ECE PARIS
ÉCOLE D'INGÉNIEURS

Blackfin and Visual DSP++

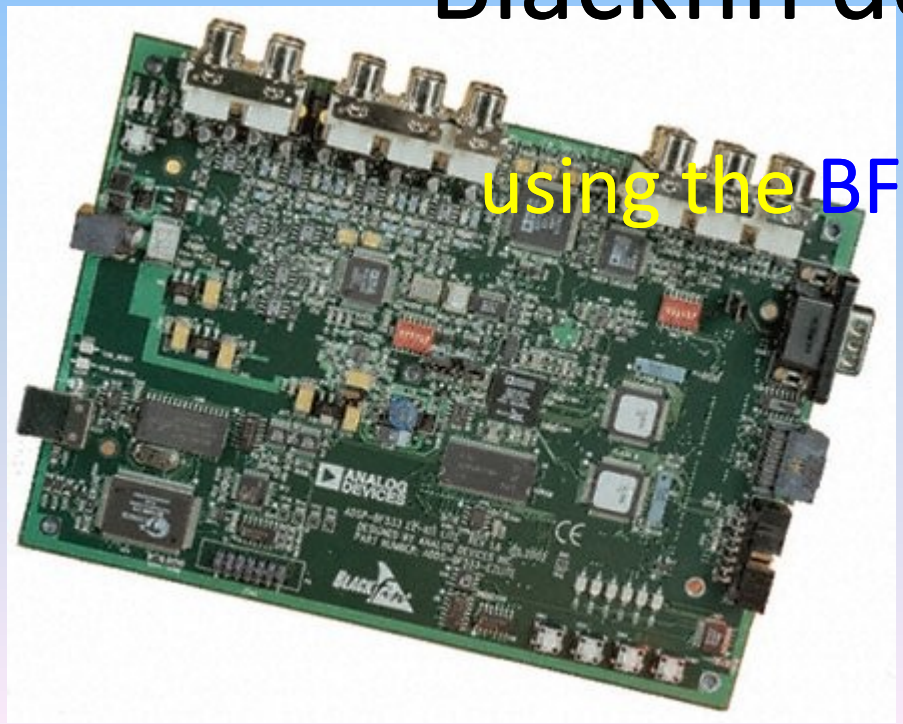
Lab. 1 – Getting to know your Blackfin BF533 DSP

