Worksheet for the afternoon course
“Tune measurements simulated with a DSP card”

CAS Egham, September 2017

D. Alves, S. Jackson, H. Schmickler

1. Introduction

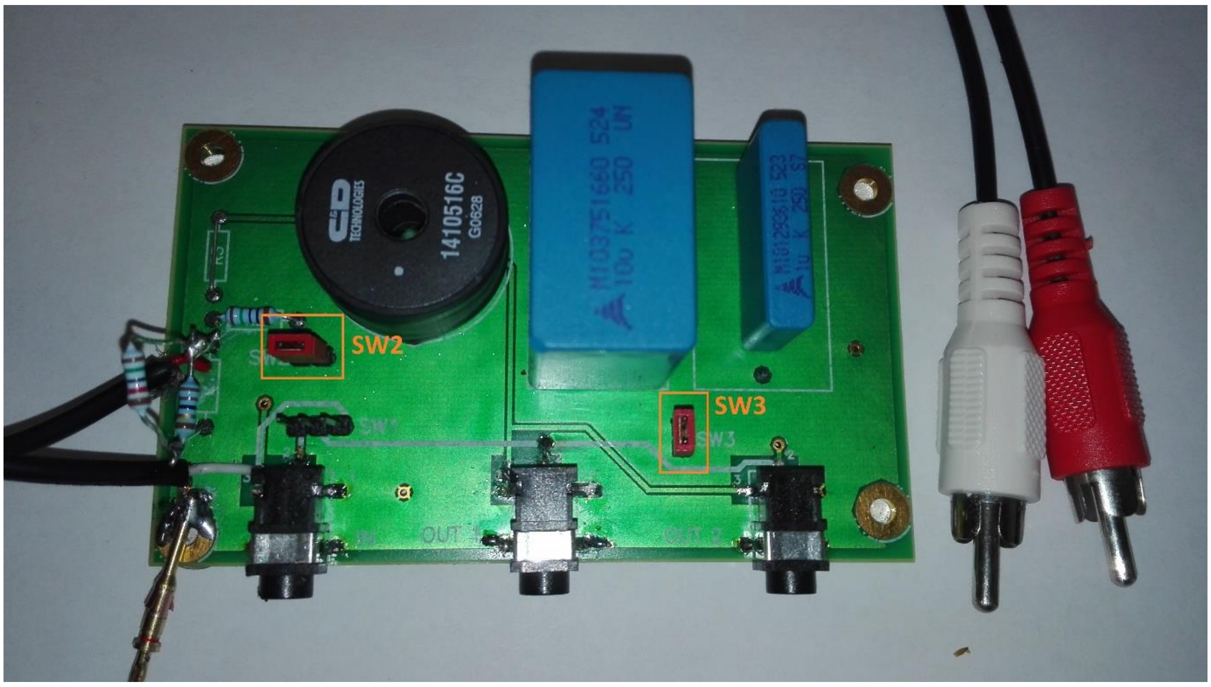
In this 3 h course we will be replacing the betatron oscillations of the beam in the horizontal and vertical plane by two parallel LC resonators. The basic learning objective of the course is:

- Understand the different ways of stimulating transverse beam oscillations in time and frequency domain.
- Understand the signals of these transverse oscillations collected from a beam position monitors in time and frequency domain
- Generate various excitation modes with the help of an evaluation board carrying an “Analog Devices SHARC DSP processor”. The same DSP is also used for a treatment and display of the resulting “beam oscillation”.
  The whole measurement sequence is programmed in C++, but the code is freely available and can be modified by skilled students.
- Generate a principle understanding of the necessary signal treatment for tune measurements and exercise the use of a simple two channel oscilloscope.

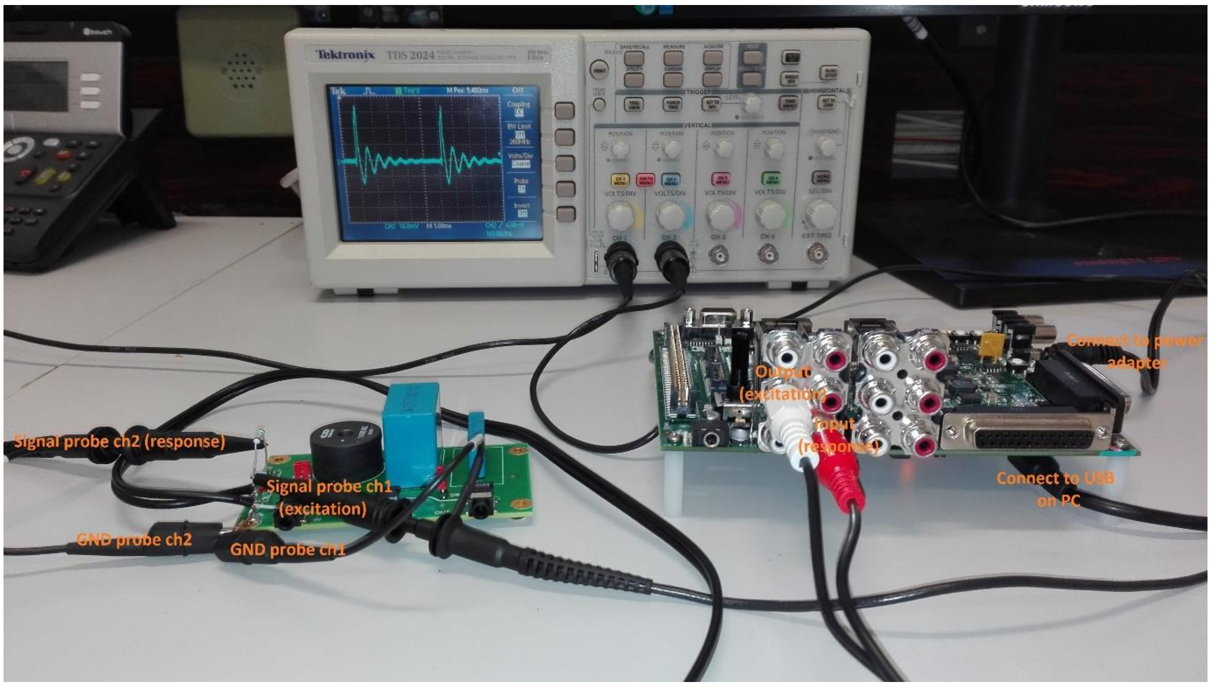
2. Description of the setup

Below the complete schematic of the two LC resonators representing the “beam”. The schematic has been generated with a freeware simulation tool:
http://www.linear.com/designtools/software/#LTspice

With one jumper (SW3) on the beam PCB you can change the resonant frequency by adding one capacitor in parallel. Also you can put into another jumper (SW2) in order to put a resistor in parallel to the LC resonator. This emulates different damping times of the beam (i.e. chromaticity). During all exercises you can manipulate these jumpers and look at the resulting differences.
Please locate all components on the board…if you know a little of electronics you can read the values of the resistances from their colour code.

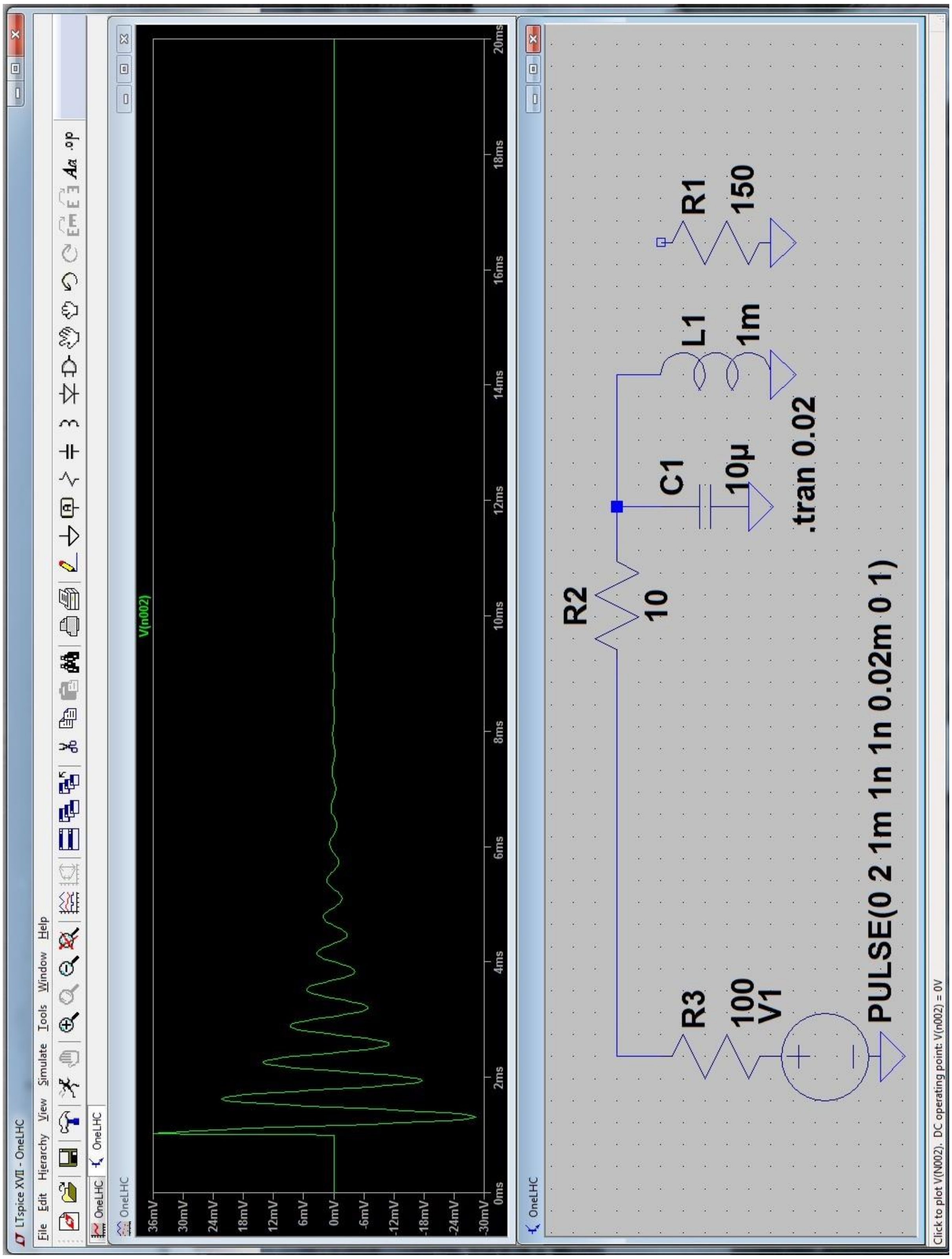


Both, the tool and the schematics are on the indico website of the CAS course. I recommend to install this on the private laptop. This way every experiment can also be simulated and the results compared to real measurements. Some of the simulation snapshots will also be included in this document.



LTspice simulation:

The snapshot below shows the schematic (one beam) and its response for an excitation with a single pulse of 20 us length (exercise 3).



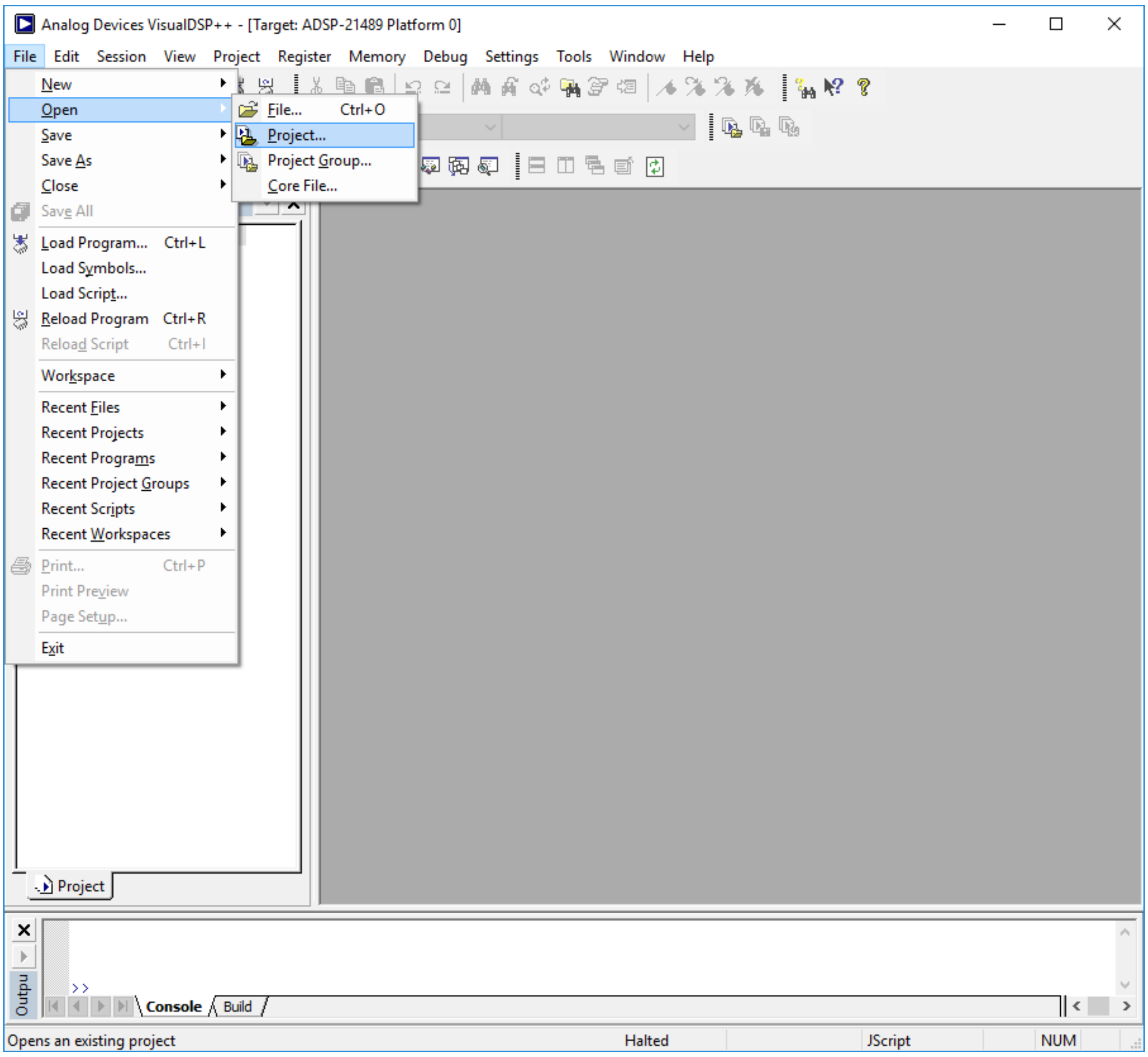
Click to plot V(N002). DC operating point: V(n002) = 0V

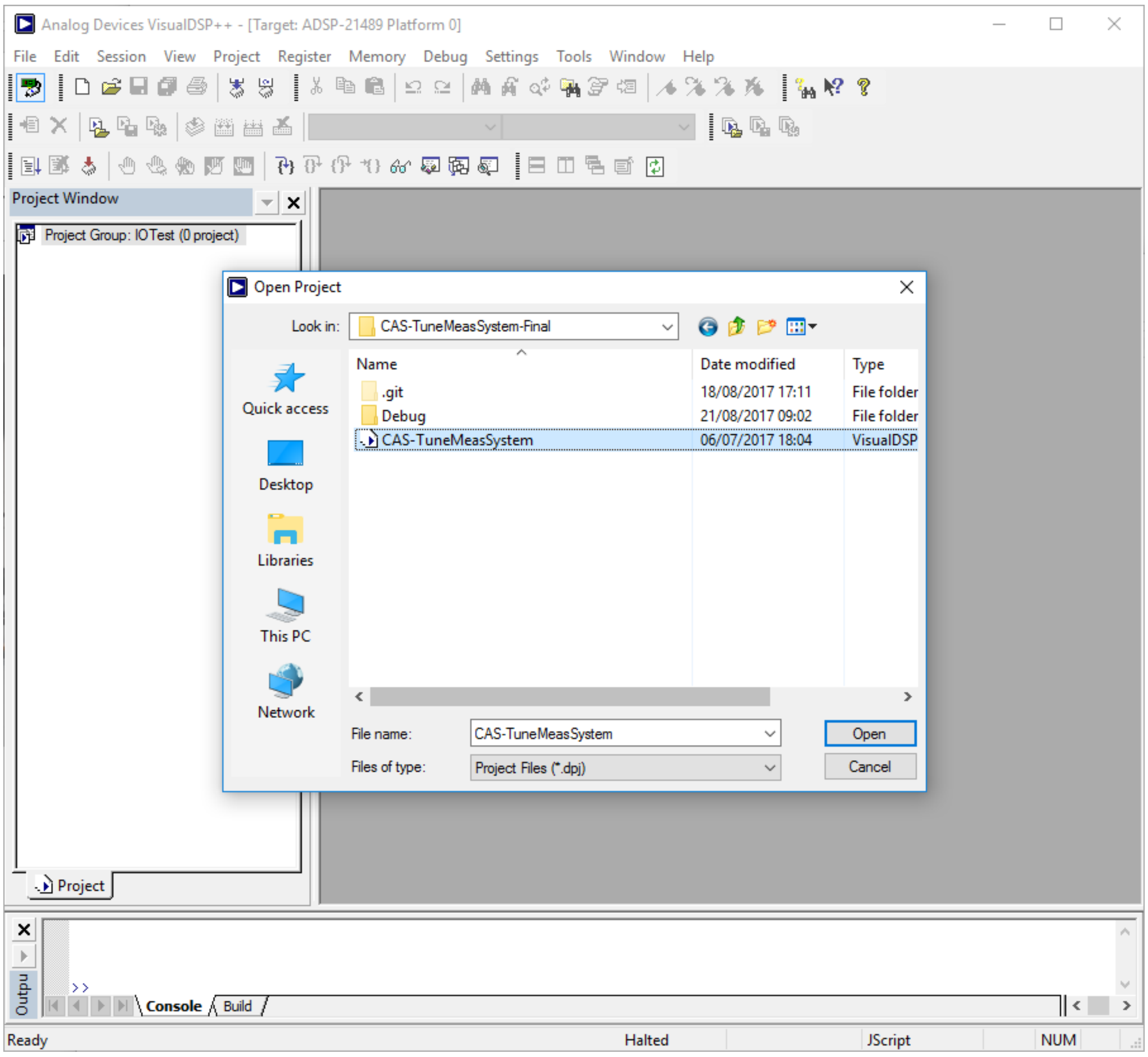
3. The DSP development environment

The DSP development environment is preinstalled on the note-books, which are linked by a USB cable to the evaluation kit.

