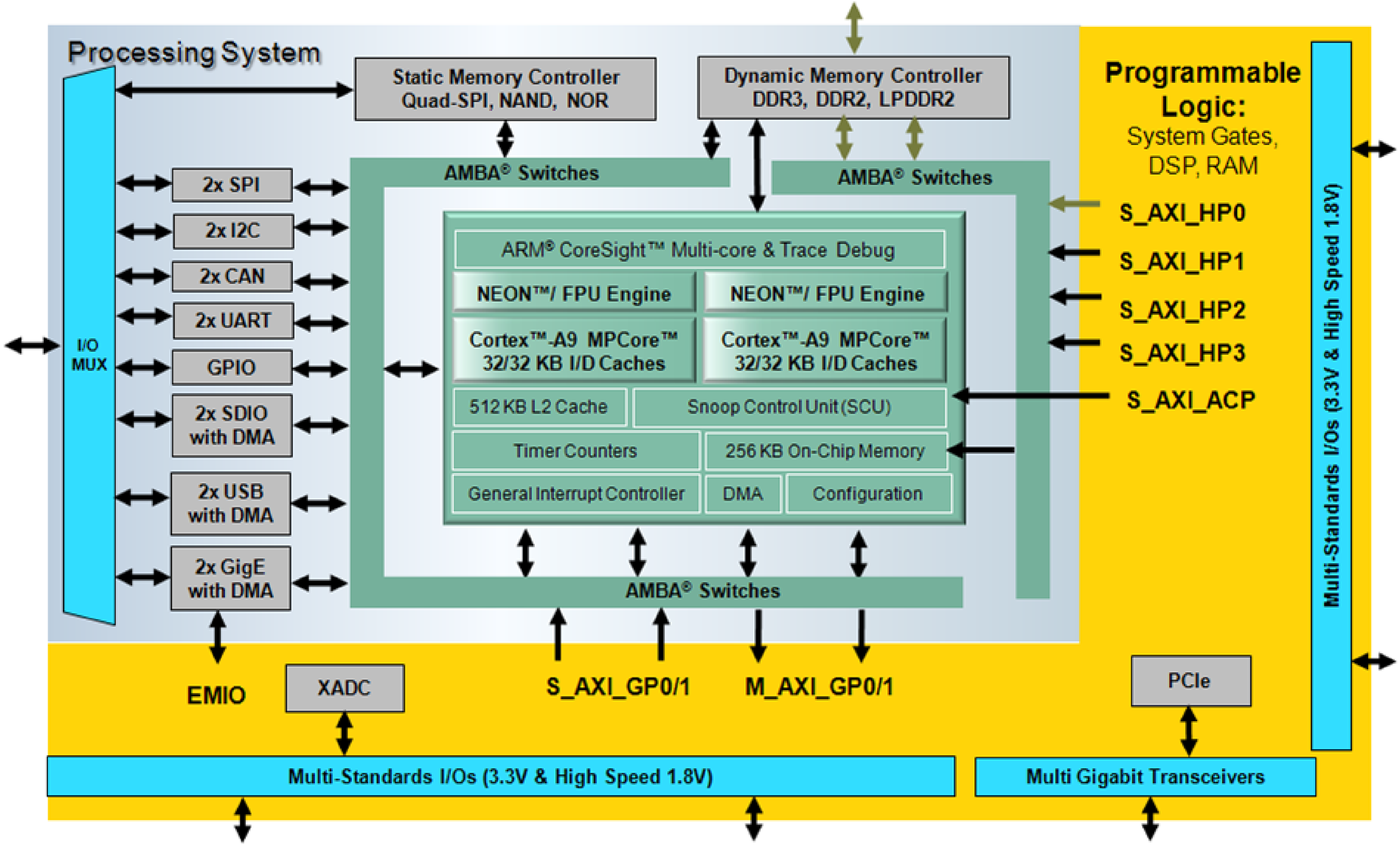
ELD Lab 9
FFT Using ARM Processor

Objective

- Implement 4-point FFT on ARM Cortex A9 processor of Zynq SoC
- Homework 1: Implement 8-point and 16-point FFT on ARM Cortex A9 processor of Zynq SoC

Theory & Lab

Zynq Architecture: PS and PL



Fourier Transform (FT)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi k n}{N} i}$$