Stefan Ataman

2013 – 2014

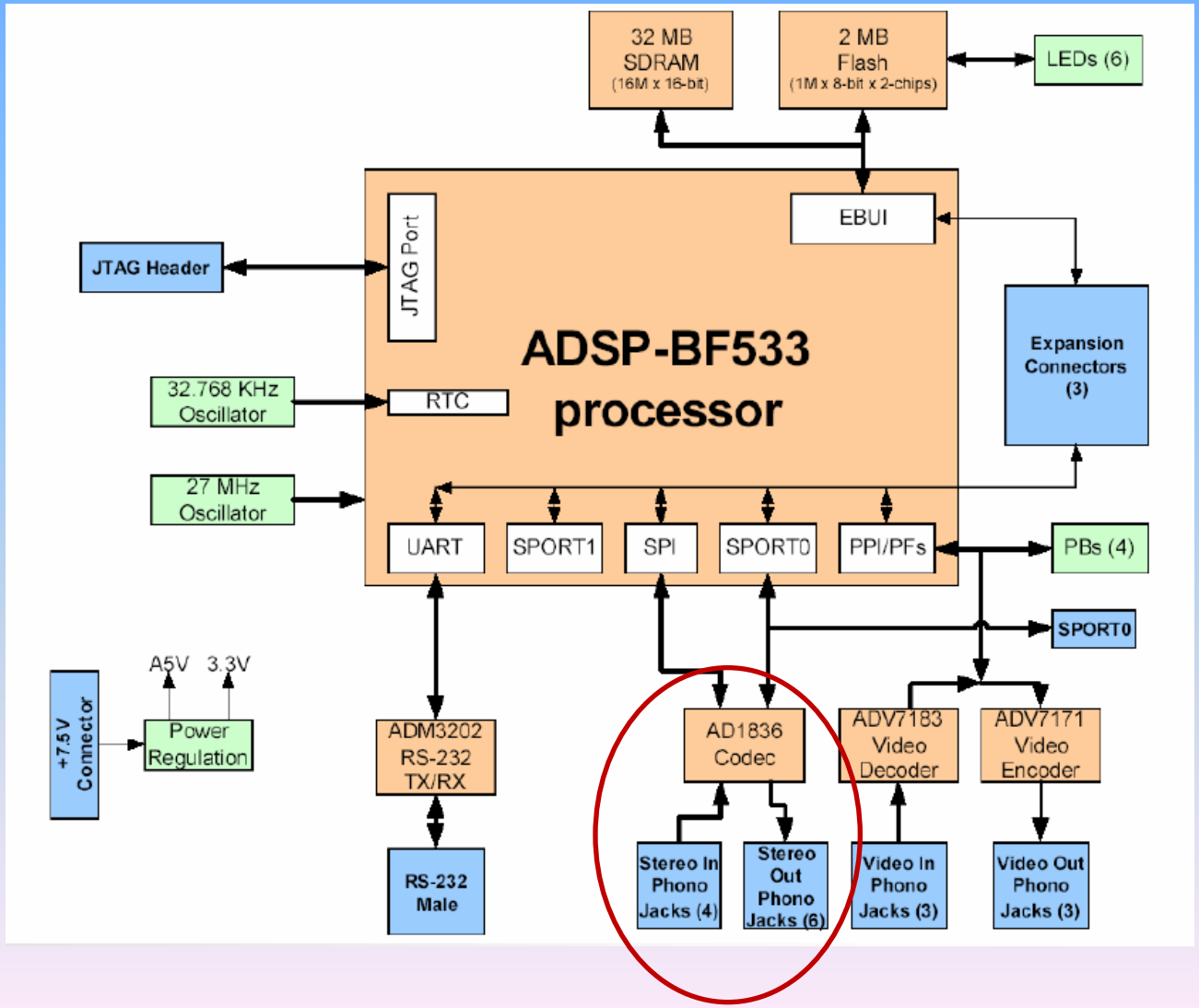
Reminder:
Blackfin development

using the BF533 EZ-KIT Lite



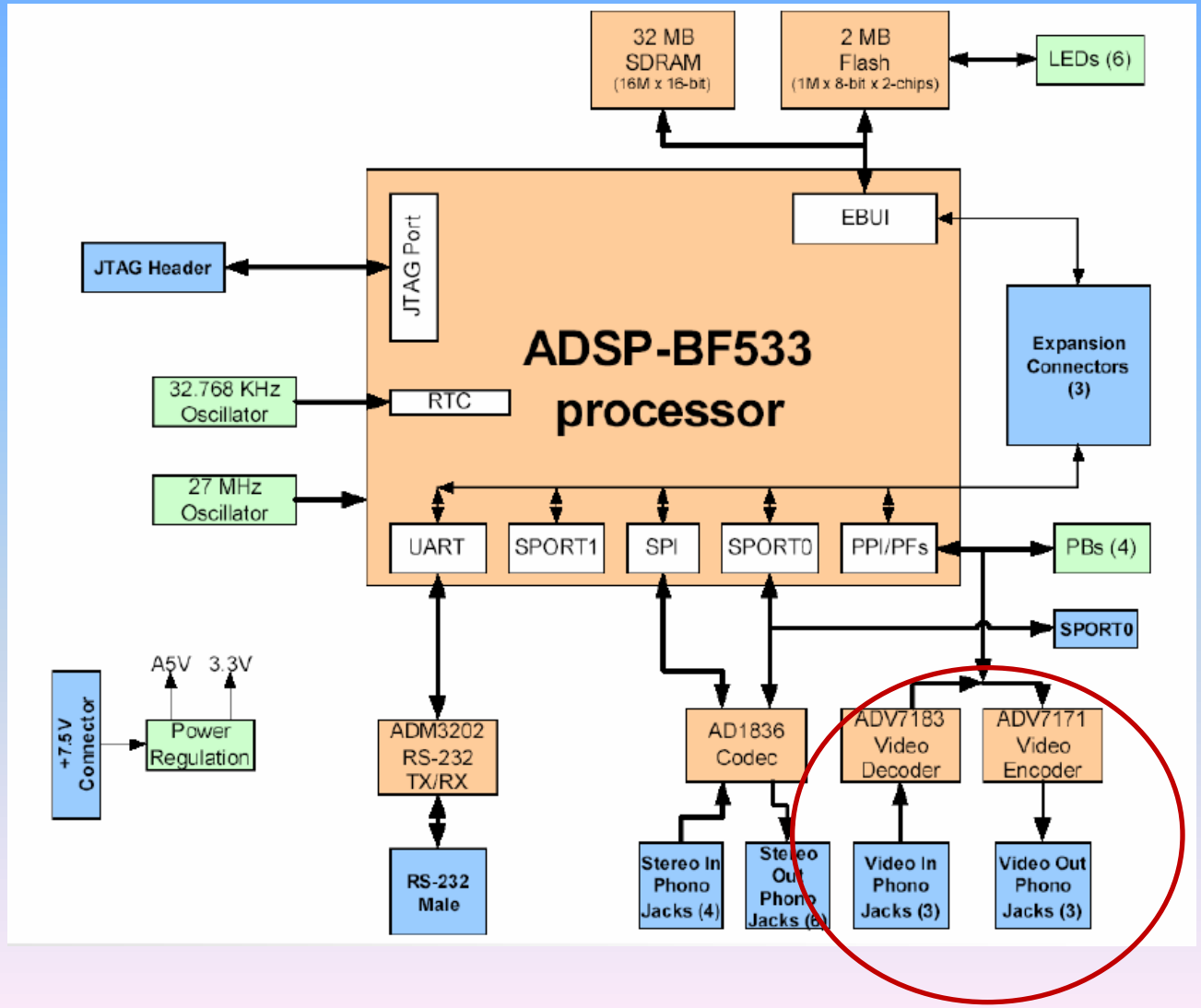
ADSP-BF533 EZ-KIT Lite

Allows you to do:
- real-time audio processing
  - AD1836 codec