After startup first the course project needs to be opened:

- Download zip file with project from: https://indico.cern.ch/event/509762/contributions/2450804/attachments/1487281/2355803/CAS-TuneMeasSystem-Final.zip
- Unzip file to Desktop
- Import project into VisualDSP++





Project Group: IOTest (0 project)

Look in: CAS-TuneMeasSystem-Final

Name	Date modified	Type
.git	18/08/2017 17:11	File folder
Debug	21/08/2017 09:02	File folder
CAS-TuneMeasSystem	06/07/2017 18:04	VisualDSP

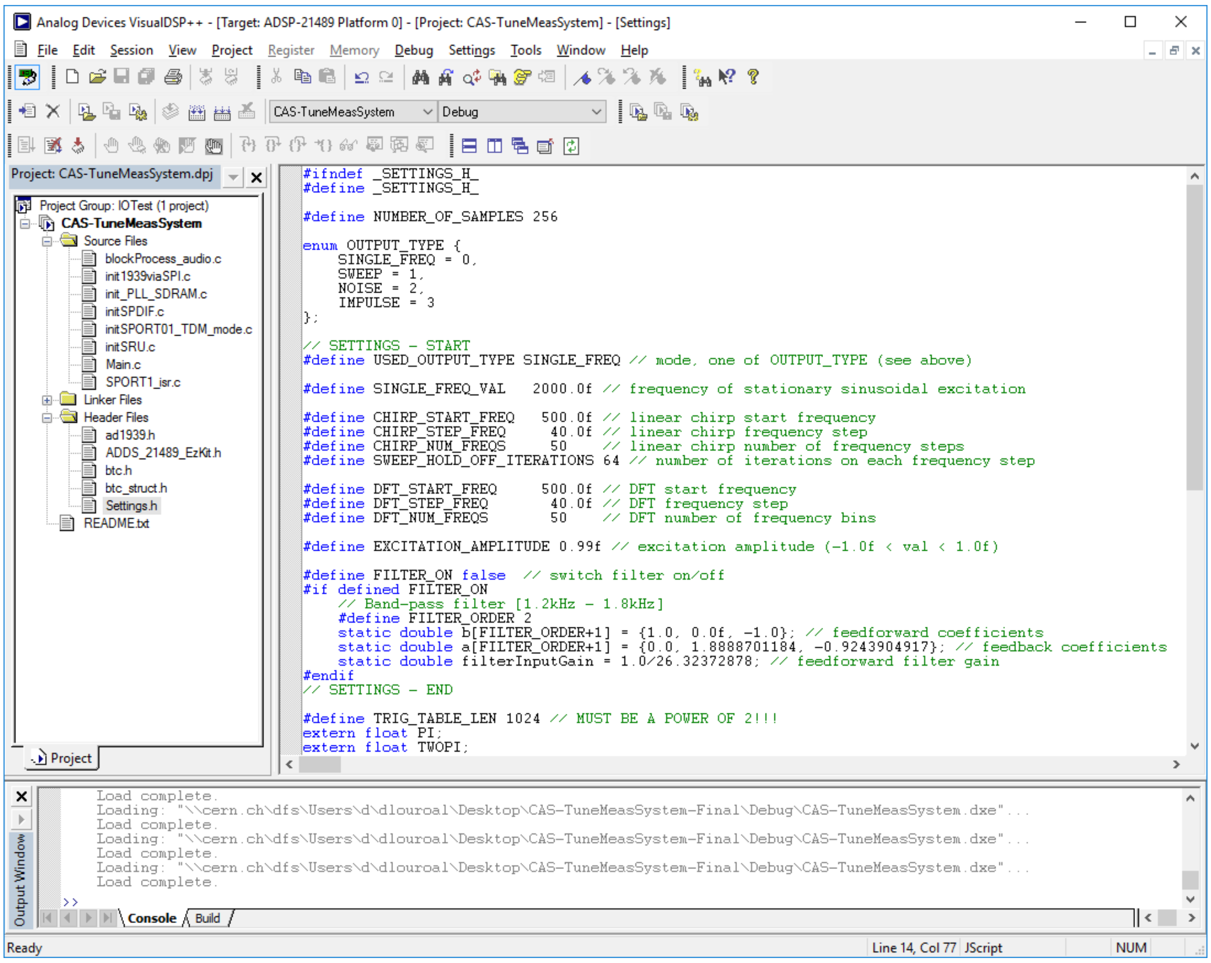
File name: CAS-TuneMeasSystem

Files of type: Project Files (*.dpj)

The software is not written with a graphical user interface. It is a single program, which can do all steps of the exercise, but for each step the program has to be recompiled after one has modified a parameter file, which configures the program for each step.

The parameter file is called “Settings.h” and the following picture shows part of its content. The meaning of each parameter shall be described in each exercise.

Configuration file: Settings.h



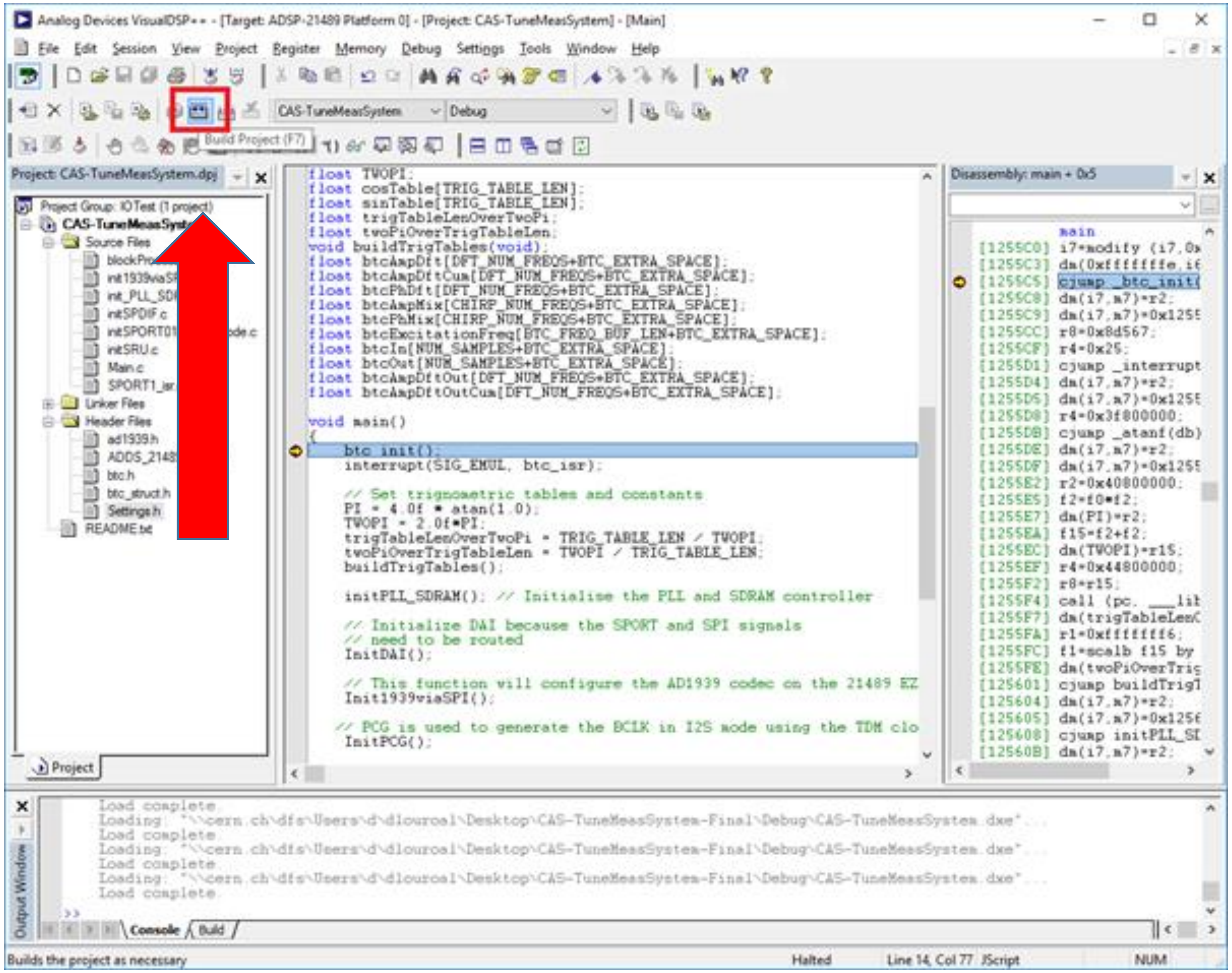
```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   2000.0f // frequency of stationary sinusoidal excitation

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each frequency step

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1.0f)

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coefficients
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; // feedback coefficients
    static double filterInputGain = 1.0/26.32372878; // feedforward filter gain
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
    Load complete.
    Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
    Load complete.
    Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
    Load complete.
    Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
    Load complete.
>>
```

DSP environment, compiling…..

The following screen shots explain the various steps of invoking the program, compiling the code and loading the executable code into the DSP for execution.
More explanations will be given in the course.

- Build project (first time you build it after importing):



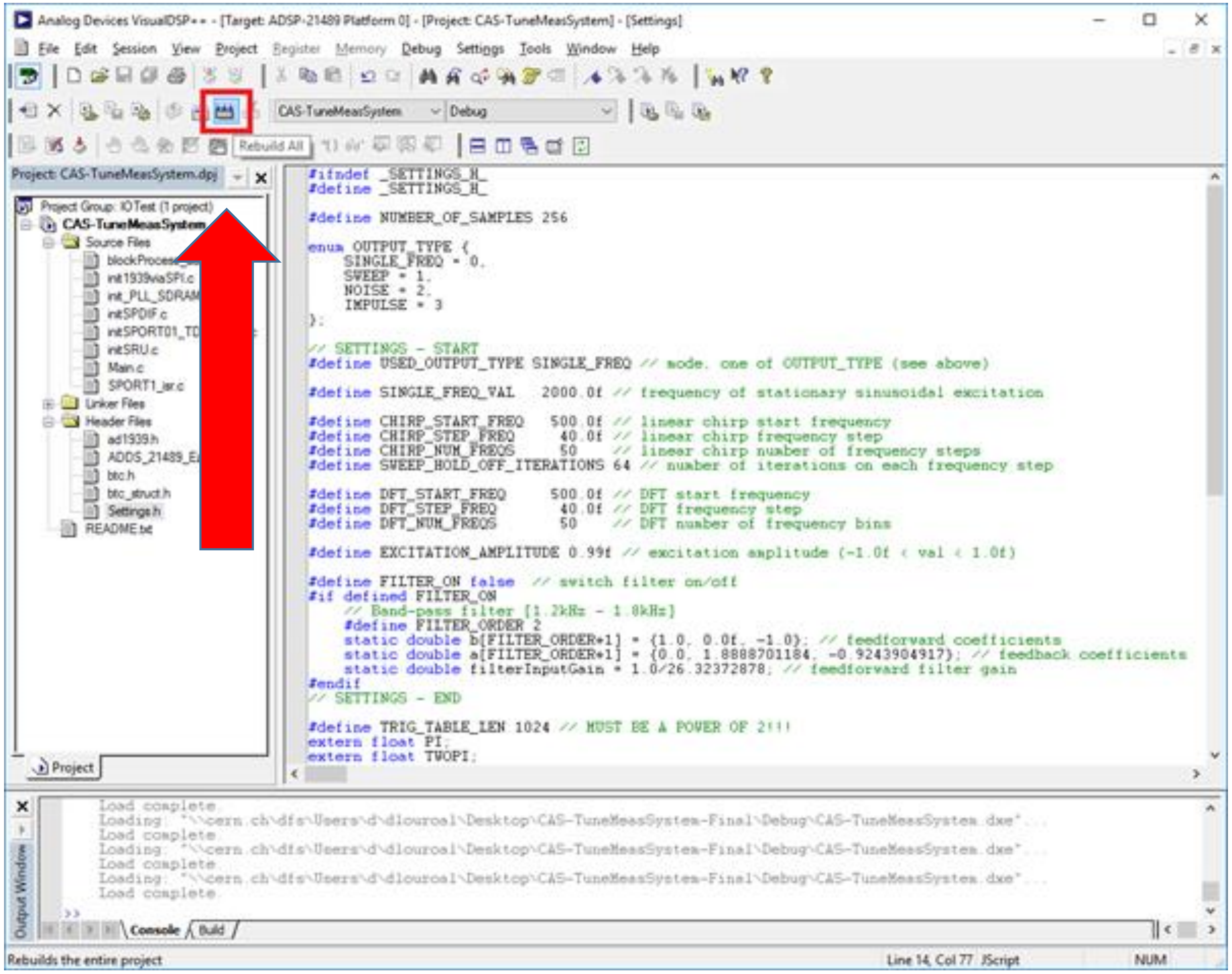
```
float TWOPI;
float cosTable[TRIG_TABLE_LEN];
float sinTable[TRIG_TABLE_LEN];
float trigTableLenOverTwoPi;
float twoPiOverTrigTableLen;
void buildTrigTables(void);
float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
float btcExcitationFreq[BTC_FREQ_BUF_LEN+BTC_EXTRA_SPACE];
float btcIn[NUM_SAMPLES+BTC_EXTRA_SPACE];
float btcOut[NUM_SAMPLES+BTC_EXTRA_SPACE];
float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

void main()
{
    btc_init();
    interrupt(SIG_EMUL, btc_isr);

    // Set trignometric tables and constants
    PI = 4.0f * atan(1.0);
    TWOPI = 2.0f*PI;
    trigTableLenOverTwoPi = TRIG_TABLE_LEN / TWOPI;
    twoPiOverTrigTableLen = TWOPI / TRIG_TABLE_LEN;
    buildTrigTables();

    initPLL_SDRAM(); // Initialise the PLL and SDRAM controller

    // Initialize DAI because the SPORT and SPI signals
    // need to be routed
    InitDAI();

    // This function will configure the AD1939 codec on the 21489 EZ
    Init1939viaSPI();

   // PCG is used to generate the BCLK in I2S mode using the TDM clo
    InitPCG();
```

```
          main
[1255C0] i7=modify (i7,0x
[1255C3] dm(0xfffffffe,i6
[1255C5] cjump _btc_init(
[1255C8] dm(i7,m7)=r2;
[1255C9] dm(i7,m7)=0x1255
[1255CC] r8=0x8d567;
[1255CF] r4=0x25;
[1255D1] cjump _interrupt
[1255D4] dm(i7,m7)=r2;
[1255D5] dm(i7,m7)=0x1255
[1255D8] r4=0x3f800000;
[1255DB] cjump _atanf(db)
[1255DE] dm(i7,m7)=r2;
[1255DF] dm(i7,m7)=0x1255
[1255E2] r2=0x40800000;
[1255E5] f2=f0*f2;
[1255E7] dm(PI)=r2;
[1255EA] f15=f2+f2;
[1255EC] dm(TWOPI)=r15;
[1255EF] r4=0x44800000;
[1255F2] r8=r15;
[1255F4] call (pc, ___lib
[1255F7] dm(trigTableLenO
[1255FA] r1=0xfffffff6;
[1255FC] f1=scalb f15 by
[1255FE] dm(twoPiOverTrig
[125601] cjump buildTrigT
[125604] dm(i7,m7)=r2;
[125605] dm(i7,m7)=0x1256
[125608] cjump initPLL_SI
[12560B] dm(i7,m7)=r2;
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
>>
```