- It is basically matrix vector multiplication

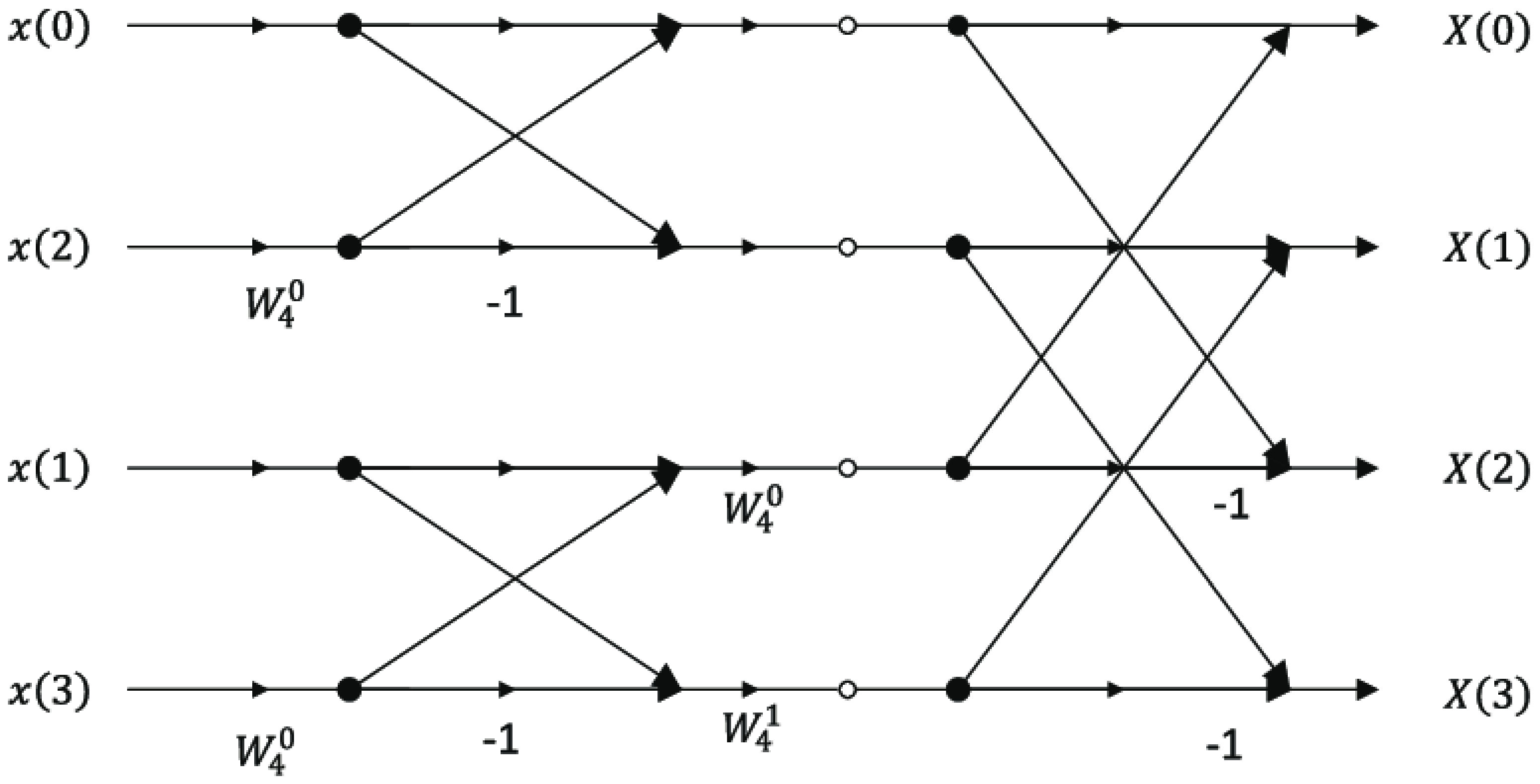
$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} e^{-\frac{2\pi 0.0}{4} i} & e^{-\frac{2\pi 0.1}{4} i} & e^{-\frac{2\pi 0.2}{4} i} & e^{-\frac{2\pi 0.3}{4} i} \\ e^{-\frac{2\pi 1.0}{4} i} & e^{-\frac{2\pi 1.1}{4} i} & e^{-\frac{2\pi 1.2}{4} i} & e^{-\frac{2\pi 1.3}{4} i} \\ e^{-\frac{2\pi 2.0}{4} i} & e^{-\frac{2\pi 2.1}{4} i} & e^{-\frac{2\pi 2.2}{4} i} & e^{-\frac{2\pi 2.3}{4} i} \\ e^{-\frac{2\pi 3.0}{4} i} & e^{-\frac{2\pi 3.1}{4} i} & e^{-\frac{2\pi 3.2}{4} i} & e^{-\frac{2\pi 3.3}{4} i} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

Fourier Transform (FT)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi k n}{N} i}$$