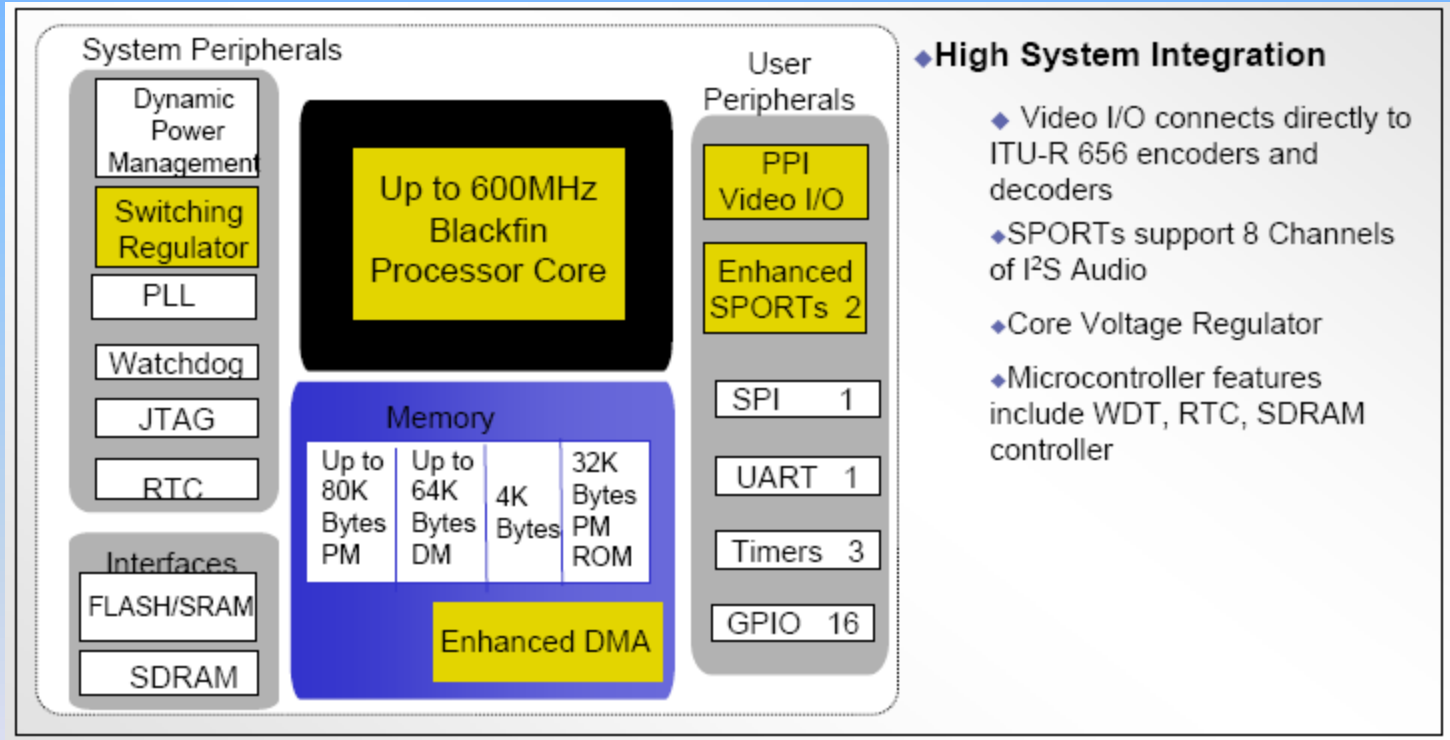


ADSP-BF533 EZ-KIT Lite

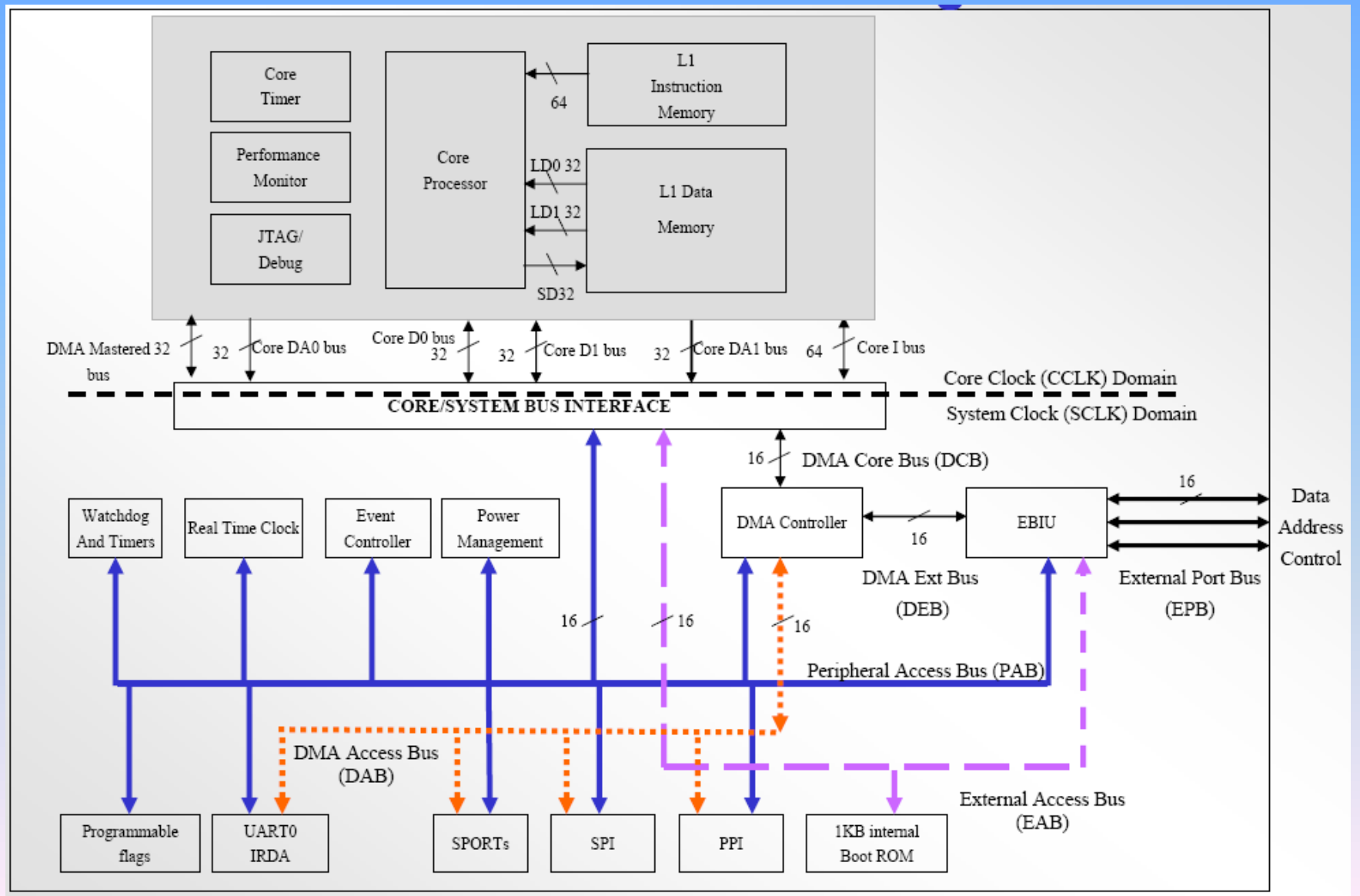
Allows you to do:
• real-time video processing



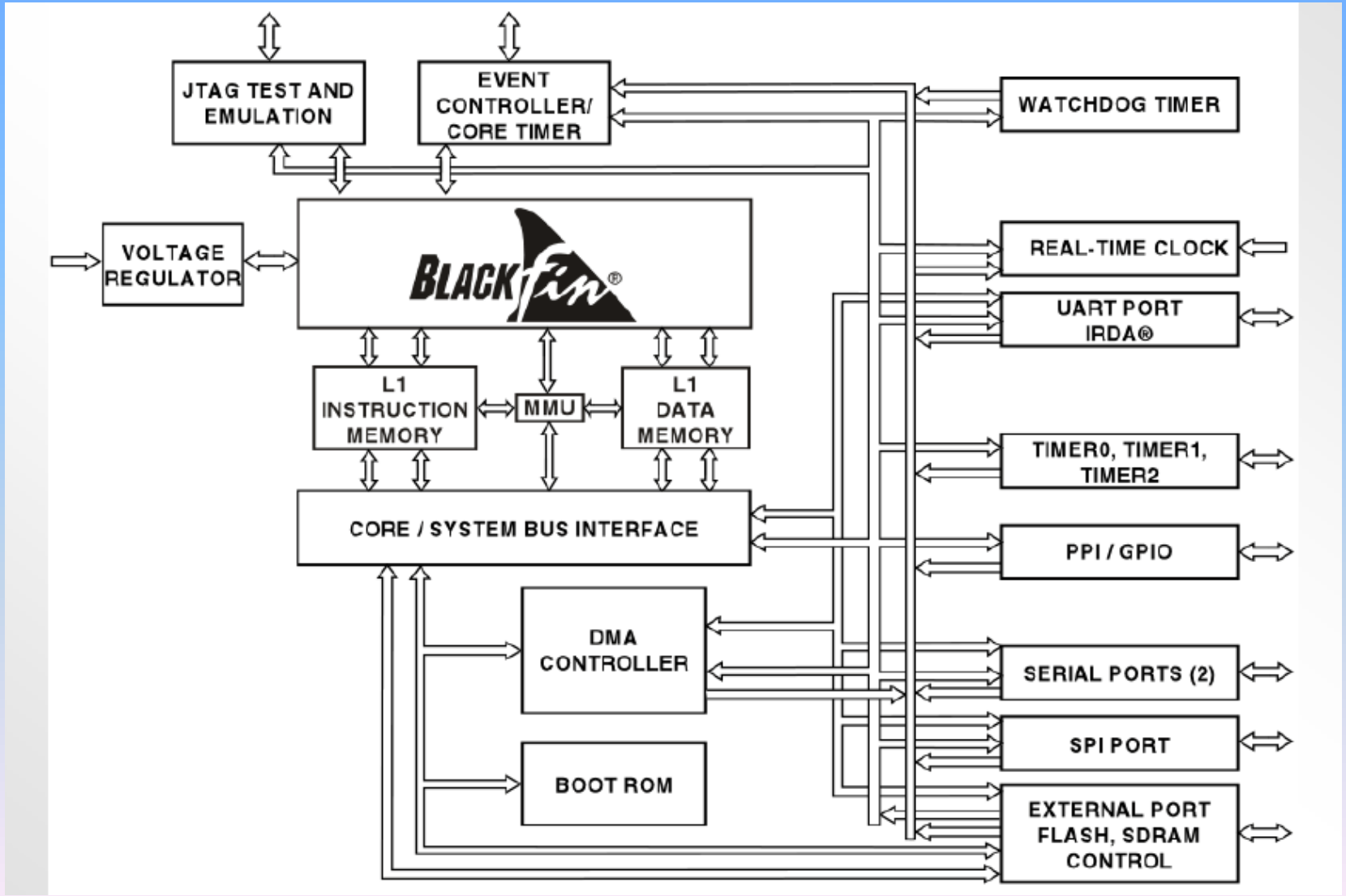
ADSP-BF533/BF532/BF531 Enhanced Blackfin Processors