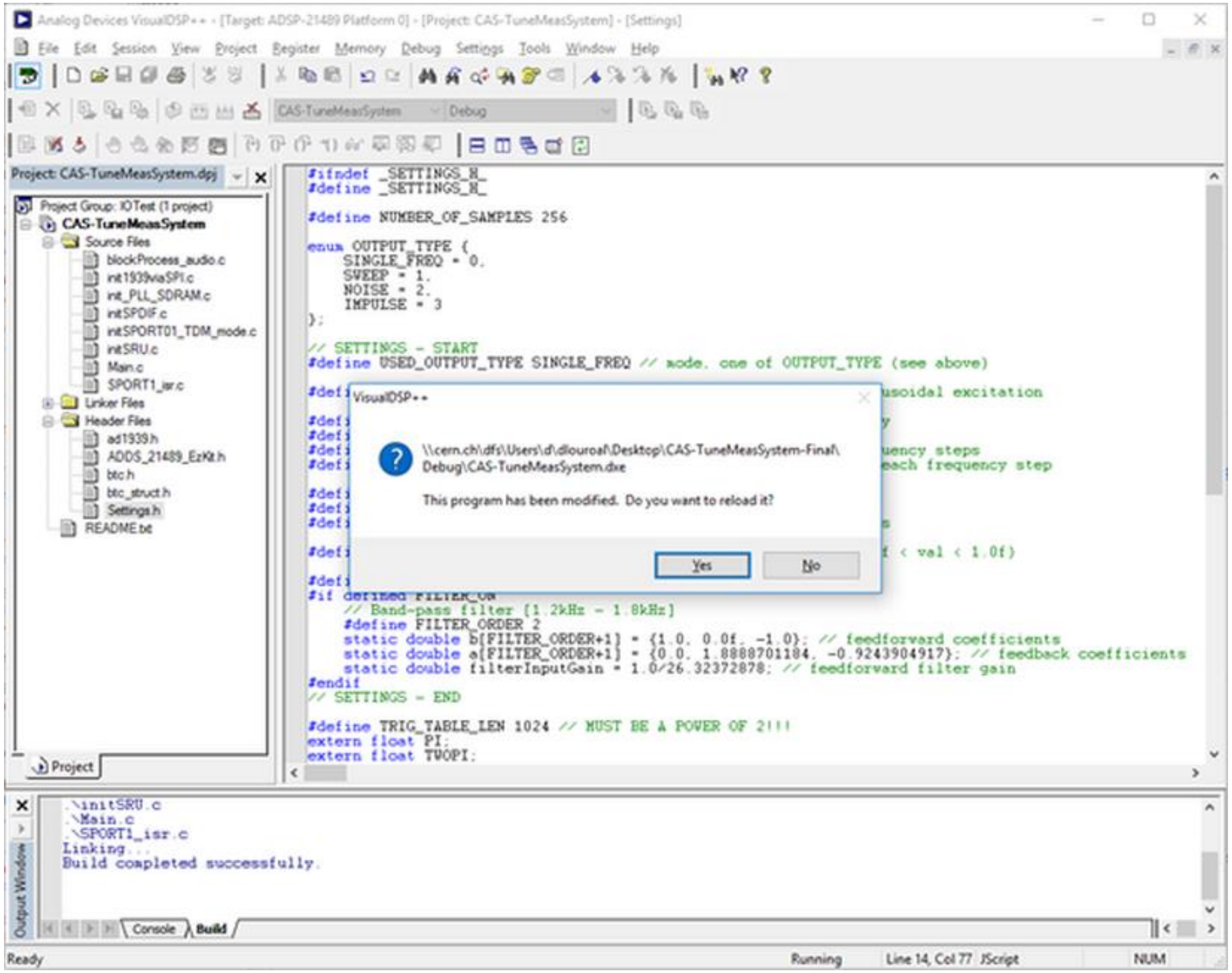
```
Builds the project as necessary          Halted          Line 14, Col 77   JScript          NUM
```

- Build project (subsequent builds):



```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   2000.0f // frequency of stationary sinusoidal excitation

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each frequency step

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1.0f)

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coefficients
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; // feedback coefficients
    static double filterInputGain = 1.0/26.32372878; // feedforward filter gain
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
>>
```


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]

Project: CAS-TuneMeasSystem.dpj

```
Project Group: IOTest (1 project)
  CAS-TuneMeasSystem
    Source Files
      blockProcess_audio.c
      init1939viaSPI.c
      init_PLL_SDRAM.c
      initSPDIF.c
      initSPORT01_TDM_mode.c
      initSRU.c
      Main.c
      SPORT1_isr.c
    Linker Files
    Header Files
      ad1939.h
      ADDS_21489_EzKit.h
      btc.h
      btc_struct.h
      Settings.h
    README.txt
```

```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (see above)

#defi                                          usoidal excitation

#defi                                          y
#defi
#defi                                          uency steps
#defi                                          each frequency step

#defi
#defi
#defi                                          s

#defi                                          f < val < 1.0f)

#defi
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coefficients
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; // feedback coefficients
    static double filterInputGain = 1.0/26.32372878; // feedforward filter gain
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

VisualDSP++

\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe

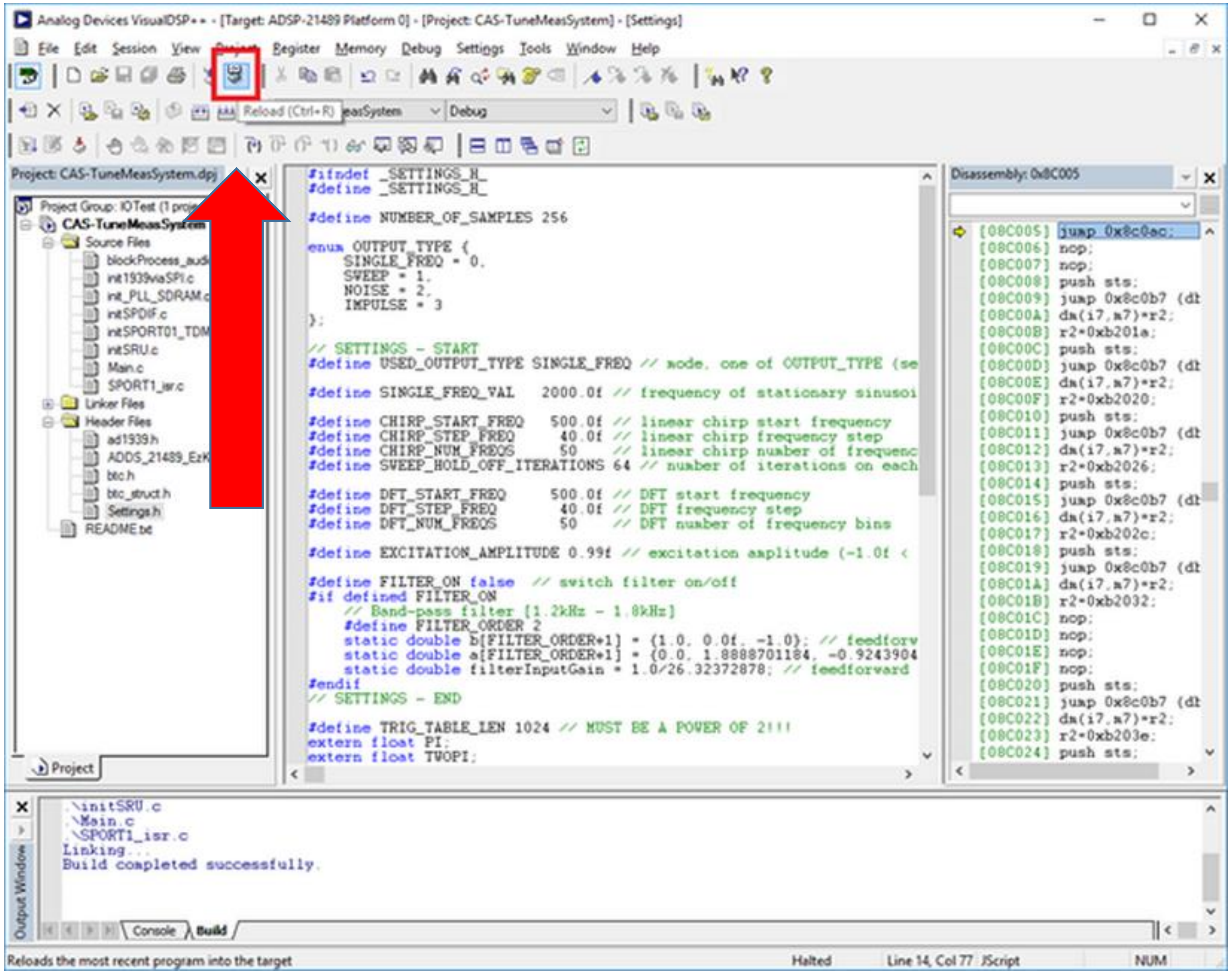
This program has been modified. Do you want to reload it?

Yes No

```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

Ready   Running   Line 14, Col 77 JScript   NUM

- Load compiled binary into DSP (automatic):


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]

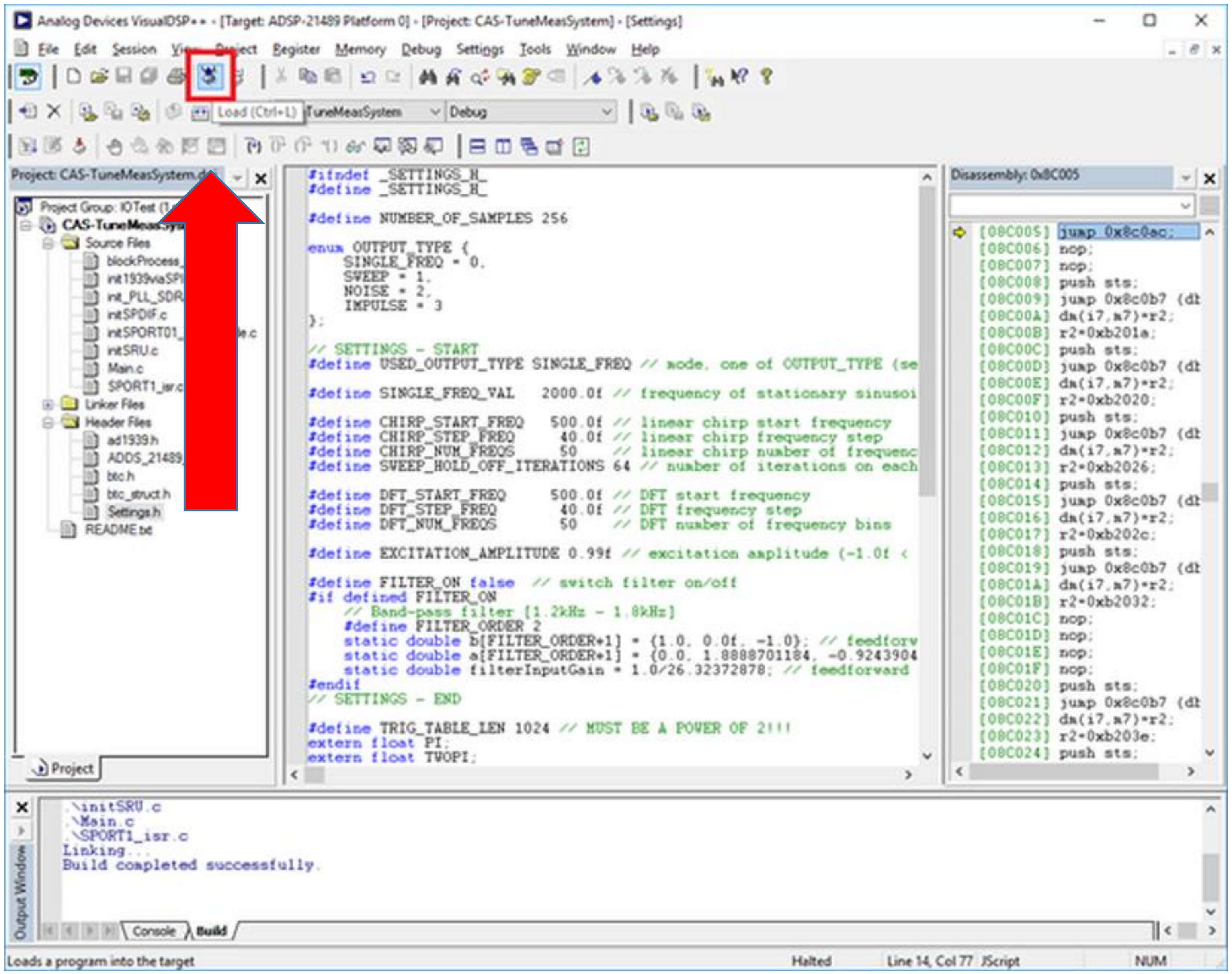
```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (se

#define SINGLE_FREQ_VAL    2000.0f // frequency of stationary sinusoi

#define CHIRP_START_FREQ    500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ      40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS      50    // linear chirp number of frequenc
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each

#define DFT_START_FREQ      500.0f // DFT start frequency
#define DFT_STEP_FREQ        40.0f // DFT frequency step
#define DFT_NUM_FREQS        50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f <

#define FILTER_ON false   // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforw
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904
    static double filterInputGain = 1.0/26.32372878; // feedforward
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
[08C005] jump 0x8c0ac;
[08C006] nop;
[08C007] nop;
[08C008] push sts;
[08C009] jump 0x8c0b7 (db
[08C00A] dm(i7,m7)=r2;
[08C00B] r2=0xb201a;
[08C00C] push sts;
[08C00D] jump 0x8c0b7 (db
[08C00E] dm(i7,m7)=r2;
[08C00F] r2=0xb2020;
[08C010] push sts;
[08C011] jump 0x8c0b7 (db
[08C012] dm(i7,m7)=r2;
[08C013] r2=0xb2026;
[08C014] push sts;
[08C015] jump 0x8c0b7 (db
[08C016] dm(i7,m7)=r2;
[08C017] r2=0xb202c;
[08C018] push sts;
[08C019] jump 0x8c0b7 (db
[08C01A] dm(i7,m7)=r2;
[08C01B] r2=0xb2032;
[08C01C] nop;
[08C01D] nop;
[08C01E] nop;
[08C01F] nop;
[08C020] push sts;
[08C021] jump 0x8c0b7 (db
[08C022] dm(i7,m7)=r2;
[08C023] r2=0xb203e;
[08C024] push sts;
```

```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

Reloads the most recent program into the target   Halted   Line 14, Col 77 JScript   NUM

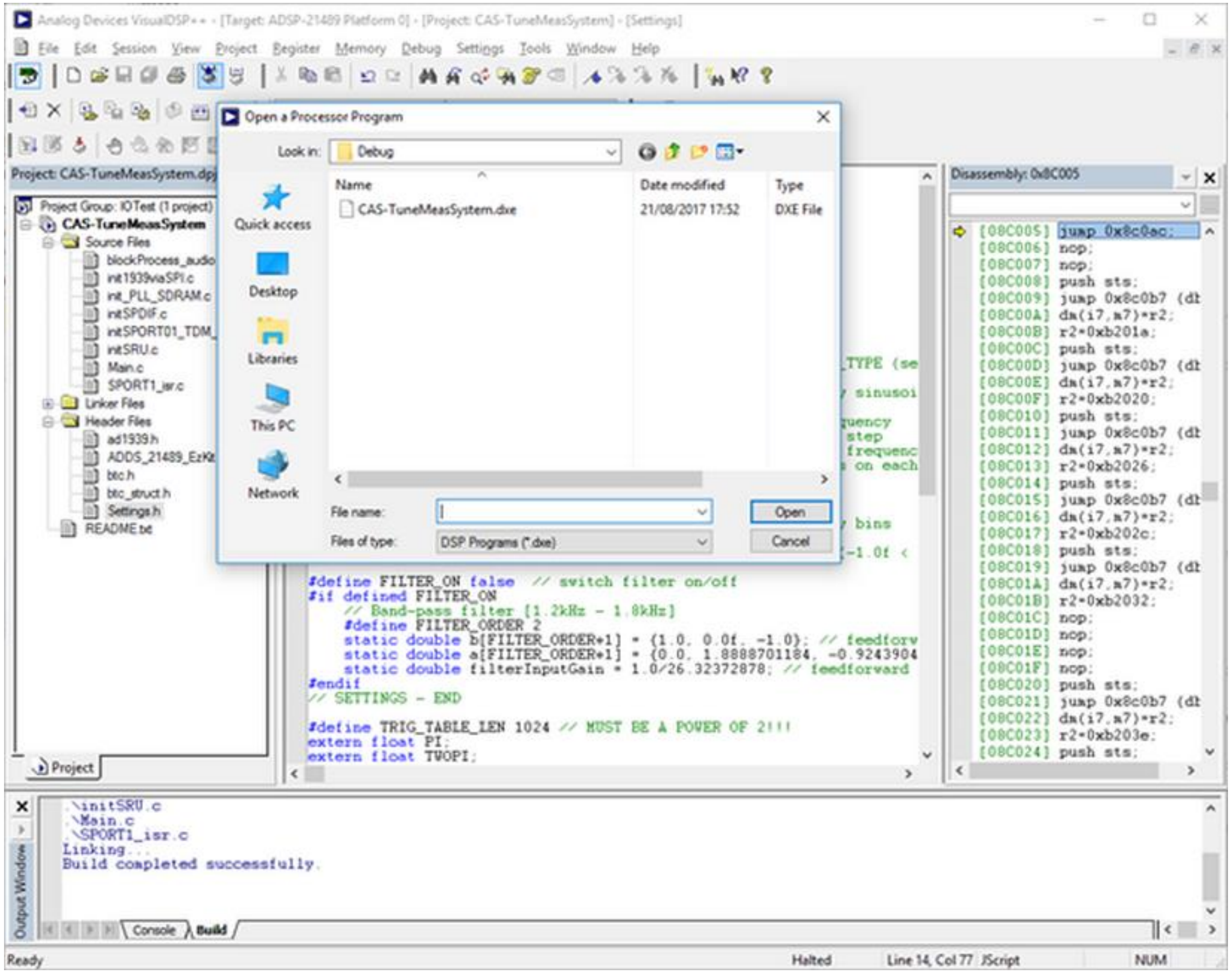
- Load binary file to DSP (manual):



```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (se

#define SINGLE_FREQ_VAL    2000.0f // frequency of stationary sinusoi

#define CHIRP_START_FREQ    500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ      40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS      50    // linear chirp number of frequenc
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each

#define DFT_START_FREQ      500.0f // DFT start frequency
#define DFT_STEP_FREQ        40.0f // DFT frequency step
#define DFT_NUM_FREQS        50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f <

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforw
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904
    static double filterInputGain = 1.0/26.32372878; // feedforward
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
[08C005] jump 0x8c0ac;
[08C006] nop;
[08C007] nop;
[08C008] push sts;
[08C009] jump 0x8c0b7 (db
[08C00A] dm(i7,m7)=r2;
[08C00B] r2=0xb201a;
[08C00C] push sts;
[08C00D] jump 0x8c0b7 (db
[08C00E] dm(i7,m7)=r2;
[08C00F] r2=0xb2020;
[08C010] push sts;
[08C011] jump 0x8c0b7 (db
[08C012] dm(i7,m7)=r2;
[08C013] r2=0xb2026;
[08C014] push sts;
[08C015] jump 0x8c0b7 (db
[08C016] dm(i7,m7)=r2;
[08C017] r2=0xb202c;
[08C018] push sts;
[08C019] jump 0x8c0b7 (db
[08C01A] dm(i7,m7)=r2;
[08C01B] r2=0xb2032;
[08C01C] nop;
[08C01D] nop;
[08C01E] nop;
[08C01F] nop;
[08C020] push sts;
[08C021] jump 0x8c0b7 (db
[08C022] dm(i7,m7)=r2;
[08C023] r2=0xb203e;
[08C024] push sts;
```