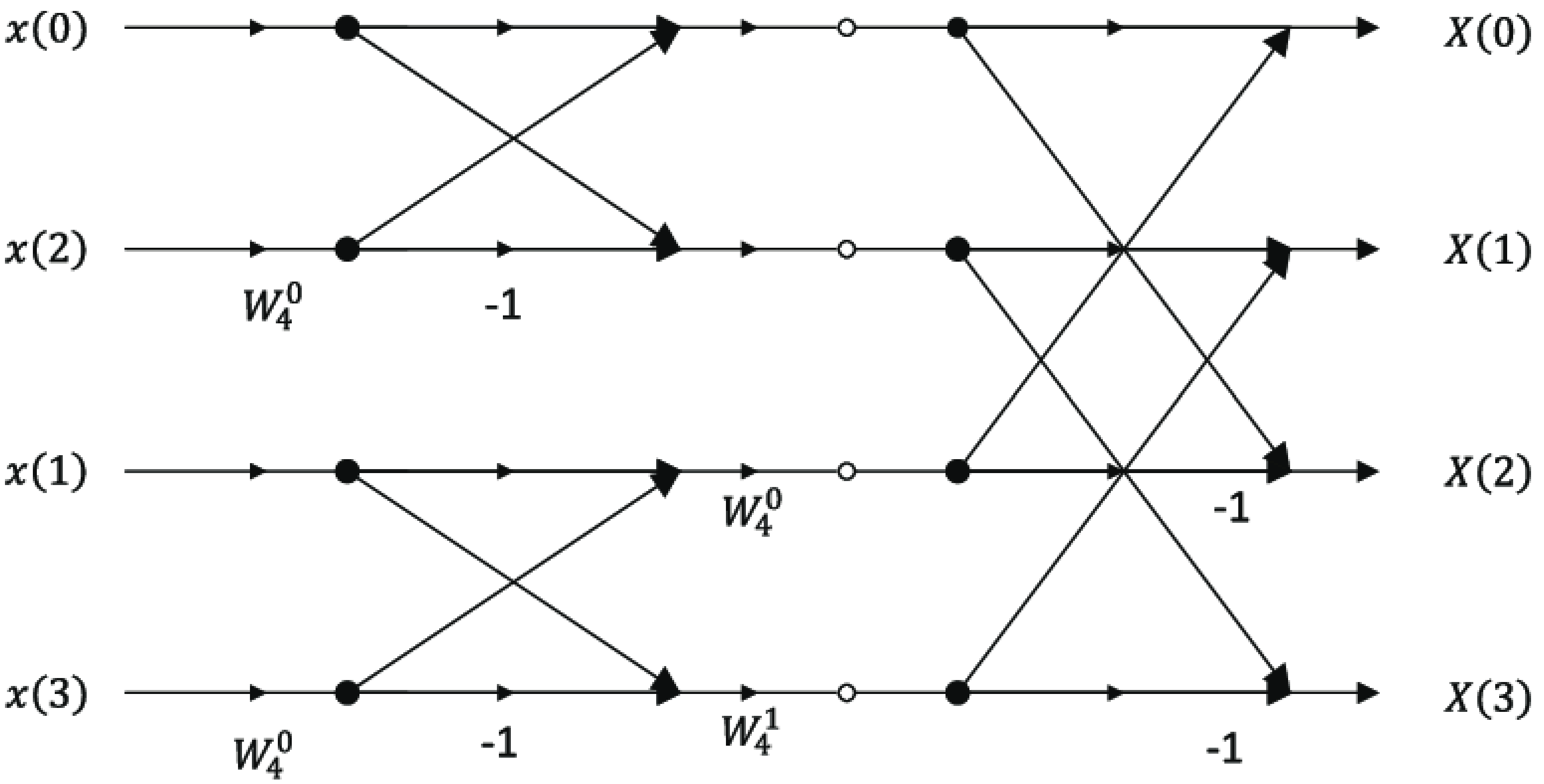
- It is basically matrix vector multiplication

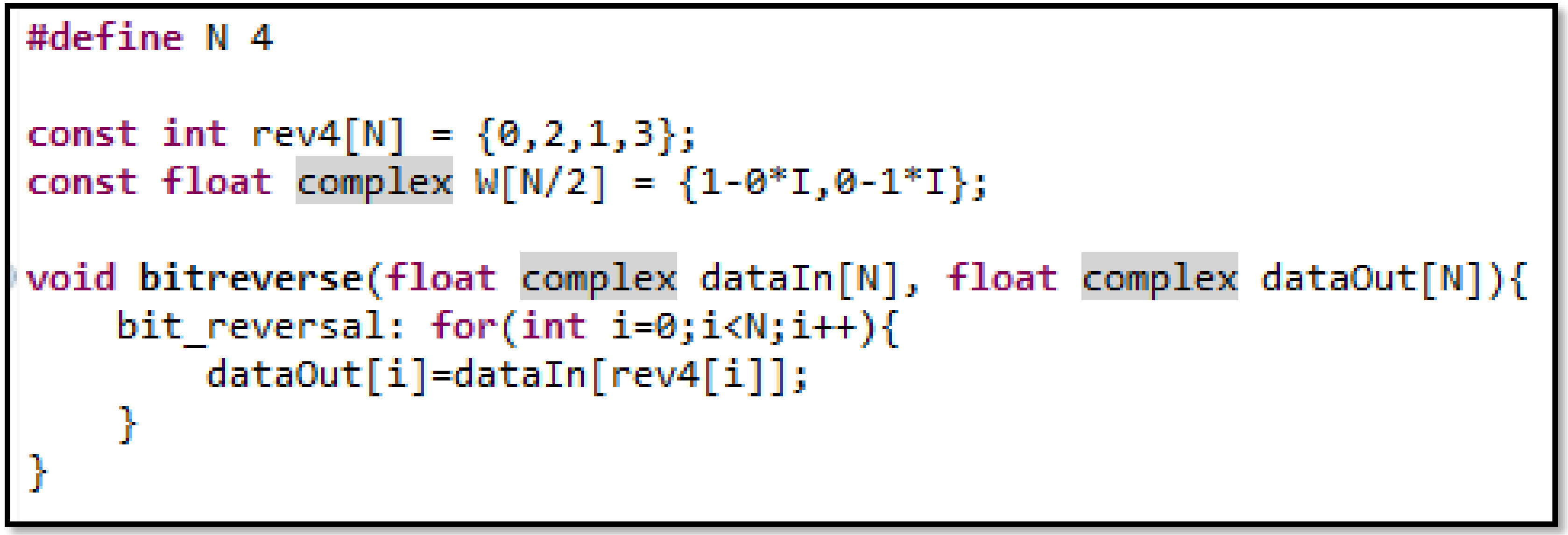
$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$