ADSP-BF533 Block Diagram



ADSP-BF533


JTAG TEST AND EMULATION
EVENT CONTROLLER/ CORE TIMER
WATCHDOG TIMER
VOLTAGE REGULATOR
BLACKfin®
REAL-TIME CLOCK
UART PORT IRDA®
L1 INSTRUCTION MEMORY
MMU
L1 DATA MEMORY
TIMER0, TIMER1, TIMER2
CORE / SYSTEM BUS INTERFACE
PPI / GPIO
DMA CONTROLLER
SERIAL PORTS (2)
SPI PORT
BOOT ROM
EXTERNAL PORT FLASH, SDRAM CONTROL

Core registers

- Data Registers: R0-R7
- Accumulator Registers: A0, A1
- Pointer Registers: P0-P5, FP, SP,USP
- DAG Registers: I0-I3, M0-M3, B0-B3, L0-L3
- Cycle Counters: Cycles, cycles2
- Program Sequencer: SEQSTAT
- System Configuration Register: SYSCFG
- Loop Registers: LT[1:0], LB[1:0], LC[1:0]
- Interrupt Return Registers: RETI, RETX, RETN, RETE

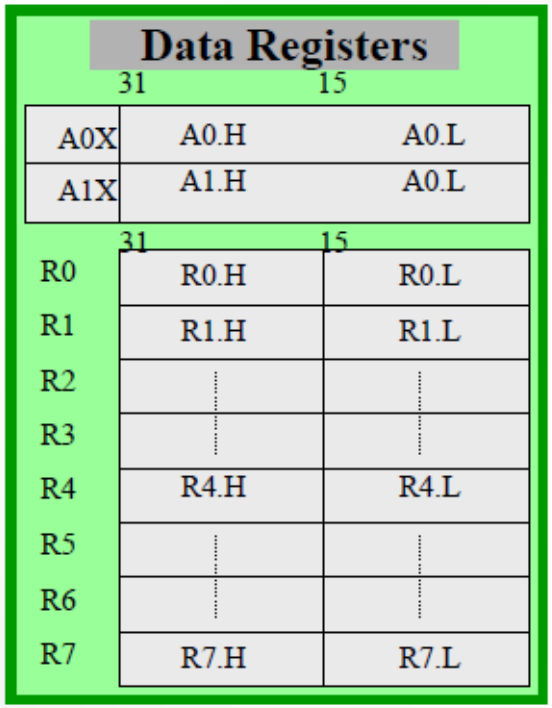
Core Registers

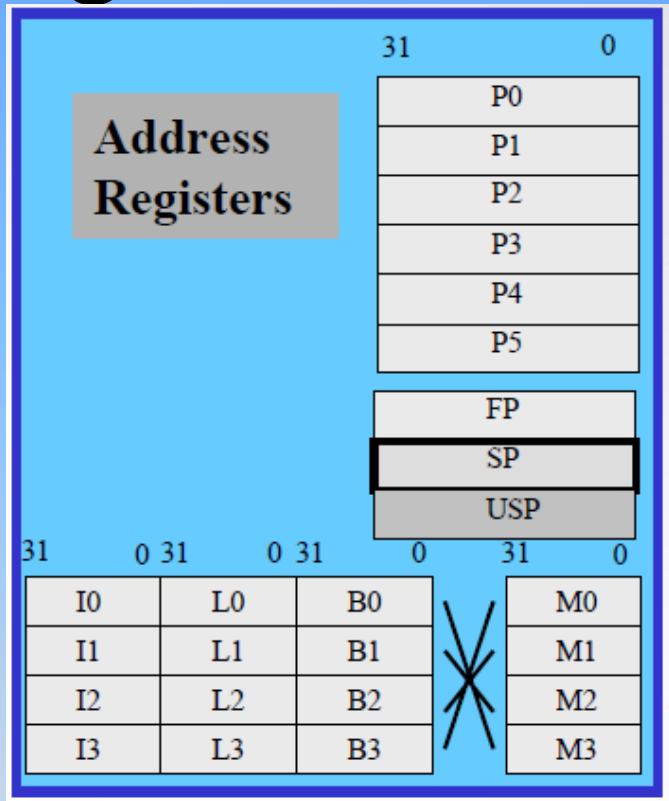
Data Registers:

R0-R7 are referred to as "dreg"