```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

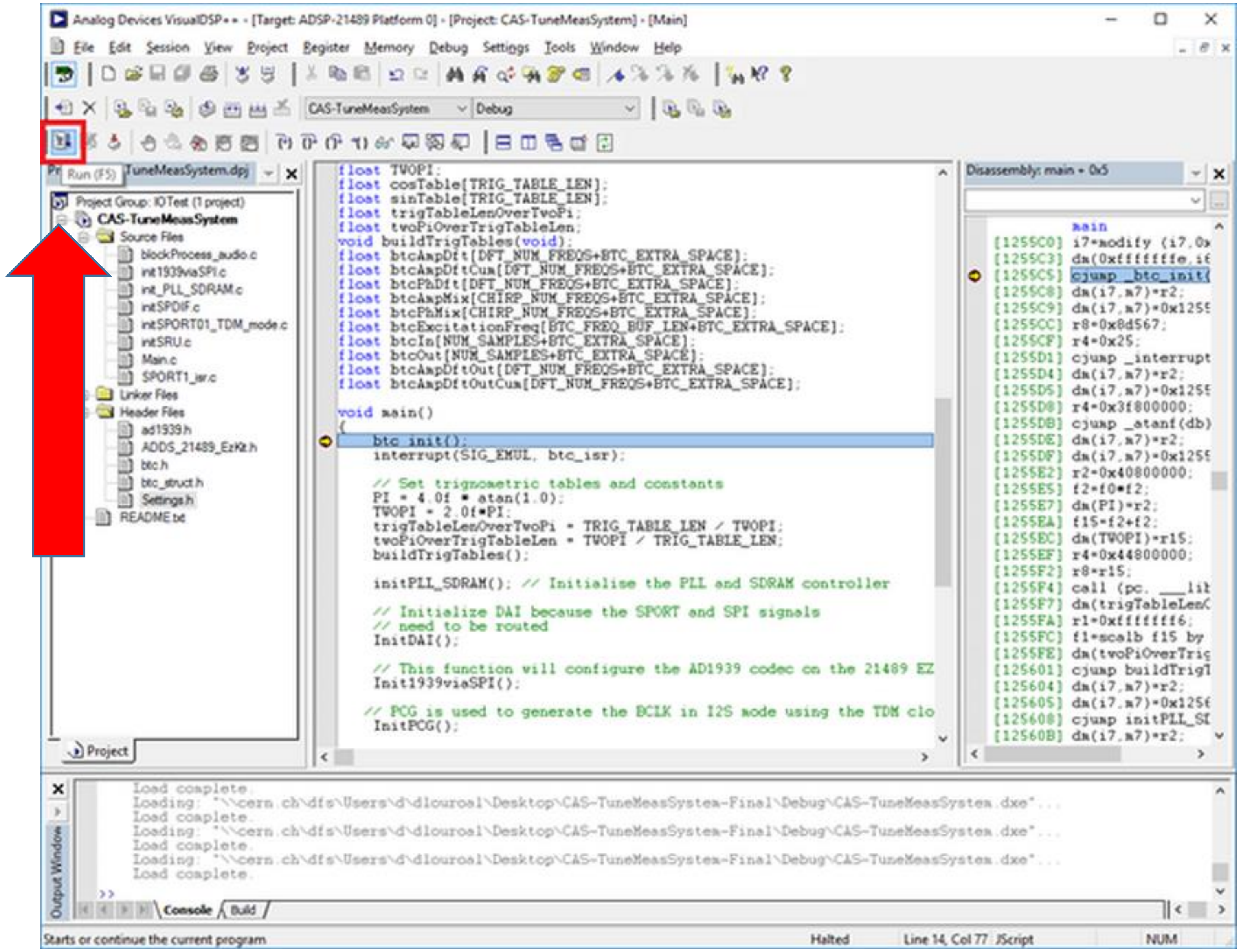


```
#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforw
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904
    static double filterInputGain = 1.0/26.32372878; // feedforward
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
[08C005] jump 0x8c0ac;
[08C006] nop;
[08C007] nop;
[08C008] push sts;
[08C009] jump 0x8c0b7 (db
[08C00A] dm(i7,m7)=r2;
[08C00B] r2=0xb201a;
[08C00C] push sts;
[08C00D] jump 0x8c0b7 (db
[08C00E] dm(i7,m7)=r2;
[08C00F] r2=0xb2020;
[08C010] push sts;
[08C011] jump 0x8c0b7 (db
[08C012] dm(i7,m7)=r2;
[08C013] r2=0xb2026;
[08C014] push sts;
[08C015] jump 0x8c0b7 (db
[08C016] dm(i7,m7)=r2;
[08C017] r2=0xb202c;
[08C018] push sts;
[08C019] jump 0x8c0b7 (db
[08C01A] dm(i7,m7)=r2;
[08C01B] r2=0xb2032;
[08C01C] nop;
[08C01D] nop;
[08C01E] nop;
[08C01F] nop;
[08C020] push sts;
[08C021] jump 0x8c0b7 (db
[08C022] dm(i7,m7)=r2;
[08C023] r2=0xb203e;
[08C024] push sts;
```

```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

- Run code:

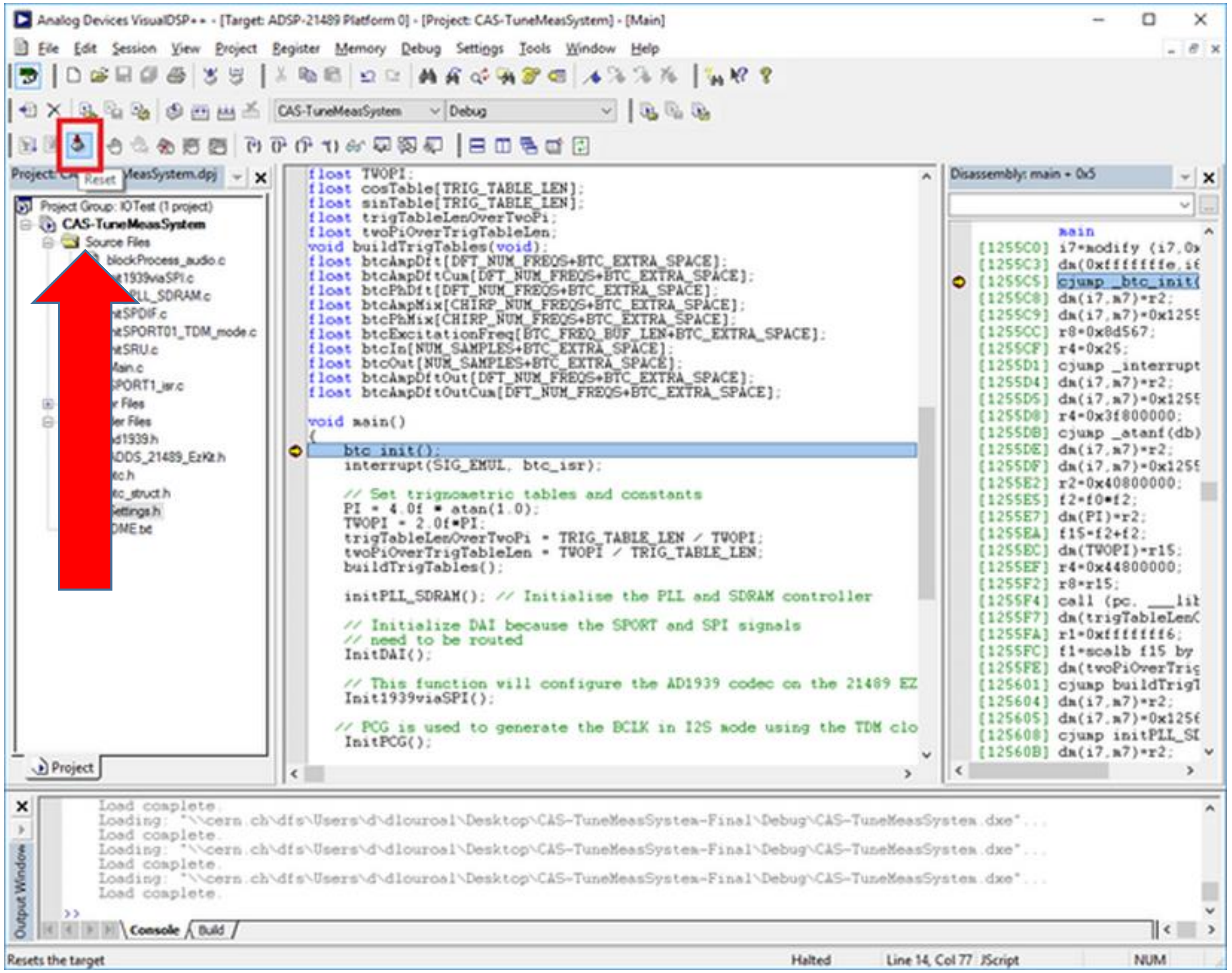


```
float TWOPI;
float cosTable[TRIG_TABLE_LEN];
float sinTable[TRIG_TABLE_LEN];
float trigTableLenOverTwoPi;
float twoPiOverTrigTableLen;
void buildTrigTables(void);
float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
float btcExcitationFreq[BTC_FREQ_BUF_LEN+BTC_EXTRA_SPACE];
float btcIn[NUM_SAMPLES+BTC_EXTRA_SPACE];
float btcOut[NUM_SAMPLES+BTC_EXTRA_SPACE];
float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

void main()
{
    btc_init();
    interrupt(SIG_EMUL, btc_isr);

    // Set trigonometric tables and constants
    PI = 4.0f * atan(1.0);
    TWOPI = 2.0f*PI;
    trigTableLenOverTwoPi = TRIG_TABLE_LEN / TWOPI;
    twoPiOverTrigTableLen = TWOPI / TRIG_TABLE_LEN;
    buildTrigTables();

    initPLL_SDRAM(); // Initialise the PLL and SDRAM controller

    // Initialize DAI because the SPORT and SPI signals
    // need to be routed
    InitDAI();

    // This function will configure the AD1939 codec on the 21489 EZ
    Init1939viaSPI();

   // PCG is used to generate the BCLK in I2S mode using the TDM clo
    InitPCG();
```

```
                main
[1255C0] i7=modify (i7,0x
[1255C3] dm(0xfffffffe,i6
[1255C5] cjump _btc_init(
[1255C8] dm(i7,m7)=r2;
[1255C9] dm(i7,m7)=0x1255
[1255CC] r8=0x8d567;
[1255CF] r4=0x25;
[1255D1] cjump _interrupt
[1255D4] dm(i7,m7)=r2;
[1255D5] dm(i7,m7)=0x1255
[1255D8] r4=0x3f800000;
[1255DB] cjump _atanf(db)
[1255DE] dm(i7,m7)=r2;
[1255DF] dm(i7,m7)=0x1255
[1255E2] r2=0x40800000;
[1255E5] f2=f0*f2;
[1255E7] dm(PI)=r2;
[1255EA] f15=f2+f2;
[1255EC] dm(TWOPI)=r15;
[1255EF] r4=0x44800000;
[1255F2] r8=r15;
[1255F4] call (pc, ___lib
[1255F7] dm(trigTableLenO
[1255FA] r1=0xfffffff6;
[1255FC] f1=scalb f15 by
[1255FE] dm(twoPiOverTrig
[125601] cjump buildTrigT
[125604] dm(i7,m7)=r2;
[125605] dm(i7,m7)=0x1256
[125608] cjump initPLL_SD
[12560B] dm(i7,m7)=r2;
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
>>
```

- Stop and reset:



```
float TWOPI;
float cosTable[TRIG_TABLE_LEN];
float sinTable[TRIG_TABLE_LEN];
float trigTableLenOverTwoPi;
float twoPiOverTrigTableLen;
void buildTrigTables(void);
float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
float btcExcitationFreq[BTC_FREQ_BUF_LEN+BTC_EXTRA_SPACE];
float btcIn[NUM_SAMPLES+BTC_EXTRA_SPACE];
float btcOut[NUM_SAMPLES+BTC_EXTRA_SPACE];
float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

void main()
{
    btc_init();
    interrupt(SIG_EMUL, btc_isr);

    // Set trigonometric tables and constants
    PI = 4.0f * atan(1.0);
    TWOPI = 2.0f*PI;
    trigTableLenOverTwoPi = TRIG_TABLE_LEN / TWOPI;
    twoPiOverTrigTableLen = TWOPI / TRIG_TABLE_LEN;
    buildTrigTables();

    initPLL_SDRAM(); // Initialise the PLL and SDRAM controller

    // Initialize DAI because the SPORT and SPI signals
    // need to be routed
    InitDAI();

    // This function will configure the AD1939 codec on the 21489 EZ
    Init1939viaSPI();

   // PCG is used to generate the BCLK in I2S mode using the TDM clo
    InitPCG();
```

```
          main
[1255C0] i7=modify (i7,0x
[1255C3] dm(0xfffffffe,i6
[1255C5] cjump _btc_init(
[1255C8] dm(i7,m7)=r2;
[1255C9] dm(i7,m7)=0x1255
[1255CC] r8=0x8d567;
[1255CF] r4=0x25;
[1255D1] cjump _interrupt
[1255D4] dm(i7,m7)=r2;
[1255D5] dm(i7,m7)=0x1255
[1255D8] r4=0x3f800000;
[1255DB] cjump _atanf(db)
[1255DE] dm(i7,m7)=r2;
[1255DF] dm(i7,m7)=0x1255
[1255E2] r2=0x40800000;
[1255E5] f2=f0*f2;
[1255E7] dm(PI)=r2;
[1255EA] f15=f2+f2;
[1255EC] dm(TWOPI)=r15;
[1255EF] r4=0x44800000;
[1255F2] r8=r15;
[1255F4] call (pc, ___lib
[1255F7] dm(trigTableLenO
[1255FA] r1=0xfffffff6;
[1255FC] f1=scalb f15 by
[1255FE] dm(twoPiOverTrig
[125601] cjump buildTrigT
[125604] dm(i7,m7)=r2;
[125605] dm(i7,m7)=0x1256
[125608] cjump initPLL_SD
[12560B] dm(i7,m7)=r2;
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
>>
```

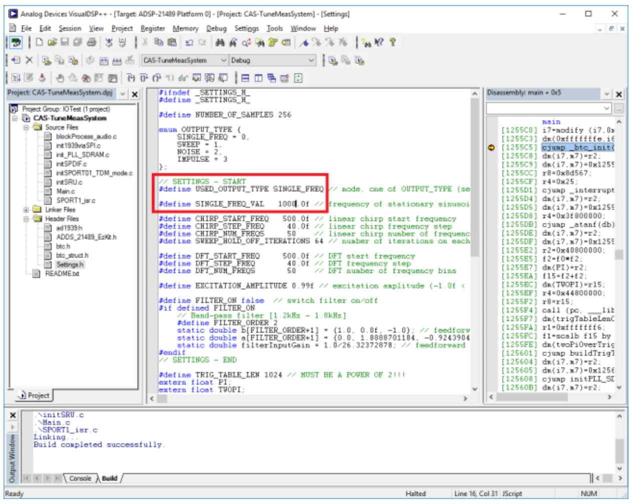
Exersises:

Exercise 0: Getting familiar

Put the setup into the mode of “continuous sine wave excitation” (=: SINGLE_FREQ) and make all connections to the DSP.

Use one channel of the oscilloscope to observe the excitation signal of the beam, the other channel to observe the beam oscillation. Choose as excitation frequency 1000 Hz, (re)build and run.

Parameter page:

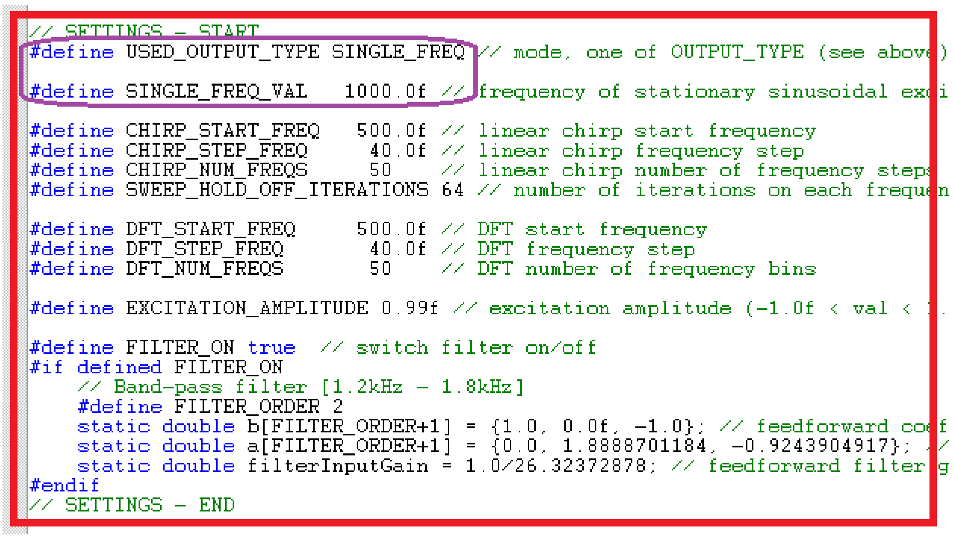


```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (se

#define SINGLE_FREQ_VAL    1000.0f // frequency of stationary sinusoi
#define CHIRP_START_FREQ    500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ      40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS      50    // linear chirp number of frequenc
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each

#define DFT_START_FREQ      500.0f // DFT start frequency
#define DFT_STEP_FREQ        40.0f // DFT frequency step
#define DFT_NUM_FREQS        50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f <