- $$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

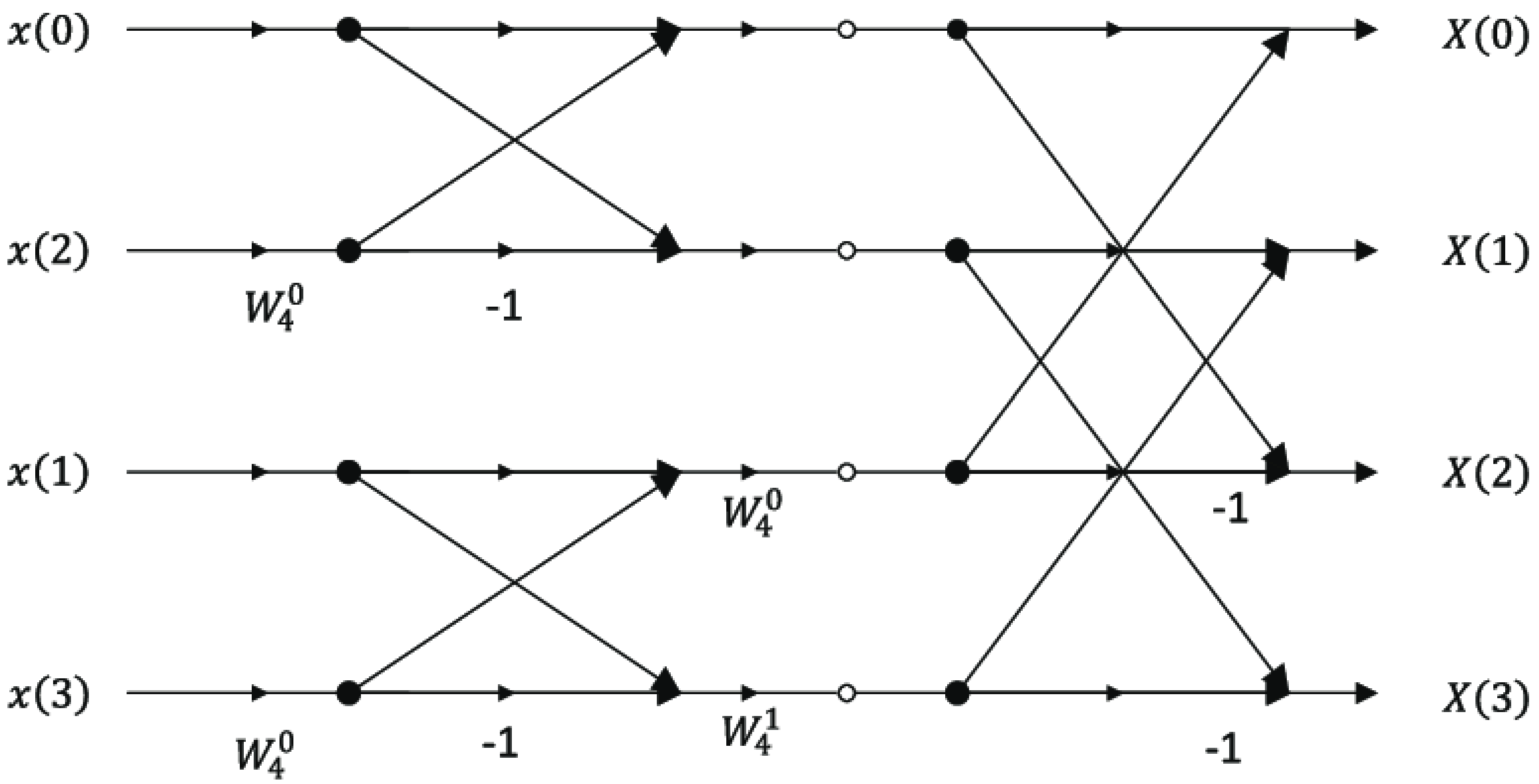
Step 1

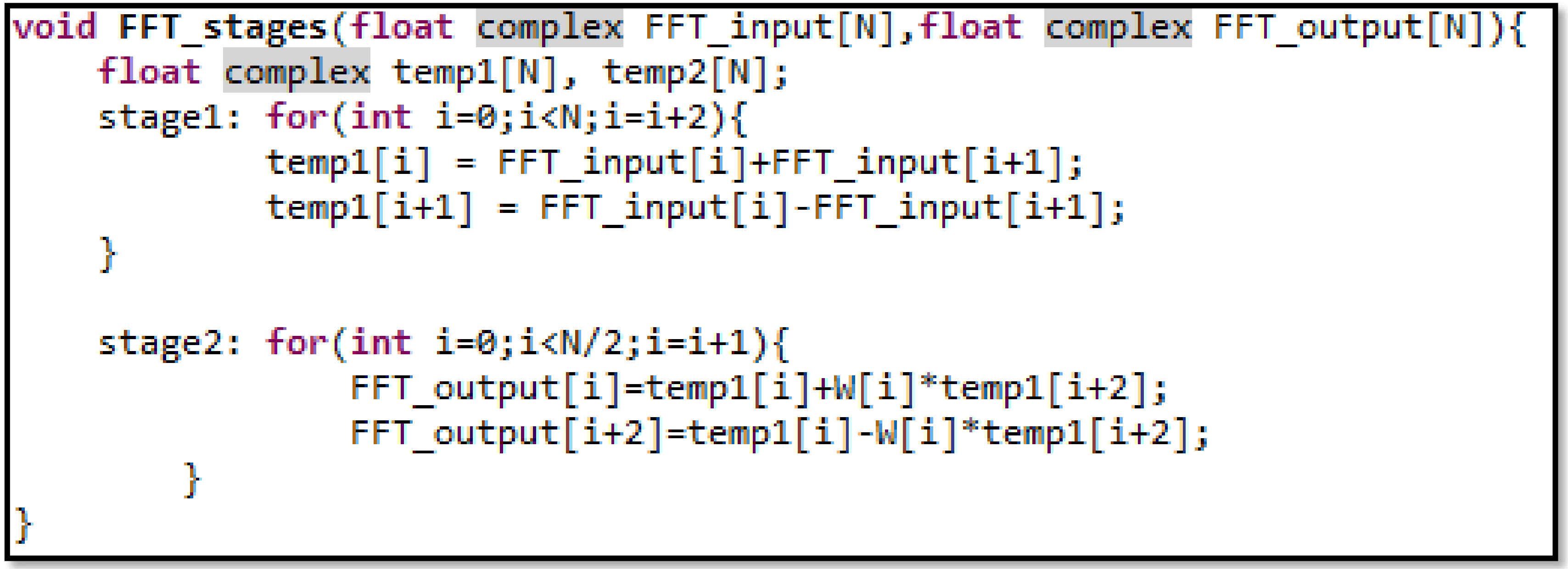




```
#define N 4

const int rev4[N] = {0,2,1,3};
const float complex W[N/2] = {1-0*I,0-1*I};

void bitreverse(float complex dataIn[N], float complex dataOut[N]){
    bit_reversal: for(int i=0;i<N;i++){
        dataOut[i]=dataIn[rev4[i]];
    }
}
```

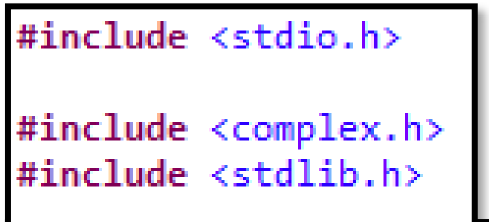
Step 2