_lo refers to .L and

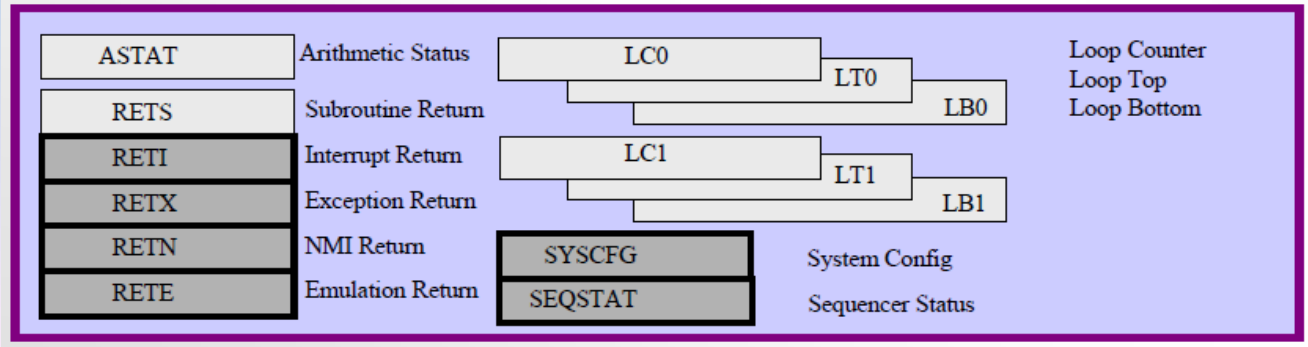
_hi refers to .H





Pointer Registers: *P0-P5* are referred to as "preg"

Index Registers: *I0-I3* are referred to as "ireg"



Shaded registers only accessible in Supervisor mode

Development tool:

- the Visual DSP++
  - graphical IDE
  - C/C++ compiler
  - simulator
  - emulator

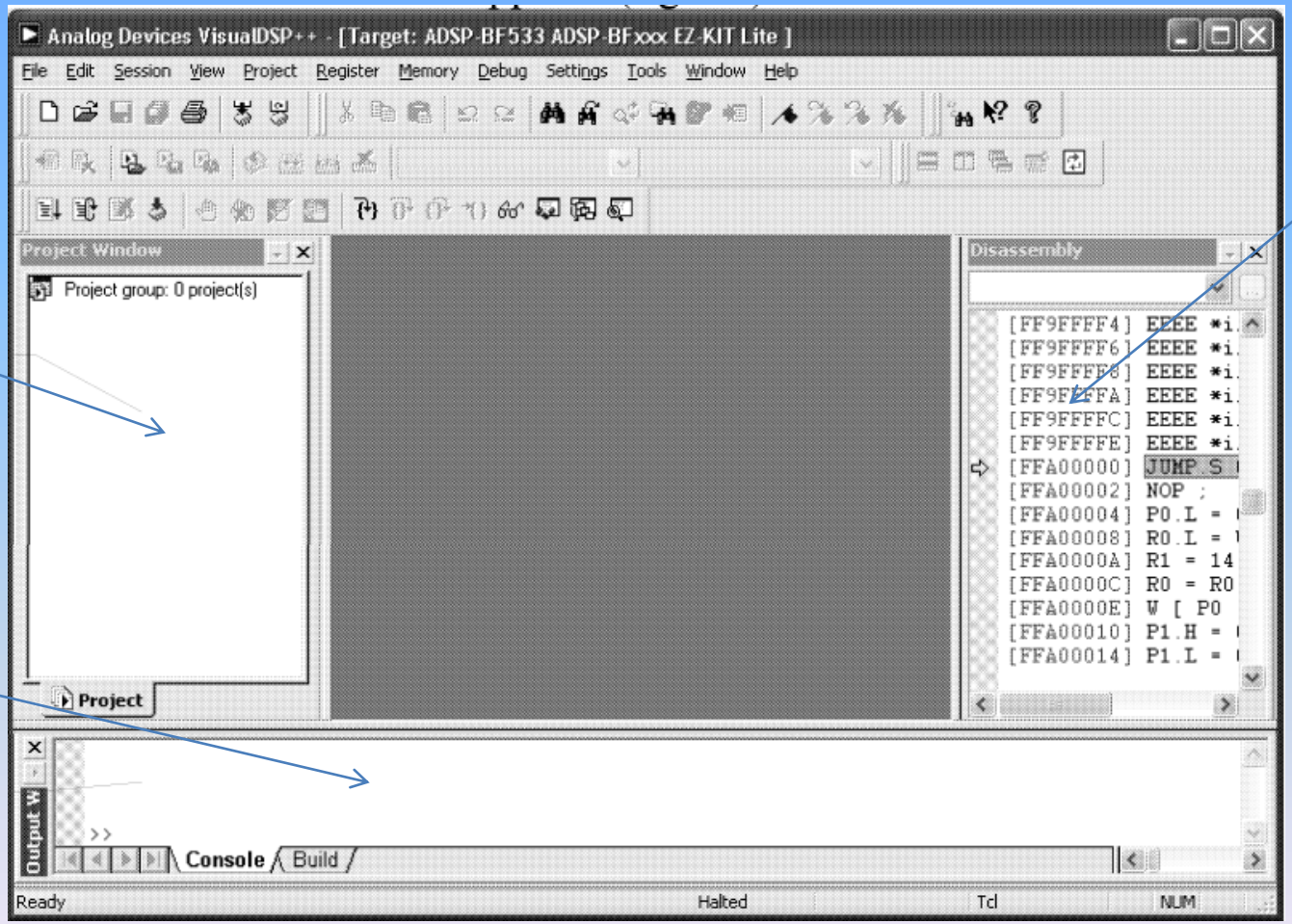
In this lab we want to:

- 1) open a project in C
- 2) open a project in assembler, edit it with the linker
- 3) implement a FIR and graphically display data

1. Open a C project in Visual DSP++ IDE

Getting to know your tool

Visual DSP++ main window

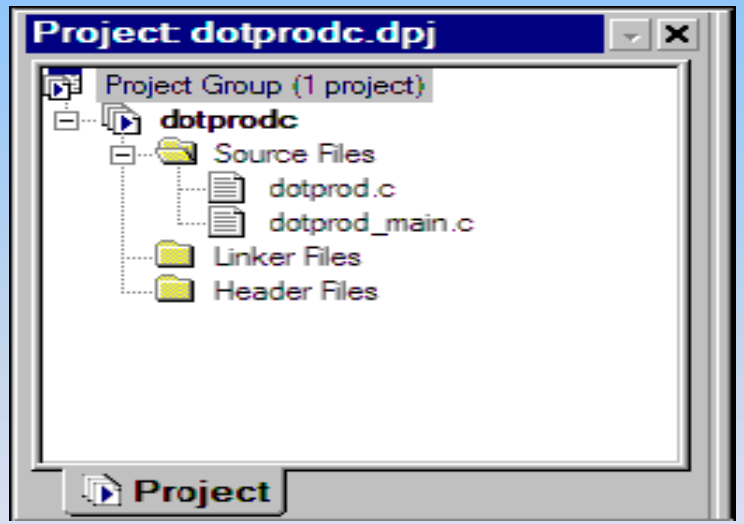


Open a Project

- From the File menu, choose Open and then Project
  - VisualDSP++ displays the Open Project dialog box.
- From campus download the archive :
  - dot_prod_c.zip
- Unpack it to your own folder, *e.g.*
  - d:\your_name_GrXX\dotprodc

The project window

- Double-click the dotprodc project (.dpj) file.
  - VisualDSP++ loads the project in the Project window, as shown



Beware

On the **General** menu, under **General Preferences**, ensure that the following options are selected.