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforw
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904
    static double filterInputGain = 1.0/26.32372878; // feedforward
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
          main
[1255C0] i7=modify (i7,0x
[1255C3] dm(0xfffffffe,i6
[1255C5] cjump _btc_init(
[1255C8] dm(i7,m7)=r2;
[1255C9] dm(i7,m7)=0x1255
[1255CC] r8=0x8d567;
[1255CF] r4=0x25;
[1255D1] cjump _interrupt
[1255D4] dm(i7,m7)=r2;
[1255D5] dm(i7,m7)=0x1255
[1255D8] r4=0x3f800000;
[1255DB] cjump _atanf(db)
[1255DE] dm(i7,m7)=r2;
[1255DF] dm(i7,m7)=0x1255
[1255E2] r2=0x40800000;
[1255E5] f2=f0*f2;
[1255E7] dm(PI)=r2;
[1255EA] f15=f2+f2;
[1255EC] dm(TWOPI)=r15;
[1255EF] r4=0x44800000;
[1255F2] r8=r15;
[1255F4] call (pc, ___lib
[1255F7] dm(trigTableLen(
[1255FA] r1=0xfffffff6;
[1255FC] f1=scalb f15 by
[1255FE] dm(twoPiOverTrig
[125601] cjump buildTrigT
[125604] dm(i7,m7)=r2;
[125605] dm(i7,m7)=0x1256
[125608] cjump initPLL_SD
[12560B] dm(i7,m7)=r2;
```

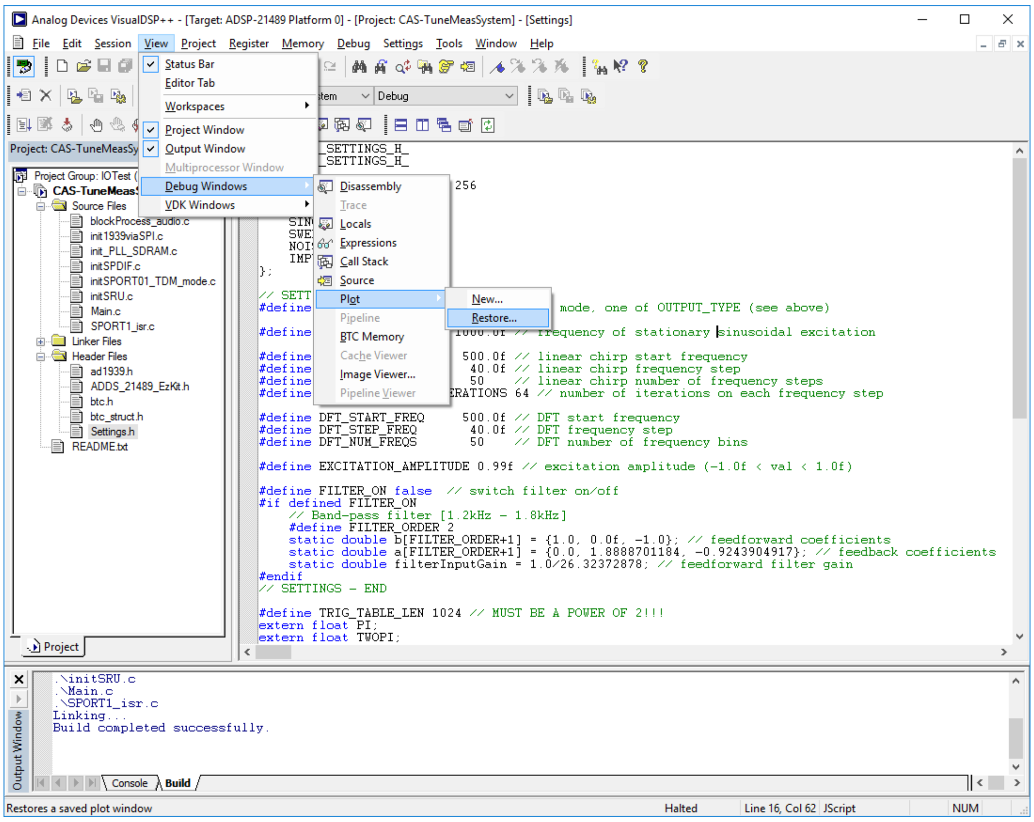
```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

Details:



```
// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL    1000.0f //frequency of stationary sinusoidal exci

#define CHIRP_START_FREQ    500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ      40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS      50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each frequen

#define DFT_START_FREQ      500.0f // DFT start frequency
#define DFT_STEP_FREQ        40.0f // DFT frequency step
#define DFT_NUM_FREQS        50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1.

#define FILTER_ON true  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coef
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; //
    static double filterInputGain = 1.0/26.32372878; // feedforward filtering
#endif
// SETTINGS - END
```

Getting the realtime displays generated by the DSP:


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]

Project: CAS-TuneMeasSy

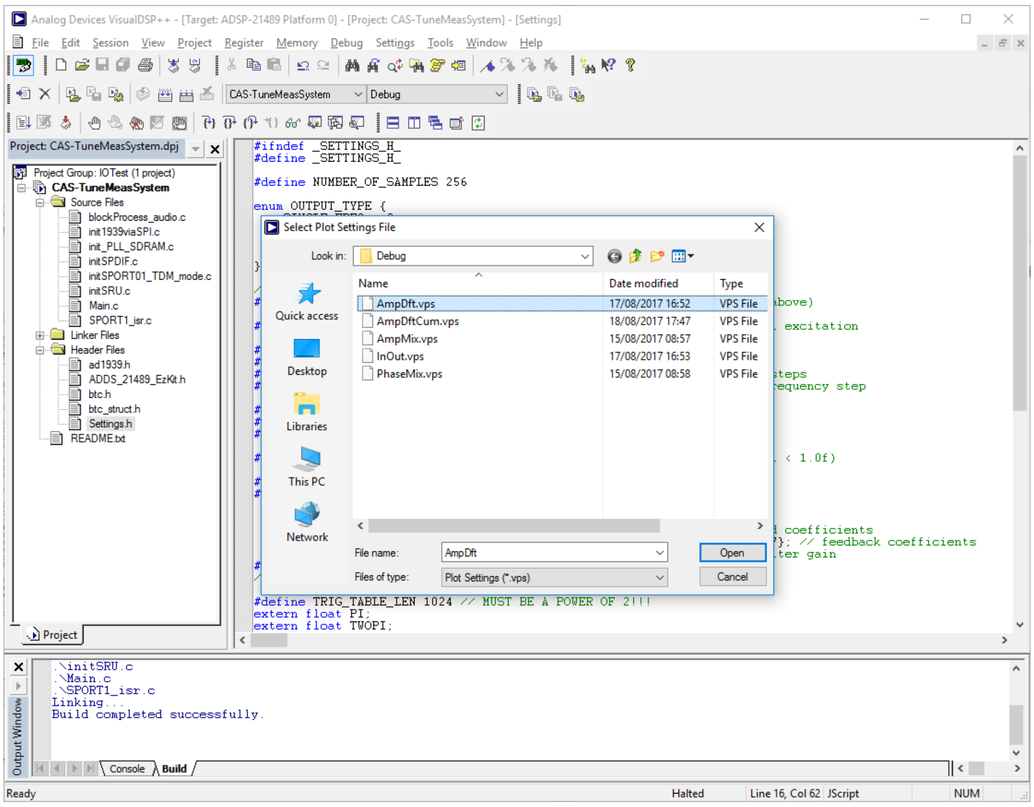
```
Project Group: IOTest (
  CAS-TuneMeas
    Source Files
      blockProcess_audio.c
      init1939viaSPI.c
      init_PLL_SDRAM.c
      initSPDIF.c
      initSPORT01_TDM_mode.c
      initSRU.c
      Main.c
      SPORT1_isr.c
    Linker Files
    Header Files
      ad1939.h
      ADSP_21489_EzKit.h
      btc.h
      btc_struct.h
      Settings.h
    README.txt
```

```
_SETTINGS_H_
_SETTINGS_H_
 256

    SIN
    SWE
    NOI
    IMP
};

// SETT
#define                            mode, one of OUTPUT_TYPE (see above)

#define                   1000.0f // frequency of stationary sinusoidal excitation

#define                    500.0f // linear chirp start frequency
#define                     40.0f // linear chirp frequency step
#define                     50    // linear chirp number of frequency steps
#define                 ERATIONS 64 // number of iterations on each frequency step

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1.0f)

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coefficients
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; // feedback coefficients
    static double filterInputGain = 1.0/26.32372878; // feedforward filter gain
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

Restores a saved plot window   Halted   Line 16, Col 62   JScript   NUM

Load AmpDft.vps, AmpMix.vps and PhaseMix.vps:


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]

Project: CAS-TuneMeasSystem.dpj

```
Project Group: IOTest (1 project)
  CAS-TuneMeasSystem
    Source Files
      blockProcess_audio.c
      init1939viaSPI.c
      init_PLL_SDRAM.c
      initSPDIF.c
      initSPORT01_TDM_mode.c
      initSRU.c
      Main.c
      SPORT1_isr.c
    Linker Files
    Header Files
      ad1939.h
      ADSP_21489_EzKit.h
      btc.h
      btc_struct.h
      Settings.h
    README.txt
```

```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
```

Select Plot Settings File

Look in: Debug

Name	Date modified	Type
AmpDft.vps	17/08/2017 16:52	VPS File
AmpDftCum.vps	18/08/2017 17:47	VPS File
AmpMix.vps	15/08/2017 08:57	VPS File
InOut.vps	17/08/2017 16:53	VPS File
PhaseMix.vps	15/08/2017 08:58	VPS File

File name: AmpDft

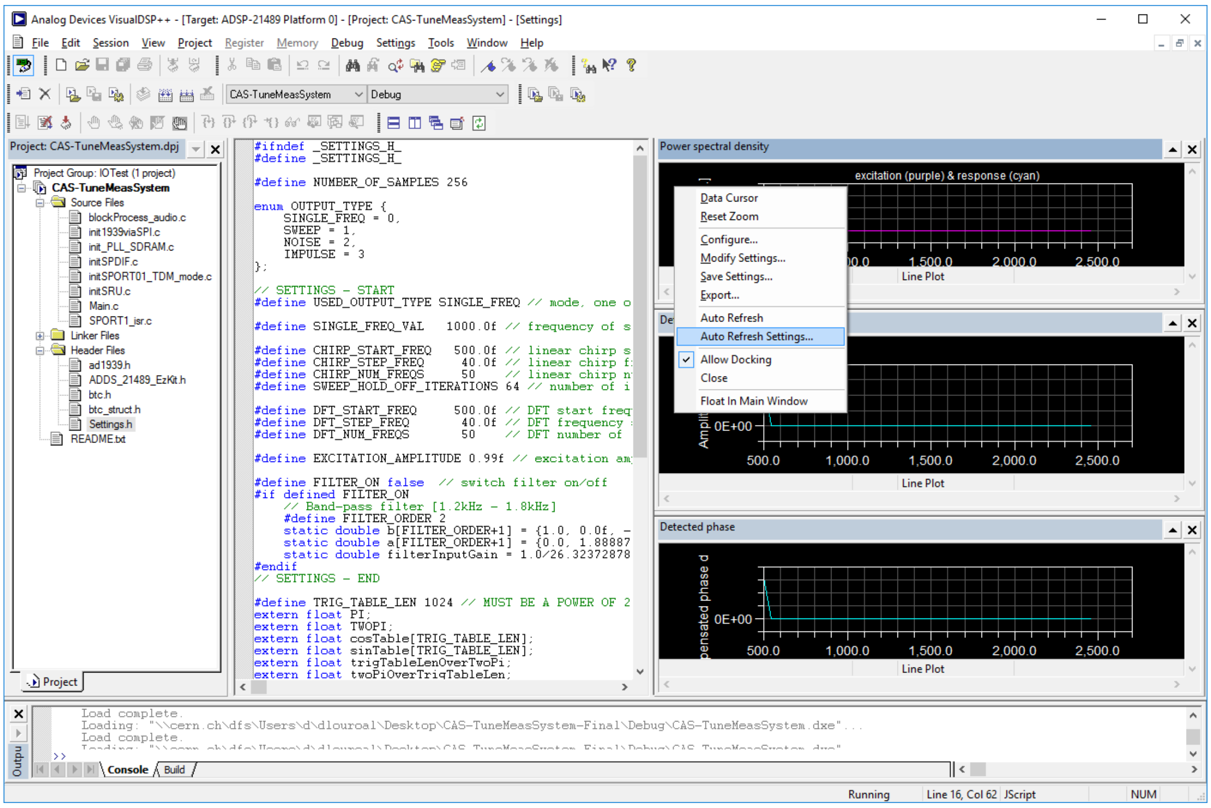
Files of type: Plot Settings (*.vps)

```
#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
```

```
.\initSRU.c
.\Main.c
.\SPORT1_isr.c
Linking...
Build completed successfully.
```

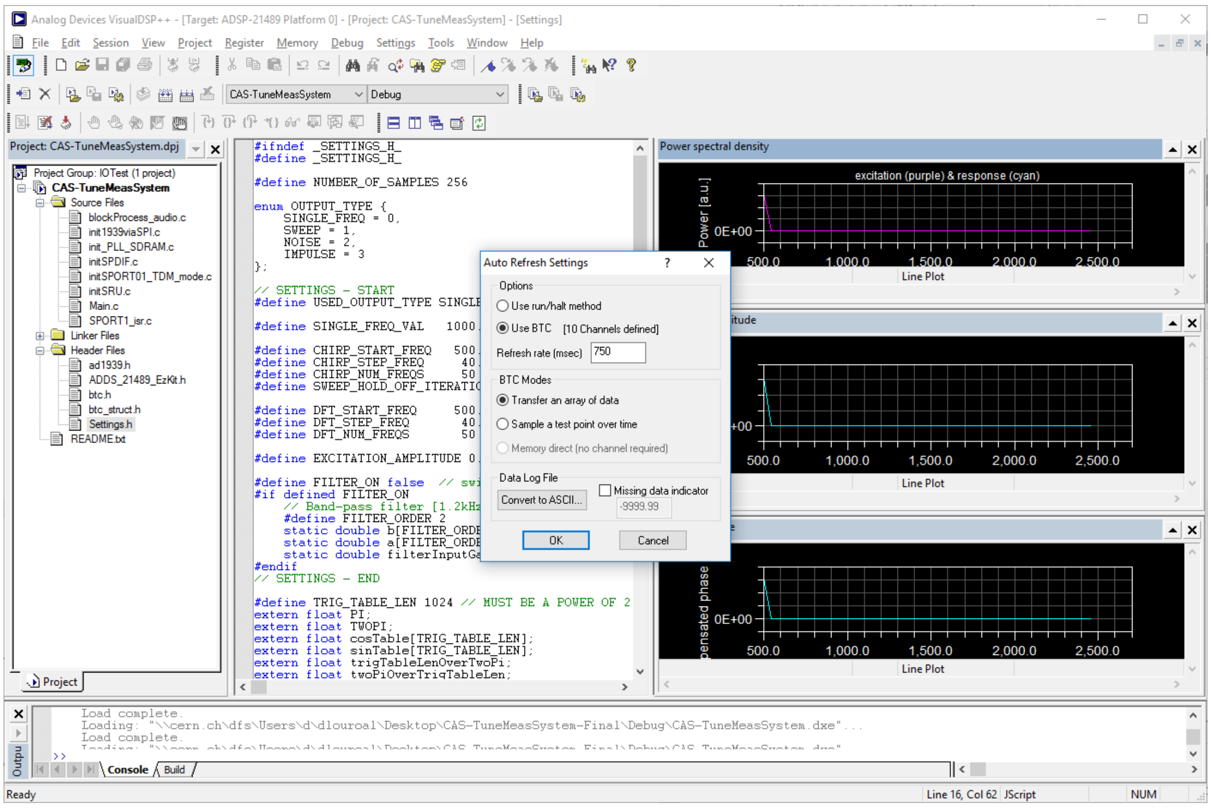
Ready   Halted   Line 16, Col 62   JScript   NUM

Right-click on any plot window to setup auto-refresh settings.


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]

```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one o

#define SINGLE_FREQ_VAL   1000.0f // frequency of s

#define CHIRP_START_FREQ   500.0f // linear chirp s
#define CHIRP_STEP_FREQ     40.0f // linear chirp f
#define CHIRP_NUM_FREQS     50    // linear chirp n
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of i

#define DFT_START_FREQ     500.0f // DFT start freq
#define DFT_STEP_FREQ       40.0f // DFT frequency
#define DFT_NUM_FREQS       50    // DFT number of