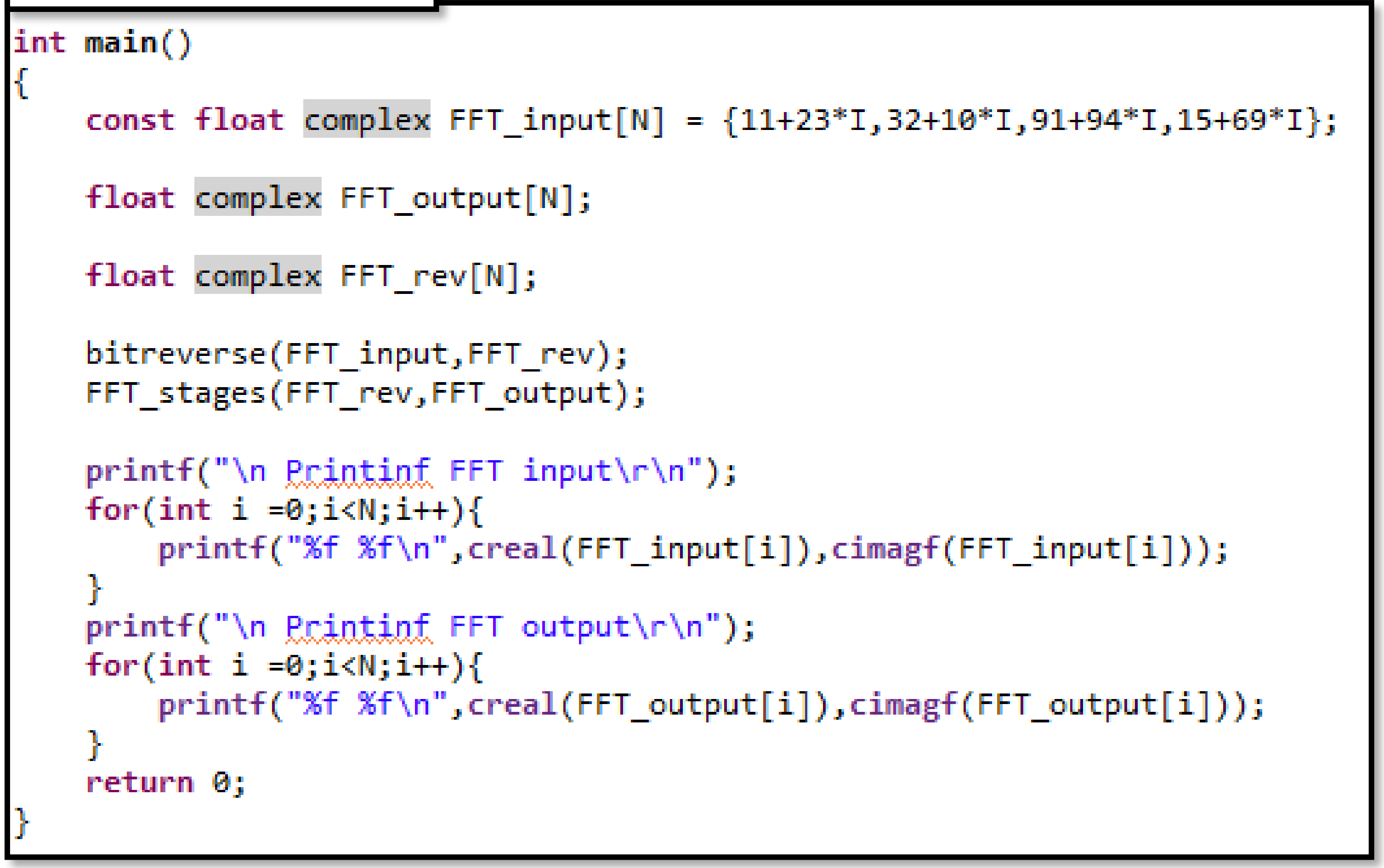


```
void FFT_stages(float complex FFT_input[N],float complex FFT_output[N]){
    float complex temp1[N], temp2[N];
    stage1: for(int i=0;i<N;i=i+2){
            temp1[i] = FFT_input[i]+FFT_input[i+1];
            temp1[i+1] = FFT_input[i]-FFT_input[i+1];
    }

    stage2: for(int i=0;i<N/2;i=i+1){
                FFT_output[i]=temp1[i]+W[i]*temp1[i+2];
                FFT_output[i+2]=temp1[i]-W[i]*temp1[i+2];
        }
}
```