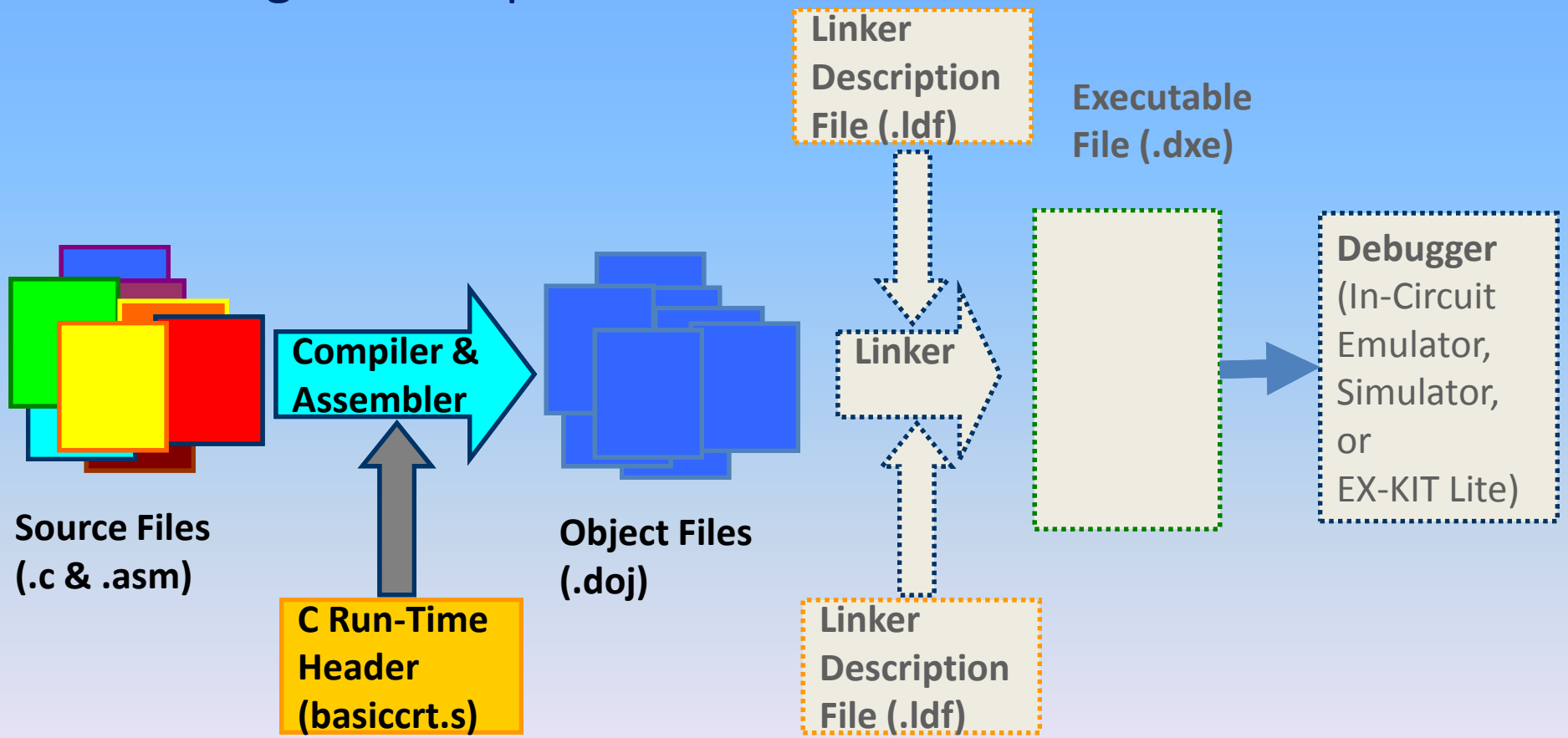
- **Run to main after load**
- **Load executable after build**

The dotprodc project

- The dotprodc project comprises two C language source files:
  - dotprod.c and
  - dotprod_main.c, which define the arrays and calculates their dot products