#define EXCITATION_AMPLITUDE 0.99f // excitation am

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -
    static double a[FILTER_ORDER+1] = {0.0, 1.88887
    static double filterInputGain = 1.0/26.32372878
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
```


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]

Auto Refresh Settings

Options
- ( ) Use run/halt method
- (●) Use BTC [10 Channels defined]

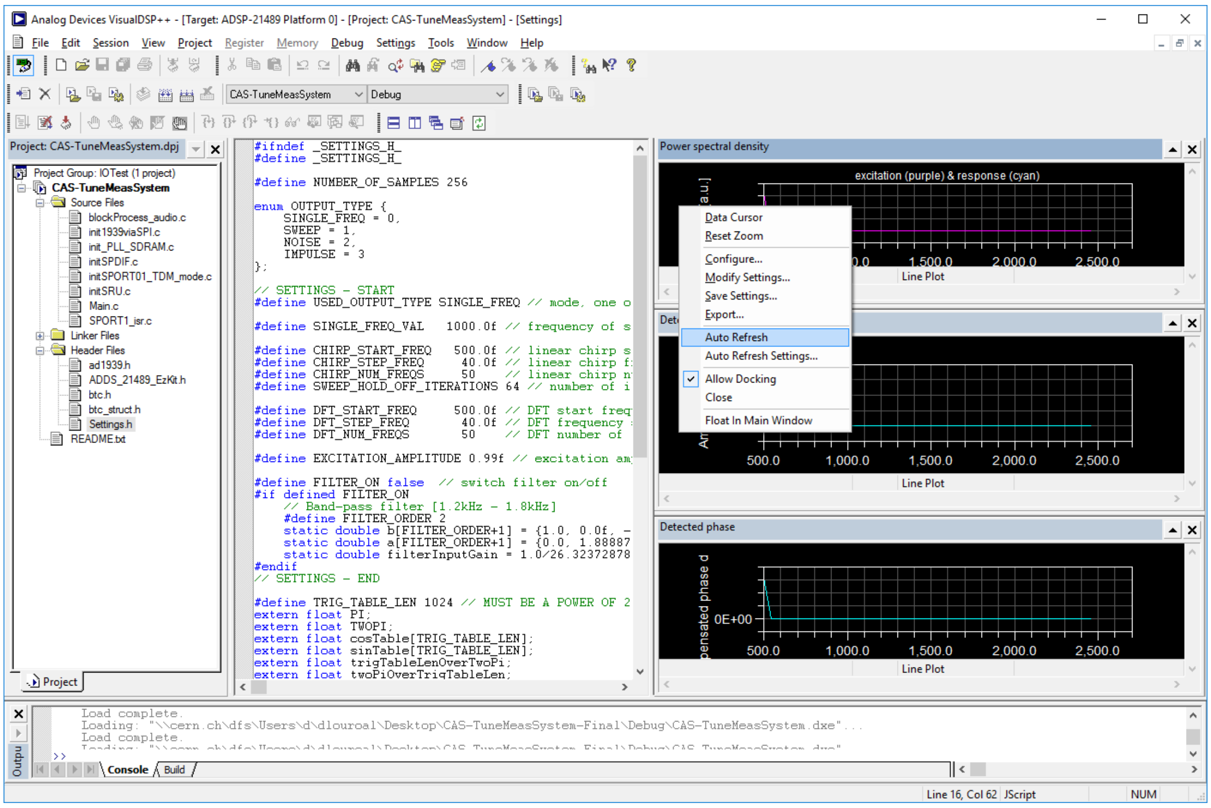
Refresh rate (msec) 750

BTC Modes
- (●) Transfer an array of data
- ( ) Sample a test point over time
- ( ) Memory direct (no channel required)

Data Log File
- Convert to ASCII...
- [ ] Missing data indicator -9999.99

OK Cancel

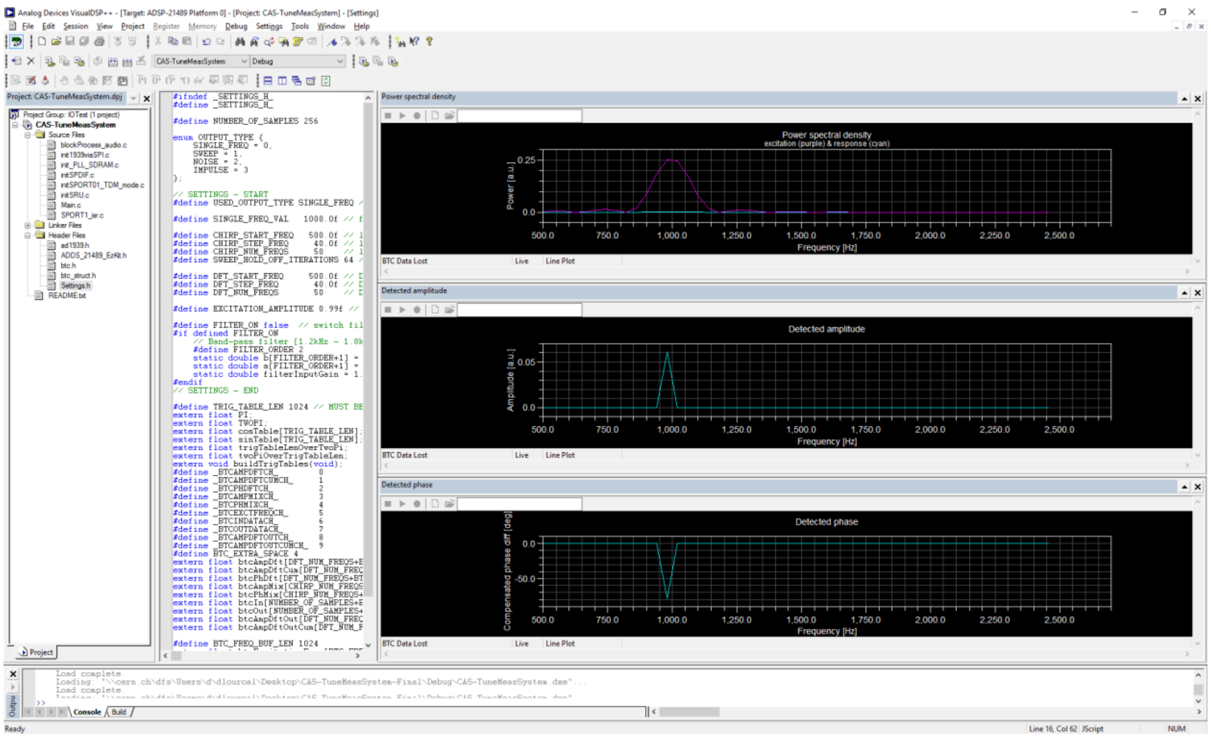
Right-click on each plot window and enable auto refresh:



```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ // mode, one o

#define SINGLE_FREQ_VAL   1000.0f // frequency of s

#define CHIRP_START_FREQ   500.0f // linear chirp s
#define CHIRP_STEP_FREQ     40.0f // linear chirp f
#define CHIRP_NUM_FREQS     50    // linear chirp n
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of i

#define DFT_START_FREQ     500.0f // DFT start freq
#define DFT_STEP_FREQ       40.0f // DFT frequency
#define DFT_NUM_FREQS       50    // DFT number of

#define EXCITATION_AMPLITUDE 0.99f // excitation am

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -
    static double a[FILTER_ORDER+1] = {0.0, 1.88887
    static double filterInputGain = 1.0/26.32372878
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
```

Finally you should see:



```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SINGLE_FREQ /

#define SINGLE_FREQ_VAL   1000.0f // f

#define CHIRP_START_FREQ   500.0f // l
#define CHIRP_STEP_FREQ     40.0f // l
#define CHIRP_NUM_FREQS     50    // l
#define SWEEP_HOLD_OFF_ITERATIONS 64 /

#define DFT_START_FREQ     500.0f // D
#define DFT_STEP_FREQ       40.0f // D
#define DFT_NUM_FREQS       50    // D

#define EXCITATION_AMPLITUDE 0.99f //

#define FILTER_ON false  // switch fil
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8k
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] =
    static double a[FILTER_ORDER+1] =
    static double filterInputGain = 1.
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
extern void buildTrigTables(void);
#define _BTCAMPDFTCH_         0
#define _BTCAMPDFTCUMCH_      1
#define _BTCPHDFTCH_          2
#define _BTCAMPMIXCH_         3
#define _BTCPHMIXCH_          4
#define _BTCEXCTFREQCH_       5
#define _BTCINDATACH_         6
#define _BTCOUTDATACH_        7
#define _BTCAMPDFTOUTCH_      8
#define _BTCAMPDFTOUTCUMCH_   9
#define BTC_EXTRA_SPACE 4
extern float btcAmpDft[DFT_NUM_FREQS+B
extern float btcAmpDftCum[DFT_NUM_FREC
extern float btcPhDft[DFT_NUM_FREQS+BT
extern float btcAmpMix[CHIRP_NUM_FREQS
extern float btcPhMix[CHIRP_NUM_FREQS+
extern float btcIn[NUMBER_OF_SAMPLES+B
extern float btcOut[NUMBER_OF_SAMPLES+
extern float btcAmpDftOut[DFT_NUM_FREC
extern float btcAmpDftOutCum[DFT_NUM_F

#define BTC_FREQ_BUF_LEN 1024
```

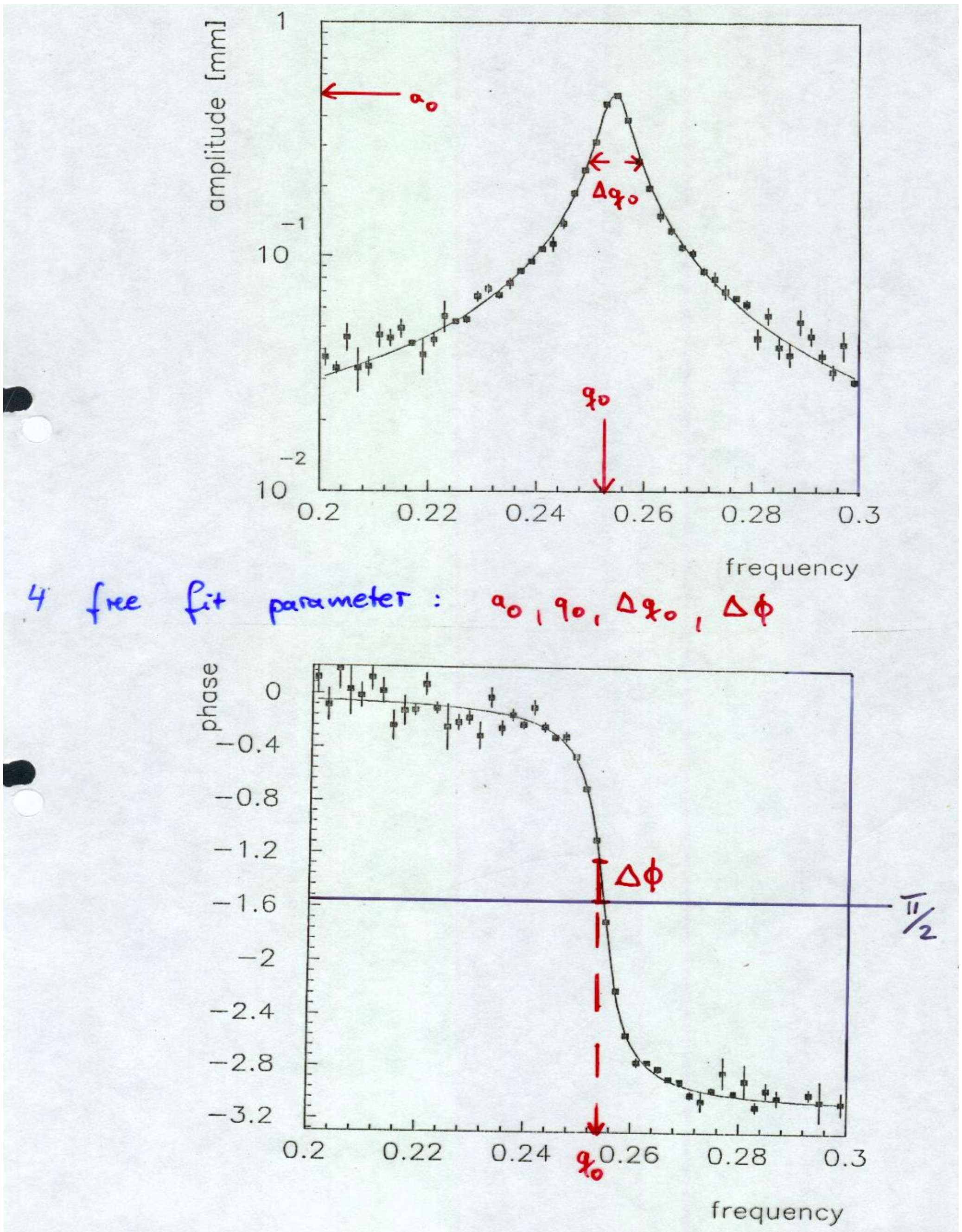
```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
```

Exercise 1: Measuring by hand the beam transfer function

Use the setup of above, but vary the fixed frequency in at least 7 steps in the range 500 Hz to 2500 Hz. At each frequency measure the amplitude of the beam oscillation and the phase between excitation and beam oscillation directly from the AmpMix and PhaseMix plots.

Draw on a piece of paper or in excel the results as a function of the excitation frequency.

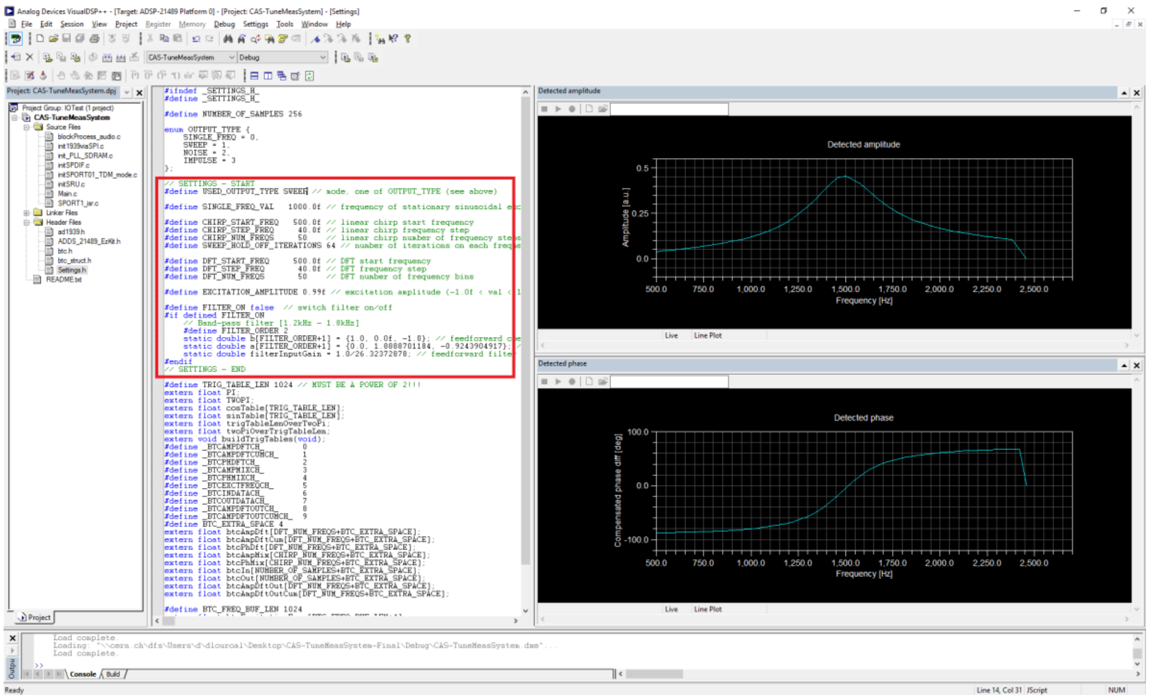
Below: Measurement on beam from about 1991



Exercise 2: Have the DSP do exercise 1

Use again the same setup, but this time put the DSP into the mode “SWEPT FREQUENCY” also often called network analysis. Use as sweep range again 500…2500 Hz.

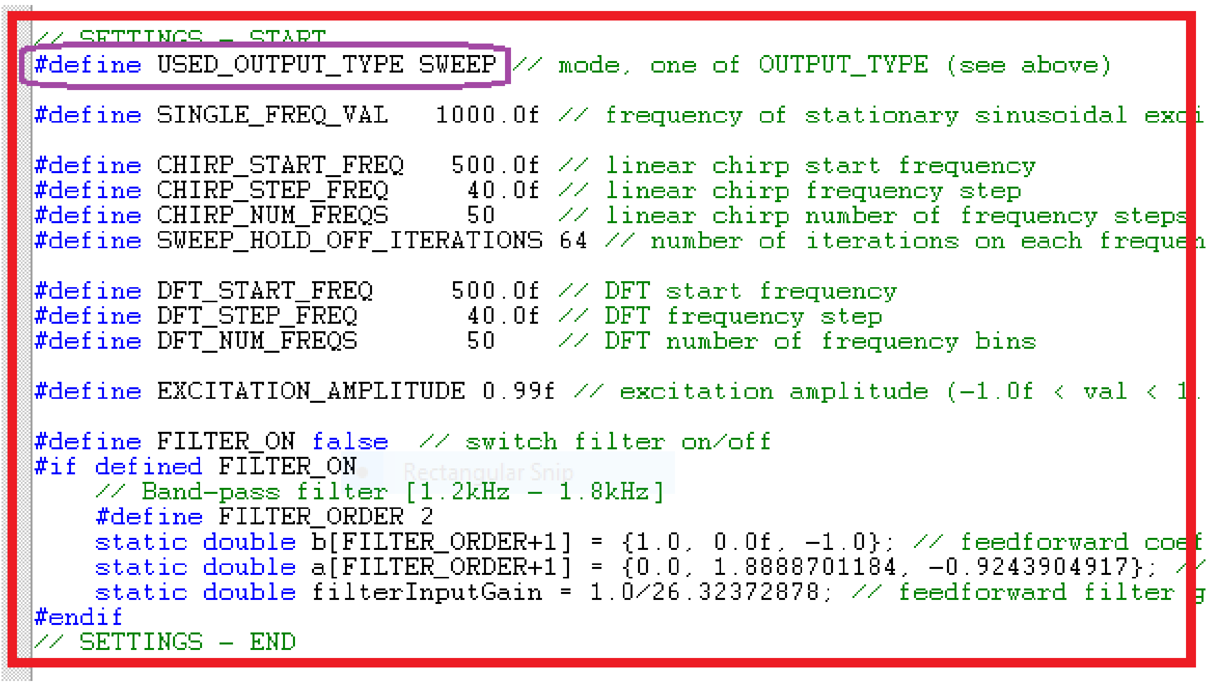
In real time the DSP will draw the curves which you have measured yourself.


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]
File Edit Session View Project Register Memory Debug Settings Tools Window Help
CAS-TuneMeasSystem
Debug
Project: CAS-TuneMeasSystem.dpj
Detected amplitude
Amplitude [a.u.]
Frequency [Hz]
Detected phase
Compensated phase diff [deg]
Live
Line Plot
Console
Build
Ready

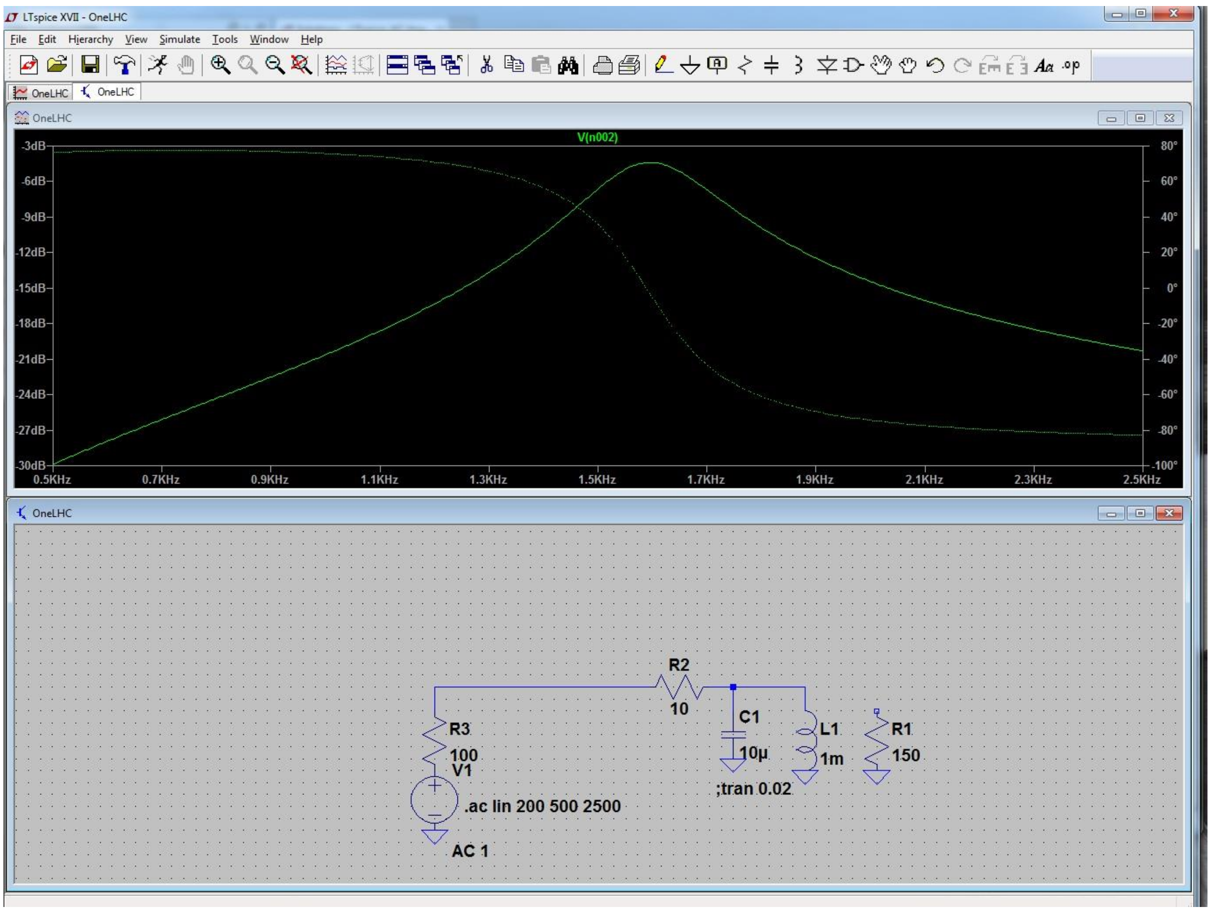
```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE SWEEP // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exc