Main Code

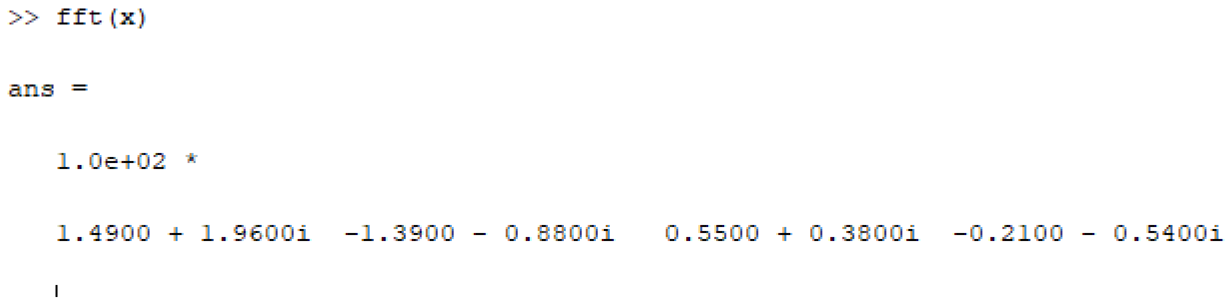


```c
#include <stdio.h>

#include <complex.h>
#include <stdlib.h>
```