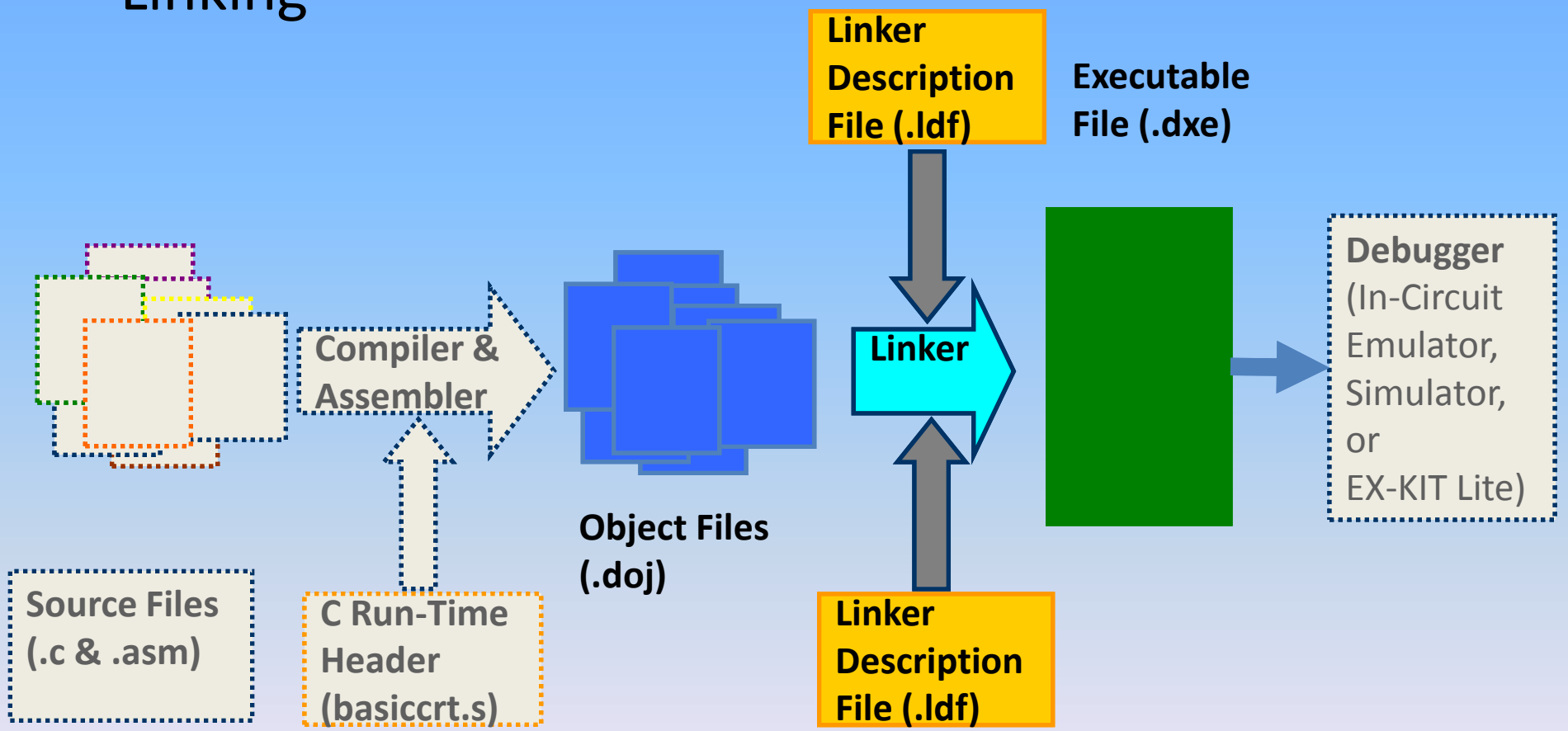
Building the dotprodc Project

Running the Compiler



Building the dotprodc Project

Linking

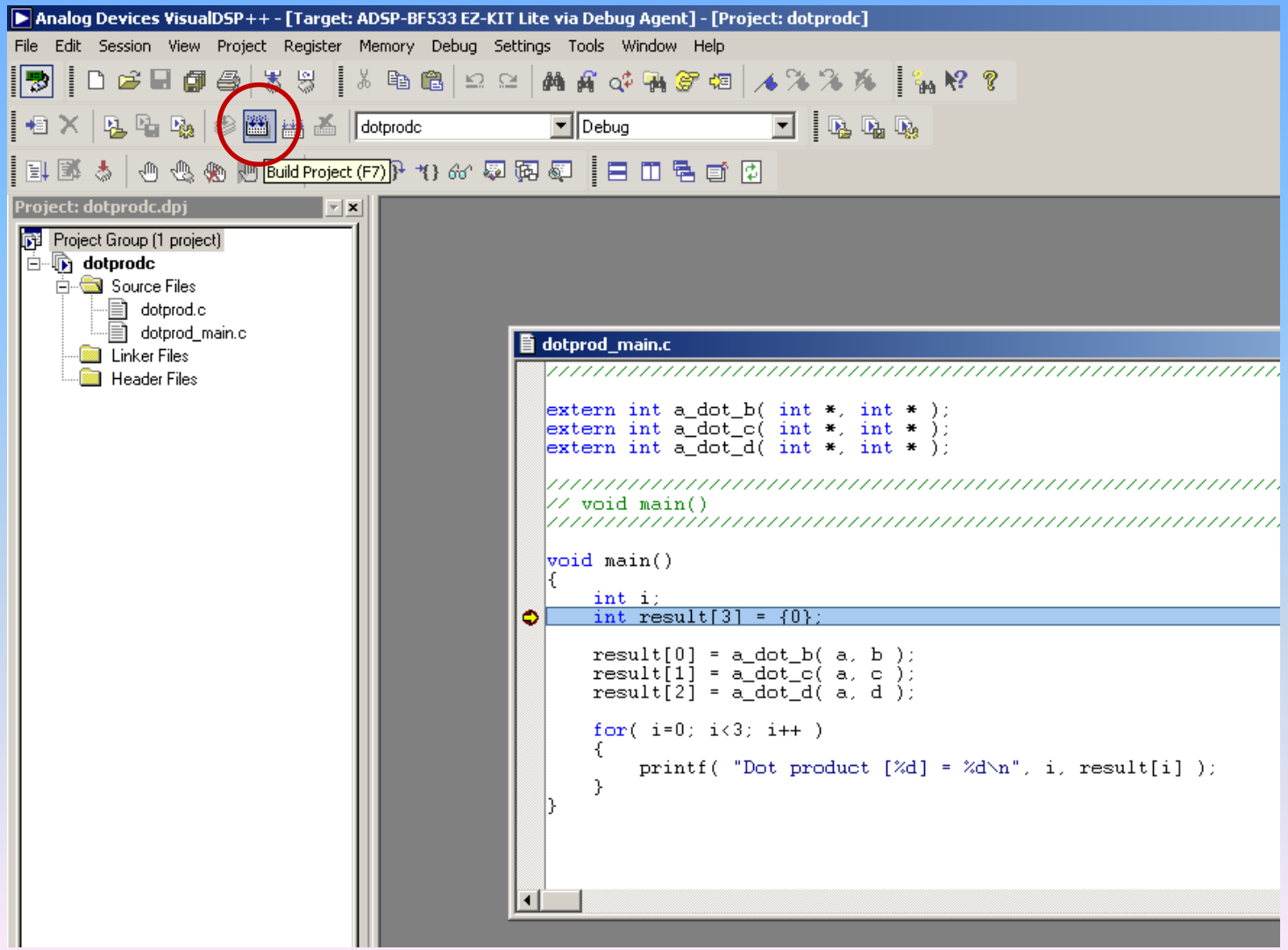


Building the dotprodc Project

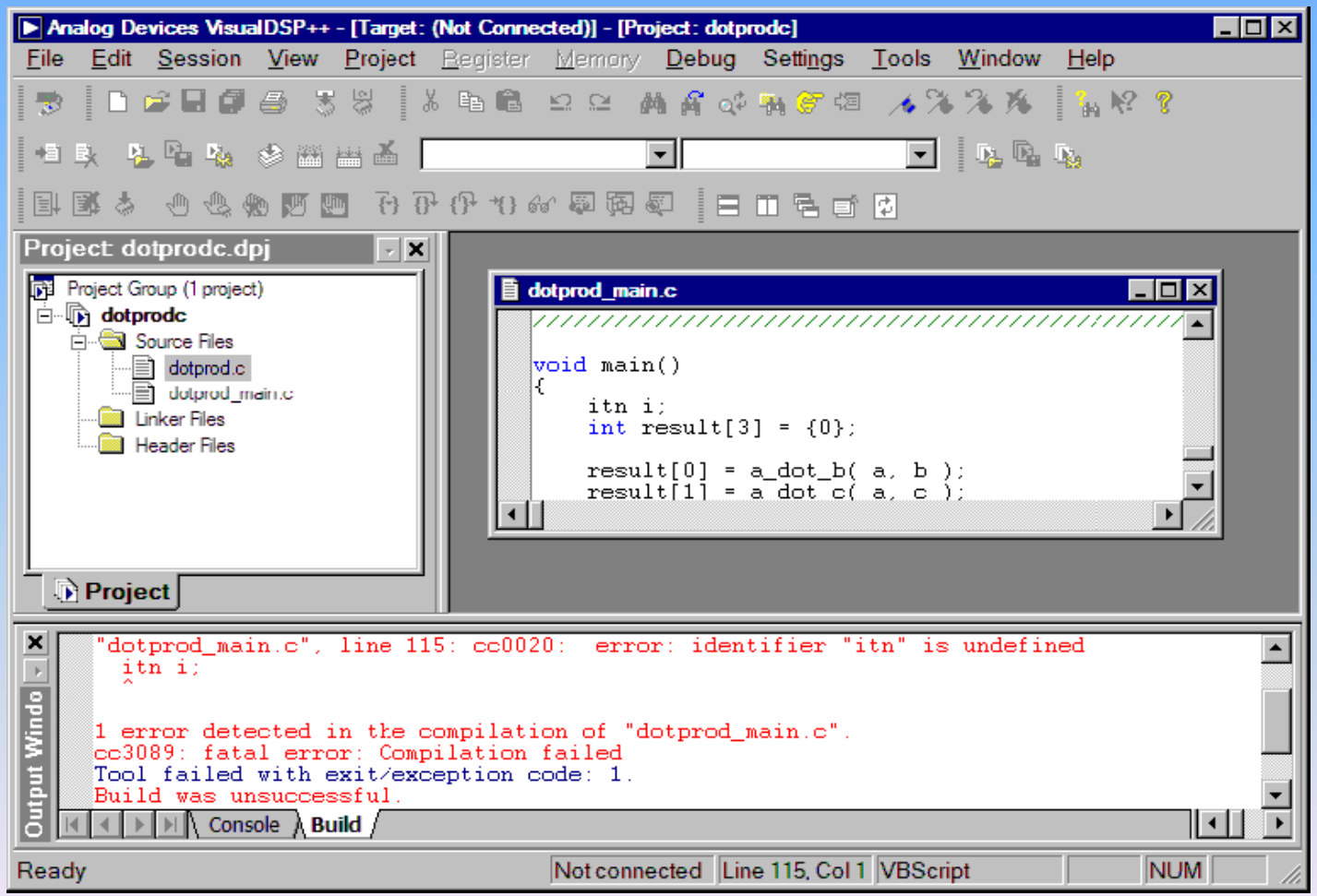
From the **Project** menu, choose **Build Project**.