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each freque

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coe
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; /
    static double filterInputGain = 1.0/26.32372878; // feedforward filte
#endif
// SETTINGS - END
#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
extern void buildTrigTables(void);
#define _BTCAMPDFTCH_          0
#define _BTCAMPDFTCUMCH_       1
#define _BTCPHDFTCH_           2
#define _BTCAMPMIXCH_          3
#define _BTCPHMIXCH_           4
#define _BTCEXCTFREQCH_        5
#define _BTCINDATACH_          6
#define _BTCOUTDATACH_         7
#define _BTCAMPDFTOUTCH_       8
#define _BTCAMPDFTOUTCUMCH_    9
#define BTC_EXTRA_SPACE 4
extern float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcIn[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcOut[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

#define BTC_FREQ_BUF_LEN 1024
```

```
Load complete.
Loading "\\cern.ch\dfs\Users\d\dlourcal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
```

Details:



```
// SETTINGS - START
#define USED_OUTPUT_TYPE SWEEP // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exci

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each frequen

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1.

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coef
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; //
    static double filterInputGain = 1.0/26.32372878; // feedforward filter g
#endif
// SETTINGS - END
```



.ac lin 200 500 2500

Above: Simulated result from LTspice

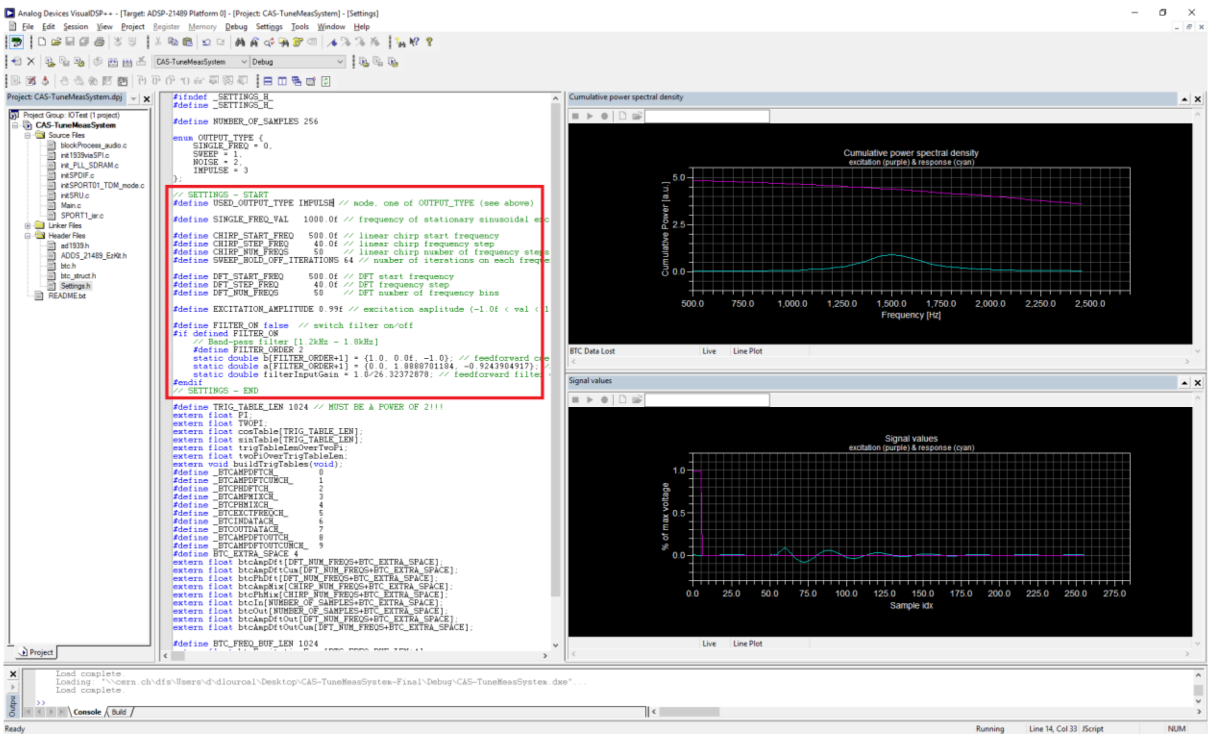
Exercise 3: Broad band excitation

We now switch to a different excitation mode of the beam, which we refer to as “noise excitation”. The advantage is that by applying noise, i.e. random excitation voltages, we excite the whole frequency spectrum, so we do can do the above measurements at the same time, with the price of reduced accuracy.
Noise can be applied in various ways, we offer three different options
a) single unipolar excitation pulse
b) regular random voltage excitation pulses
c) regular random excitation pulses after applying a filter, which is centred on the expected resonance frequency of the beam.

The advantages and disadvantages of each excitation mode will become clear during the exercise.

3a) Set the DSP to the mode “SINGLE PULSE EXCITATION”

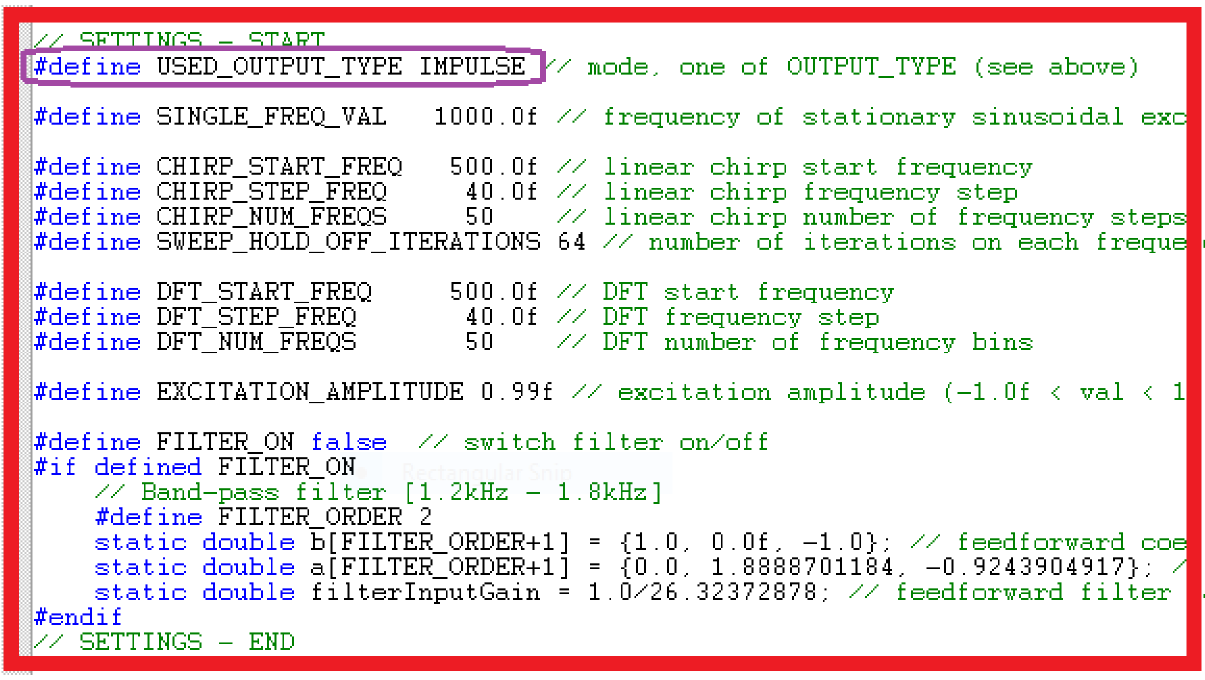
Observe with the oscilloscope the excitation signal (use this also to trigger the scope) and observe the beam oscillation. Make the appropriate code changes and load AmpDftCum.vps and InOut.vps.



```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE IMPULSE // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exc

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each freque

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coe
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; /
    static double filterInputGain = 1.0/26.32372878; // feedforward filte
#endif
// SETTINGS - END
#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
extern void buildTrigTables(void);
#define _BTCAMPDFTCH_        0
#define _BTCAMPDFTCUMCH_     1
#define _BTCPHDFTCH_         2
#define _BTCAMPMIXCH_        3
#define _BTCPHMIXCH_         4
#define _BTCEXCTFREQCH_      5
#define _BTCINDATACH_        6
#define _BTCOUTDATACH_       7
#define _BTCAMPDFTOUTCH_     8
#define _BTCAMPDFTOUTCUMCH_  9
#define BTC_EXTRA_SPACE 4
extern float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcIn[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcOut[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

#define BTC_FREQ_BUF_LEN 1024
```

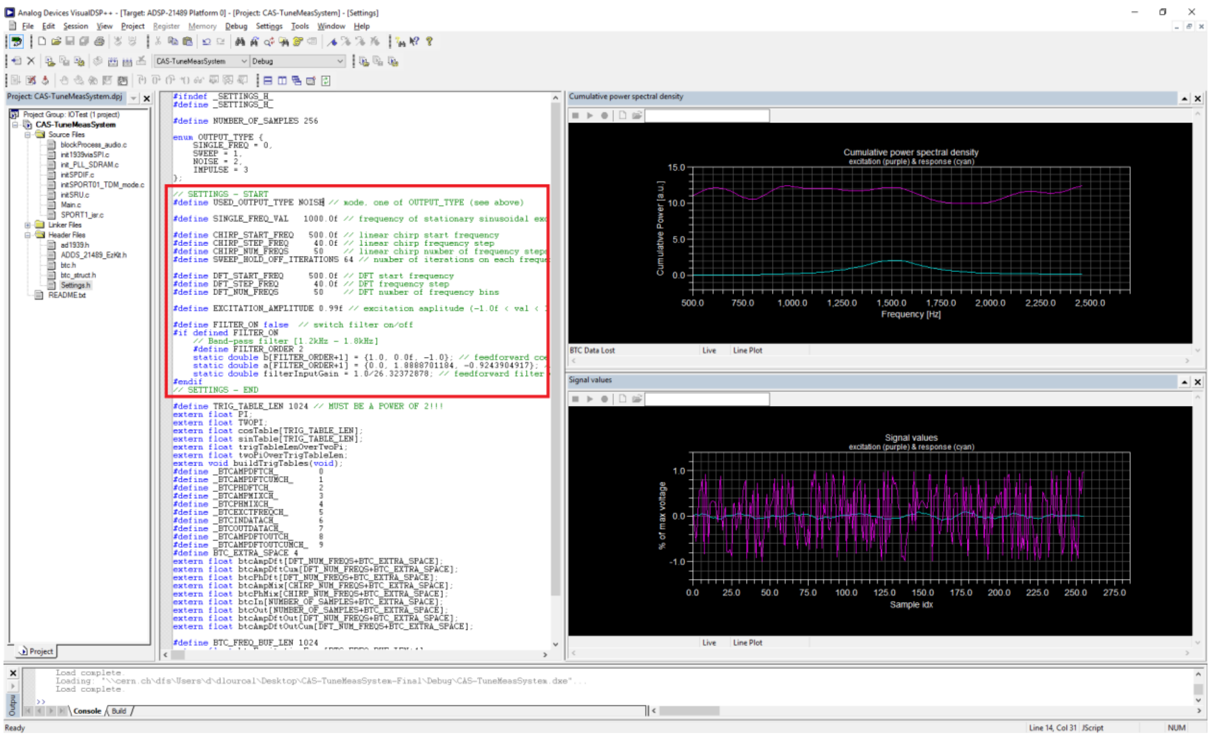
```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlourcal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
>>
```

Details of parameter file:



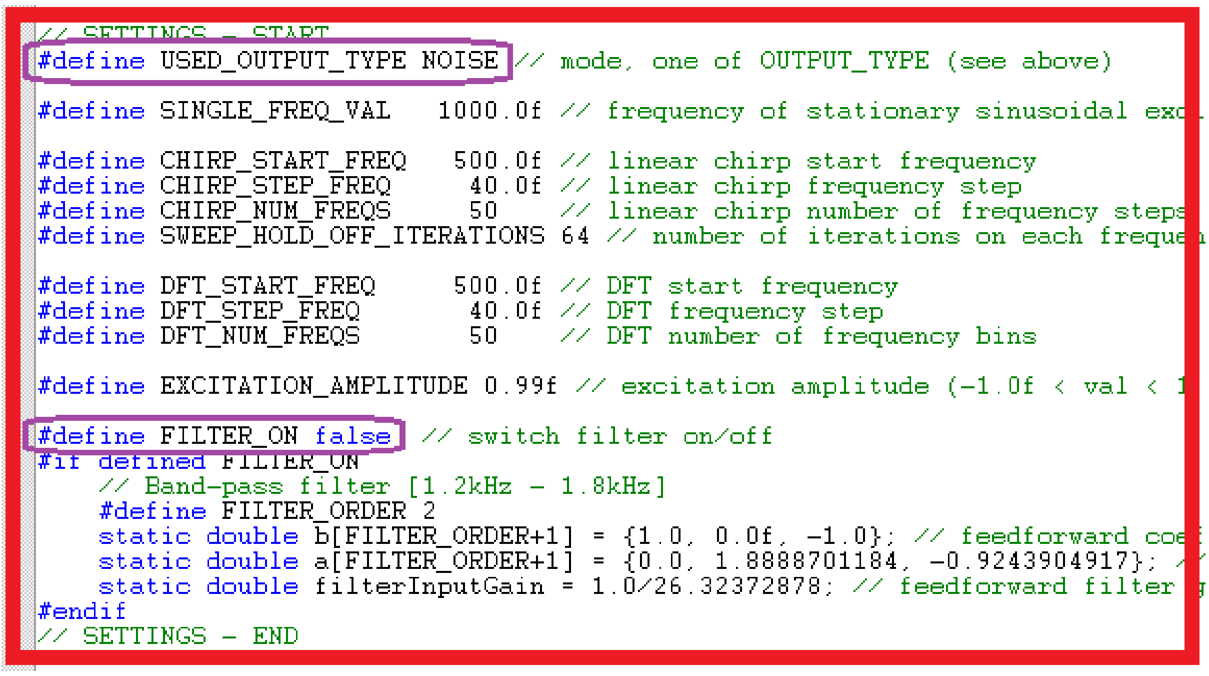
```
// SETTINGS - START
#define USED_OUTPUT_TYPE IMPULSE // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exc