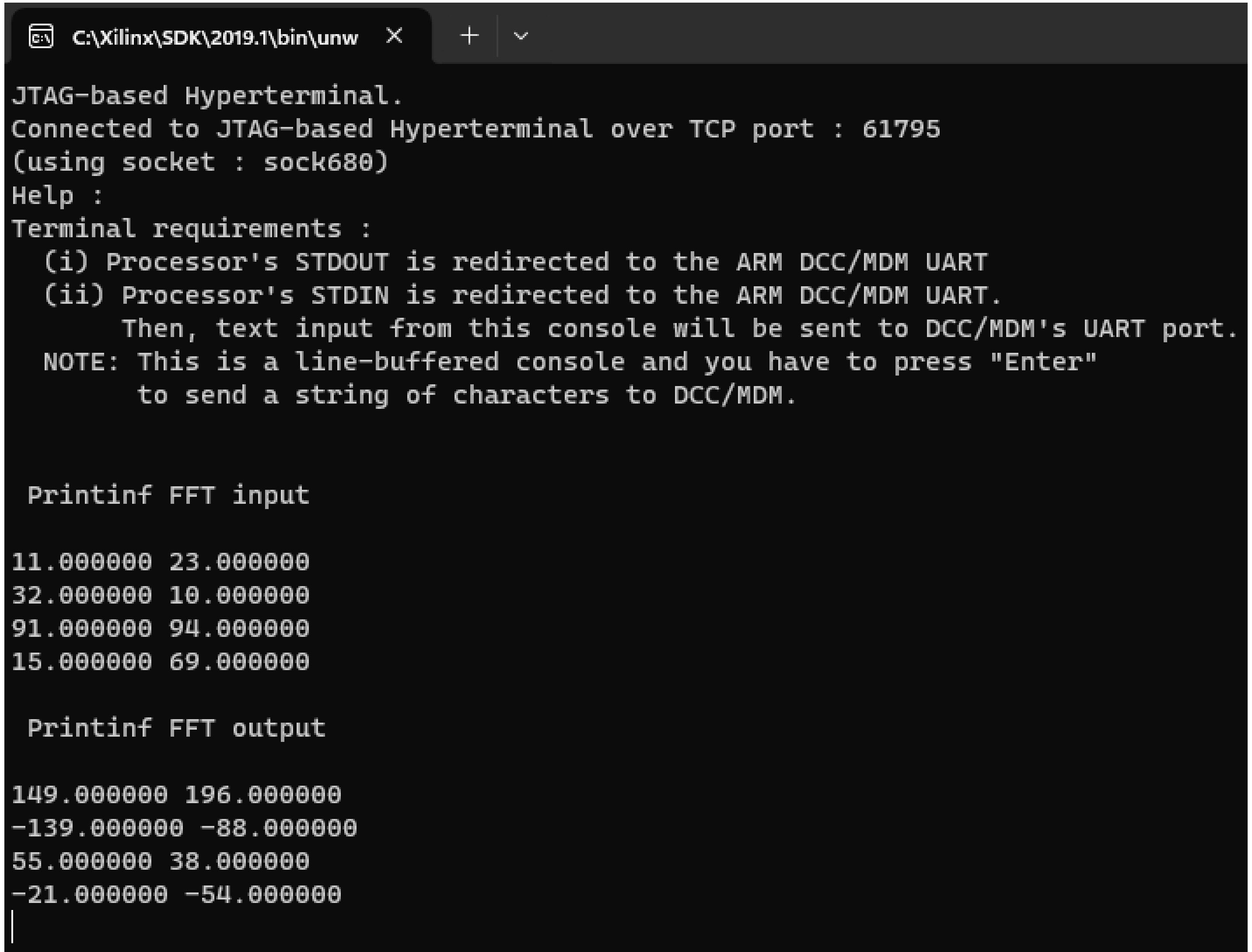


```c
int main()
{
    const float complex FFT_input[N] = {11+23*I,32+10*I,91+94*I,15+69*I};

    float complex FFT_output[N];

    float complex FFT_rev[N];

    bitreverse(FFT_input,FFT_rev);
    FFT_stages(FFT_rev,FFT_output);

    printf("\n Printinf FFT input\r\n");
    for(int i =0;i<N;i++){
        printf("%f %f\n",creal(FFT_input[i]),cimagf(FFT_input[i]));
    }
    printf("\n Printinf FFT output\r\n");
    for(int i =0;i<N;i++){
        printf("%f %f\n",creal(FFT_output[i]),cimagf(FFT_output[i]));
    }
    return 0;
}
```

Output



```
>> fft(x)

ans =

   1.0e+02 *

   1.4900 + 1.9600i  -1.3900 - 0.8800i   0.5500 + 0.3800i  -0.2100 - 0.5400i
```



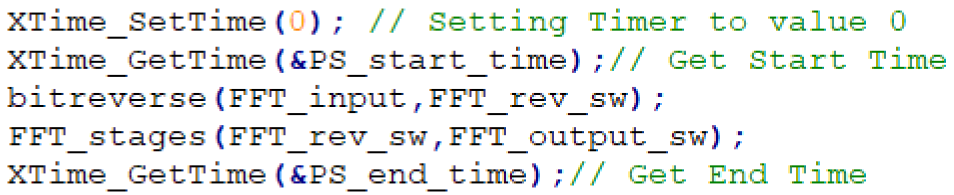
```
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 61795
(using socket : sock680)
Help :
Terminal requirements :
  (i) Processor's STDOUT is redirected to the ARM DCC/MDM UART
  (ii) Processor's STDIN is redirected to the ARM DCC/MDM UART.
       Then, text input from this console will be sent to DCC/MDM's UART port.
  NOTE: This is a line-buffered console and you have to press "Enter"
        to send a string of characters to DCC/MDM.


 Printinf FFT input

11.000000 23.000000
32.000000 10.000000
91.000000 94.000000
15.000000 69.000000

 Printinf FFT output

149.000000 196.000000
-139.000000 -88.000000
55.000000 38.000000
-21.000000 -54.000000
```

Execution Time Using Timer

- Include #include <xtime_l.h>
- Use two variables two store start and end time

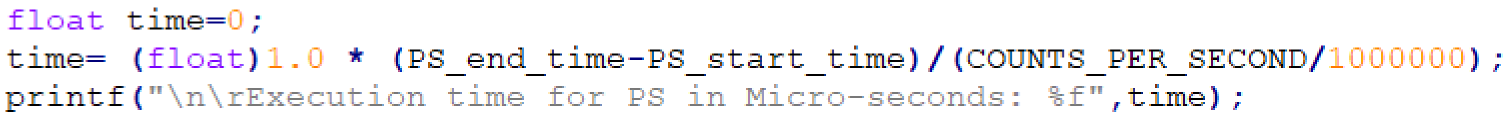
XTime PS_start_time,PS_end_time;

- Enable and disable the timer



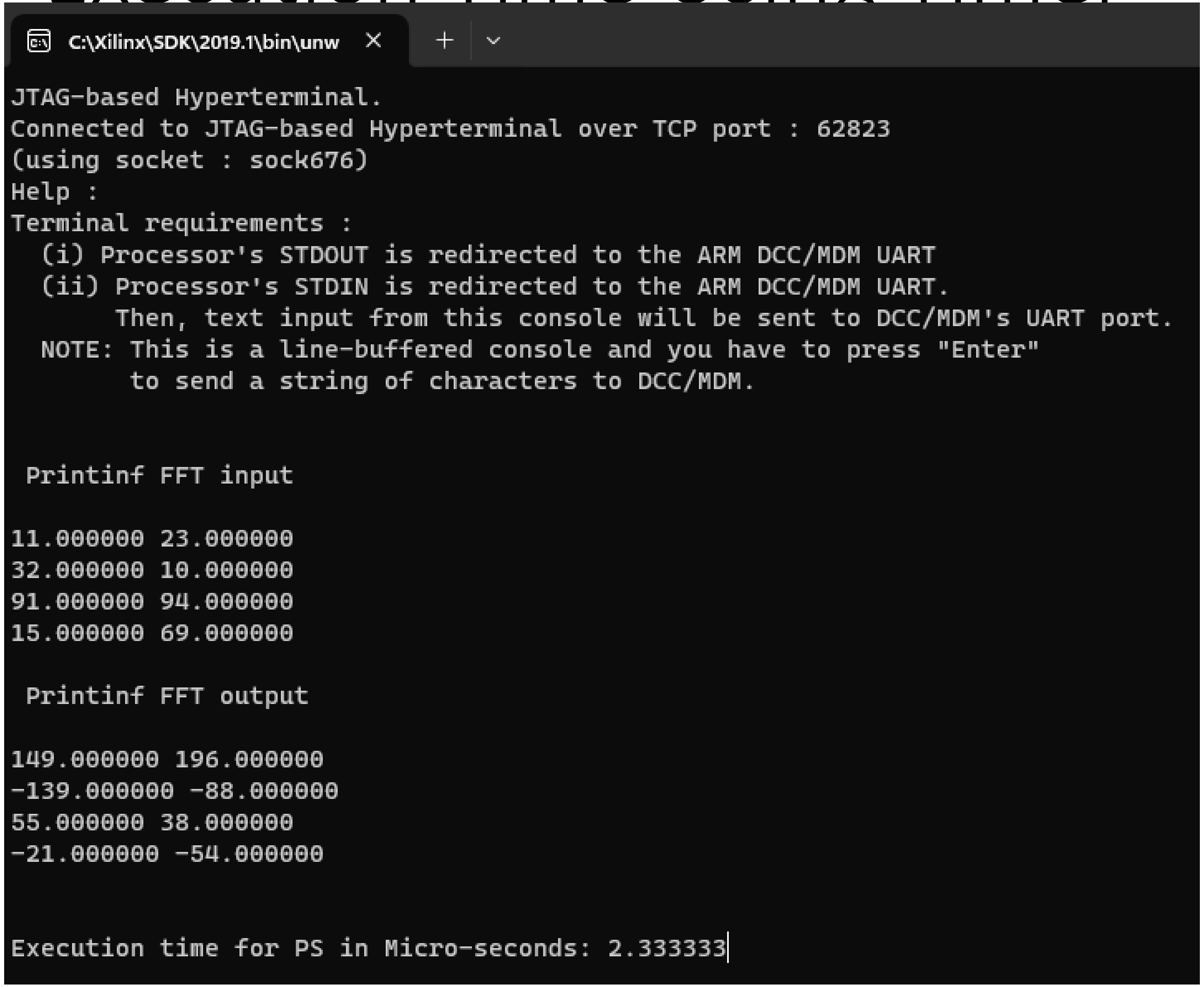
```
XTime_SetTime(0); // Setting Timer to value 0
XTime_GetTime(&PS_start_time);// Get Start Time
bitreverse(FFT_input,FFT_rev_sw);
FFT_stages(FFT_rev_sw,FFT_output_sw);
XTime_GetTime(&PS_end_time);// Get End Time
```

- Calculate execution time in Seconds



```
float time=0;
time= (float)1.0 * (PS_end_time-PS_start_time)/(COUNTS_PER_SECOND/1000000);
printf("\n\rExecution time for PS in Micro-seconds: %f",time);
```

Execution Time Using Timer



```
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 62823
(using socket : sock676)
Help :
Terminal requirements :
  (i) Processor's STDOUT is redirected to the ARM DCC/MDM UART
  (ii) Processor's STDIN is redirected to the ARM DCC/MDM UART.
       Then, text input from this console will be sent to DCC/MDM's UART port.
  NOTE: This is a line-buffered console and you have to press "Enter"
        to send a string of characters to DCC/MDM.


 Printinf FFT input

11.000000 23.000000
32.000000 10.000000
91.000000 94.000000
15.000000 69.000000

 Printinf FFT output

149.000000 196.000000
-139.000000 -88.000000
55.000000 38.000000
-21.000000 -54.000000


Execution time for PS in Micro-seconds: 2.333333
```