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each freque

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coe
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; /
    static double filterInputGain = 1.0/26.32372878; // feedforward filter
#endif
// SETTINGS - END
```

3b) Set the DSP into the mode of “NOISE EXCITATION, FILTER OFF”


Analog Devices VisualDSP++ - [Target: ADSP-21489 Platform 0] - [Project: CAS-TuneMeasSystem] - [Settings]
File Edit Session View Project Register Memory Debug Settings Tools Window Help
CAS-TuneMeasSystem
Debug
Cumulative power spectral density
excitation (purple) & response (cyan)
Cumulative Power [a.u.]
Frequency [Hz]
BTC Data Lost
Live
Line Plot
Signal values
excitation (purple) & response (cyan)
% of max voltage
Sample idx
Ready
Line 14, Col 31
NUM

```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE NOISE // mode. one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exc

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each freque

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coe
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; /
    static double filterInputGain = 1.0/26.32372878; // feedforward filter
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
extern void buildTrigTables(void);
#define _BTCAMPDFTCH_        0
#define _BTCAMPDFTCUMCH_     1
#define _BTCPHDFTCH_         2
#define _BTCAMPMIXCH_        3
#define _BTCPHMIXCH_         4
#define _BTCEXCTFREQCH_      5
#define _BTCINDATACH_        6
#define _BTCOUTDATACH_       7
#define _BTCAMPDFTOUTCH_     8
#define _BTCAMPDFTOUTCUMCH_  9
#define BTC_EXTRA_SPACE 4
extern float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcIn[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcOut[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

#define BTC_FREQ_BUF_LEN 1024
```

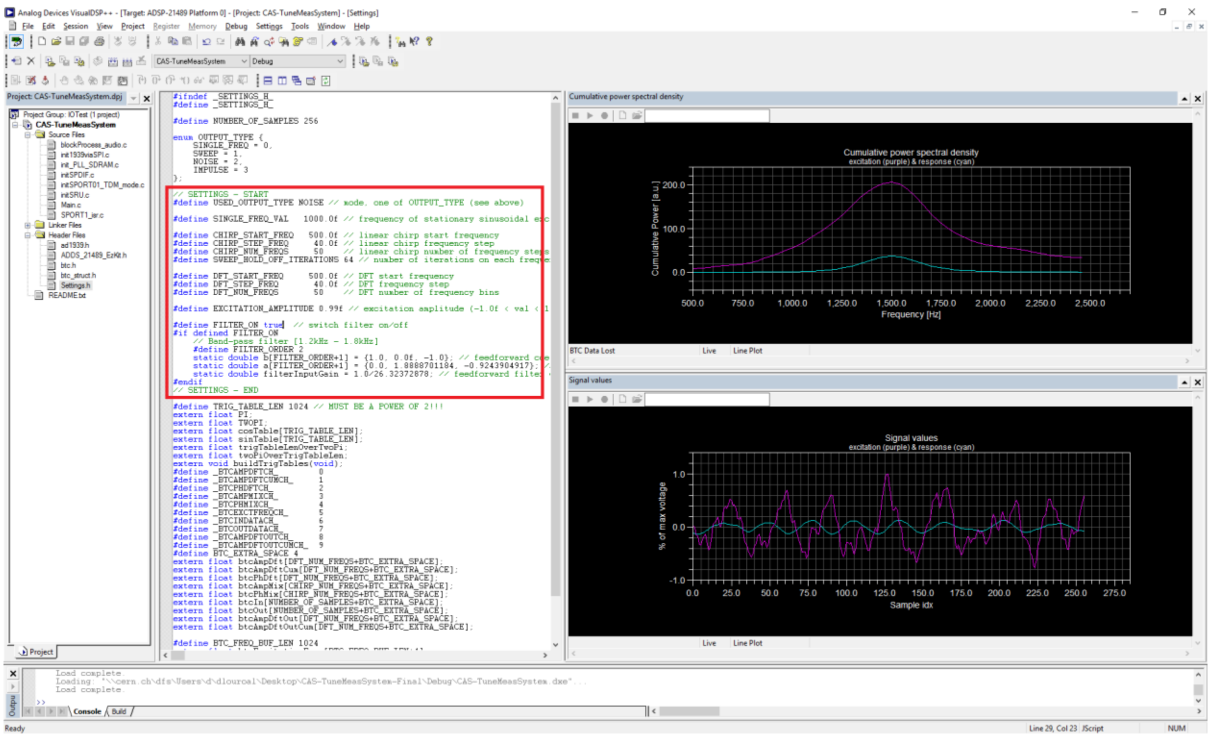
```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
```



```
// SETTINGS - START
#define USED_OUTPUT_TYPE NOISE // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exc

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each freque

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1

#define FILTER_ON false  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coe
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; /
    static double filterInputGain = 1.0/26.32372878; // feedforward filter
#endif
// SETTINGS - END
```

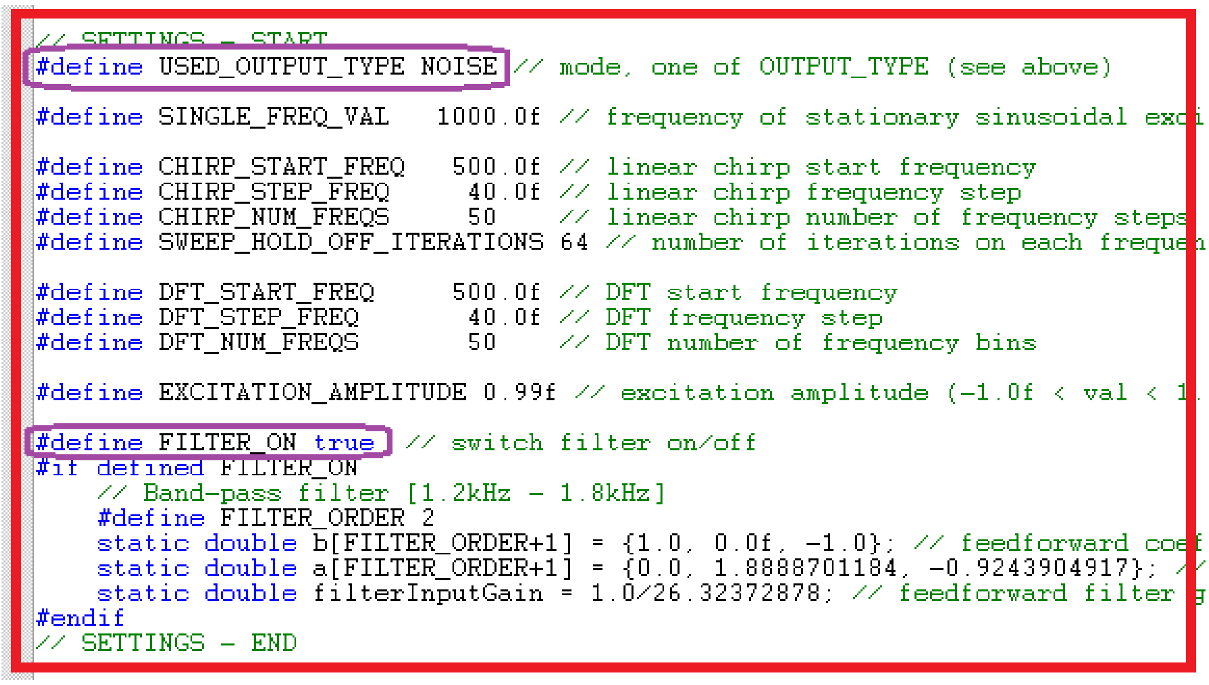
The DSP will work in an endless loop, each time sending excitation pulses, collecting the data and computing a discrete Fourier transform in the frequency range 500…2500 Hz. The amplitude spectrum of the excitation and the beam are displayed.
In the top two graphs shot by shot, and the bottom two plots as accumulated spectra.
Interpret the results.
If wanted you can also observe the time domain data with the oscilloscope, but it is difficult to get a meaningful measurement.

3c) Set the DSP into the mode of “NOISE EXCITATION, FILTER ON”



```
#ifndef _SETTINGS_H_
#define _SETTINGS_H_

#define NUMBER_OF_SAMPLES 256

enum OUTPUT_TYPE {
    SINGLE_FREQ = 0,
    SWEEP = 1,
    NOISE = 2,
    IMPULSE = 3
};

// SETTINGS - START
#define USED_OUTPUT_TYPE NOISE // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exc

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each freque

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1

#define FILTER_ON true  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coe
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; /
    static double filterInputGain = 1.0/26.32372878; // feedforward filte
#endif
// SETTINGS - END

#define TRIG_TABLE_LEN 1024 // MUST BE A POWER OF 2!!!
extern float PI;
extern float TWOPI;
extern float cosTable[TRIG_TABLE_LEN];
extern float sinTable[TRIG_TABLE_LEN];
extern float trigTableLenOverTwoPi;
extern float twoPiOverTrigTableLen;
extern void buildTrigTables(void);
#define _BTCAMPDFTCH_         0
#define _BTCAMPDFTCUMCH_      1
#define _BTCPHDFTCH_          2
#define _BTCAMPMIXCH_         3
#define _BTCPHMIXCH_          4
#define _BTCEXCTFREQCH_       5
#define _BTCINDATACH_         6
#define _BTCOUTDATACH_        7
#define _BTCAMPDFTOUTCH_      8
#define _BTCAMPDFTOUTCUMCH_   9
#define BTC_EXTRA_SPACE 4
extern float btcAmpDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhDft[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcPhMix[CHIRP_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcIn[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcOut[NUMBER_OF_SAMPLES+BTC_EXTRA_SPACE];
extern float btcAmpDftOut[DFT_NUM_FREQS+BTC_EXTRA_SPACE];
extern float btcAmpDftOutCum[DFT_NUM_FREQS+BTC_EXTRA_SPACE];

#define BTC_FREQ_BUF_LEN 1024
```

```
Load complete.
Loading: "\\cern.ch\dfs\Users\d\dlouroal\Desktop\CAS-TuneMeasSystem-Final\Debug\CAS-TuneMeasSystem.dxe"...
Load complete.
```



```
// SETTINGS - START
#define USED_OUTPUT_TYPE NOISE // mode, one of OUTPUT_TYPE (see above)

#define SINGLE_FREQ_VAL   1000.0f // frequency of stationary sinusoidal exci

#define CHIRP_START_FREQ   500.0f // linear chirp start frequency
#define CHIRP_STEP_FREQ     40.0f // linear chirp frequency step
#define CHIRP_NUM_FREQS     50    // linear chirp number of frequency steps
#define SWEEP_HOLD_OFF_ITERATIONS 64 // number of iterations on each frequen

#define DFT_START_FREQ     500.0f // DFT start frequency
#define DFT_STEP_FREQ       40.0f // DFT frequency step
#define DFT_NUM_FREQS       50    // DFT number of frequency bins

#define EXCITATION_AMPLITUDE 0.99f // excitation amplitude (-1.0f < val < 1.

#define FILTER_ON true  // switch filter on/off
#if defined FILTER_ON
    // Band-pass filter [1.2kHz - 1.8kHz]
    #define FILTER_ORDER 2
    static double b[FILTER_ORDER+1] = {1.0, 0.0f, -1.0}; // feedforward coef
    static double a[FILTER_ORDER+1] = {0.0, 1.8888701184, -0.9243904917}; //
    static double filterInputGain = 1.0/26.32372878; // feedforward filter g
#endif
// SETTINGS - END
```

The DSP will work in an endless loop, each time sending excitation pulses, collecting the data and computing a discrete Fourier transform in the frequency range 500…2500 Hz.

The only difference compared to exercise 4b is that the noise excitation is pre-filtered around the expected resonant frequency (1600 Hz; jumper for extra C “out”).

Interpret the changes to 3b)

Details on the filter:

The filter has been implemented in order to concentrate the excitation energy of the noise to the frequency range of interest.

The following procedure is implemented in the code:

- apply a digital Butterworth bandpass of 1st order

- determine the peak to peak signal level after filtering

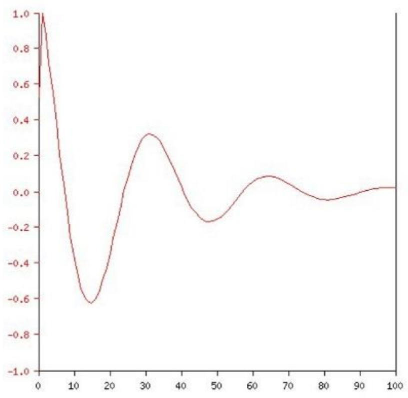
- apply a gain to these values, such that the maximum DAC input (-1…+1) is used.

Possible Discussion: Pros and cons of this method? Alternatives?

Filter data: Sampling frequency 48 kHz

Lower corner frequency 1200 Hz, higher corner frequency 1800 Hz

Below the simulated filter amplitude and phase response plus its response to a kick excitation.

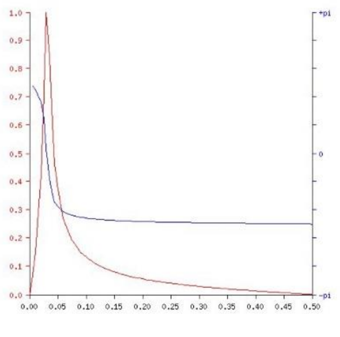


Step response

- *x* axis: time, in samples (i.e. 48000 represents 1 second)
- *y* axis (red): filter response (linear, normalized)

Magnitude (red) and phase (blue) vs. frequency

- *x* axis: frequency, as a fraction of the sampling rate (i.e. 0.5 represents the Nyquist frequency, which is 24000 Hz)
- *y* axis (red): magnitude (linear, normalized)
- *y* axis (blue): phase



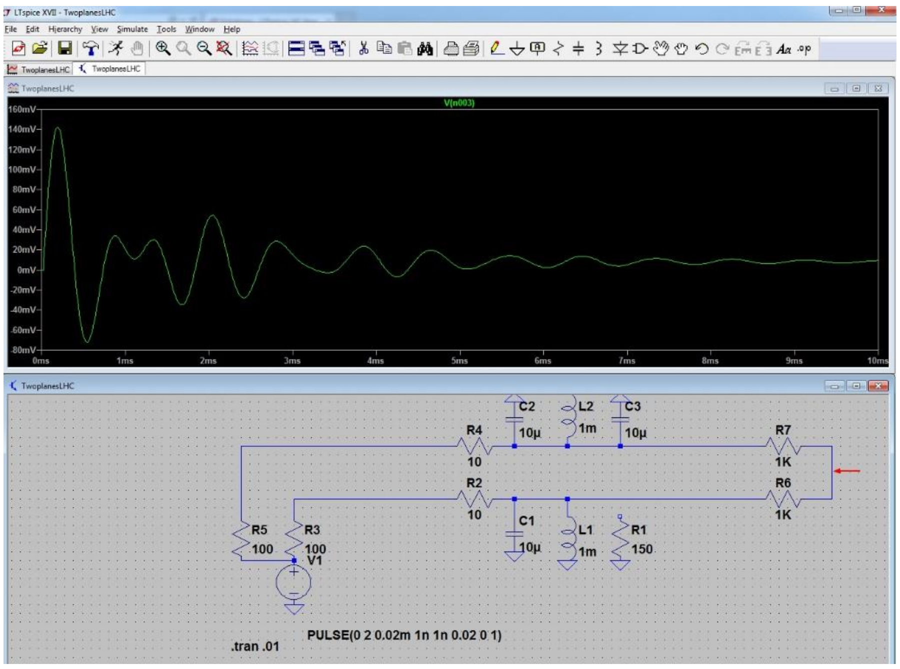
4.) Coupling

Output 1 and 2 always produce the same excitation signal. So you can connect the excitation signal to both planes of the beam in parallel.

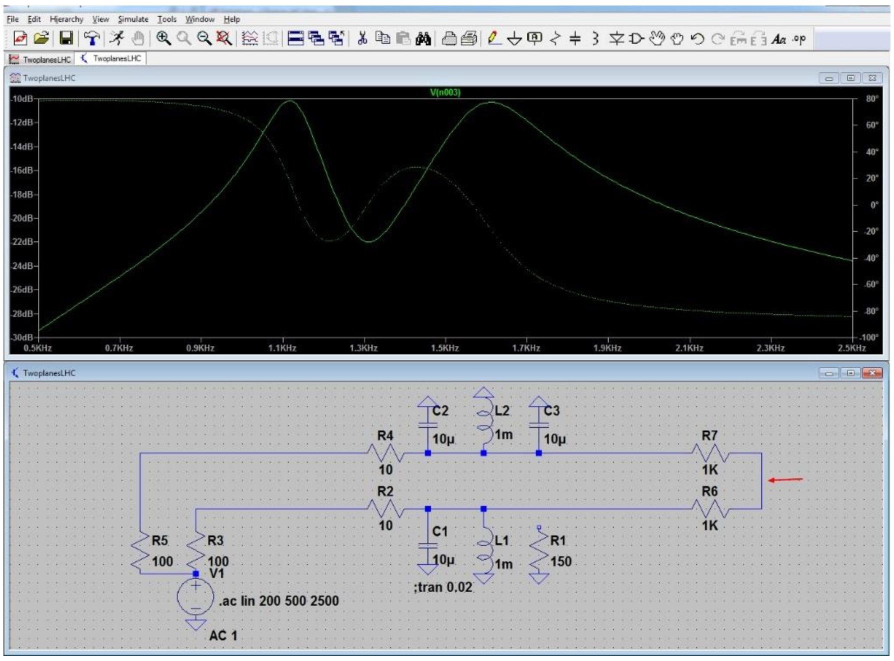
Without the jumper inserted to the middle part of the PCB, both planes are “uncoupled”.

Inserting this jumper produces a simple way of simulating (Strong) coupling of both betatron planes.

Please repeat exercise 3a and 3b with coupled beam. Best results are obtained with both resonant frequencies different. So one jumper “IN” and the same jumper on the other board “OUT”. Below the results in simulation.



```
PULSE(0 2 0.02m 1n 1n 0.02 0 1)
.tran .01
```



```
.ac lin 200 500 2500
;tran 0.02
AC 1
```