- VisualDSP++ first checks and updates the project dependencies and then
- Builds the project by using the project source files.
- **Output** window displays status messages.

Building the dotprodc Project



Building the dotprodc Project

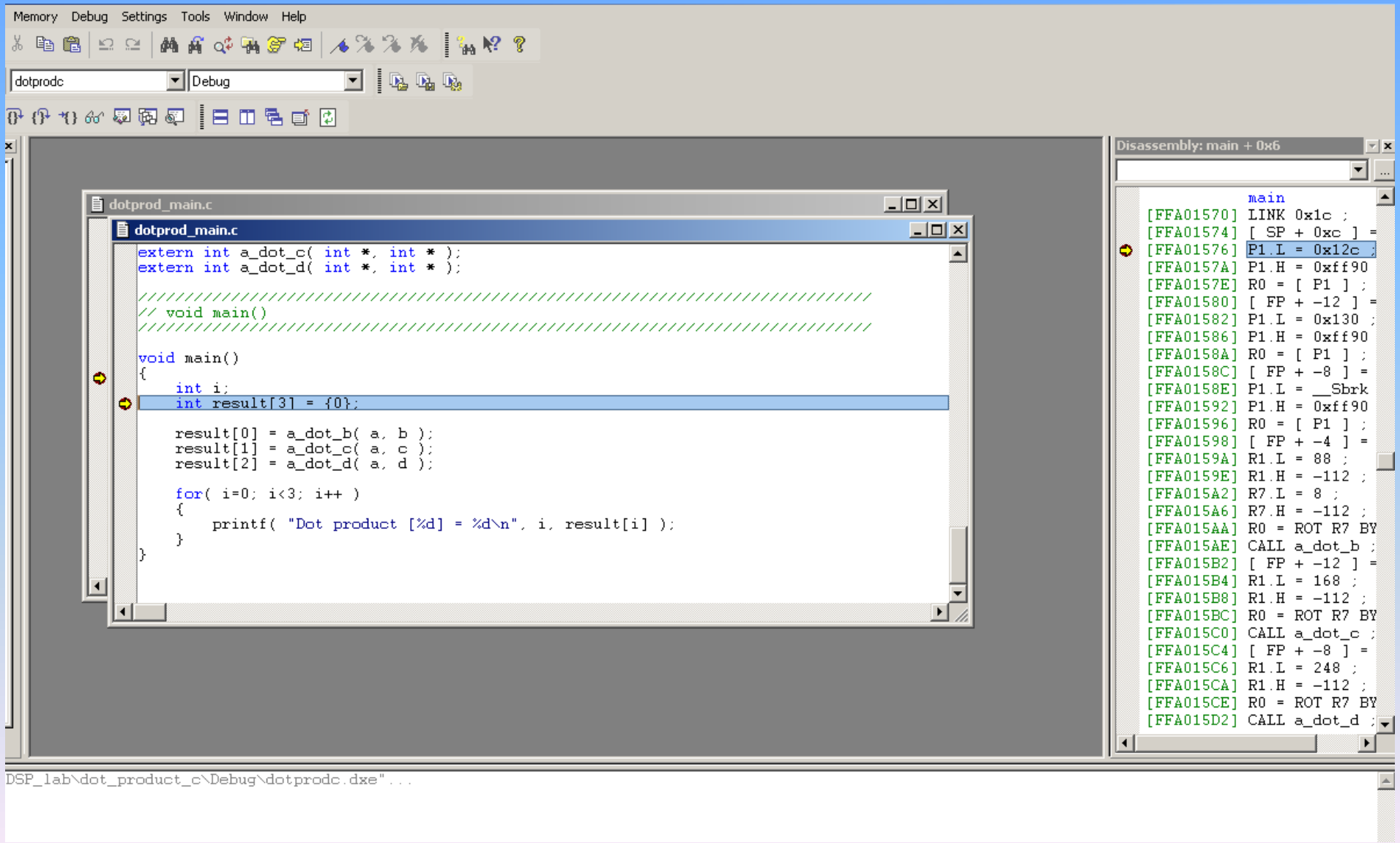


Fix the problem!

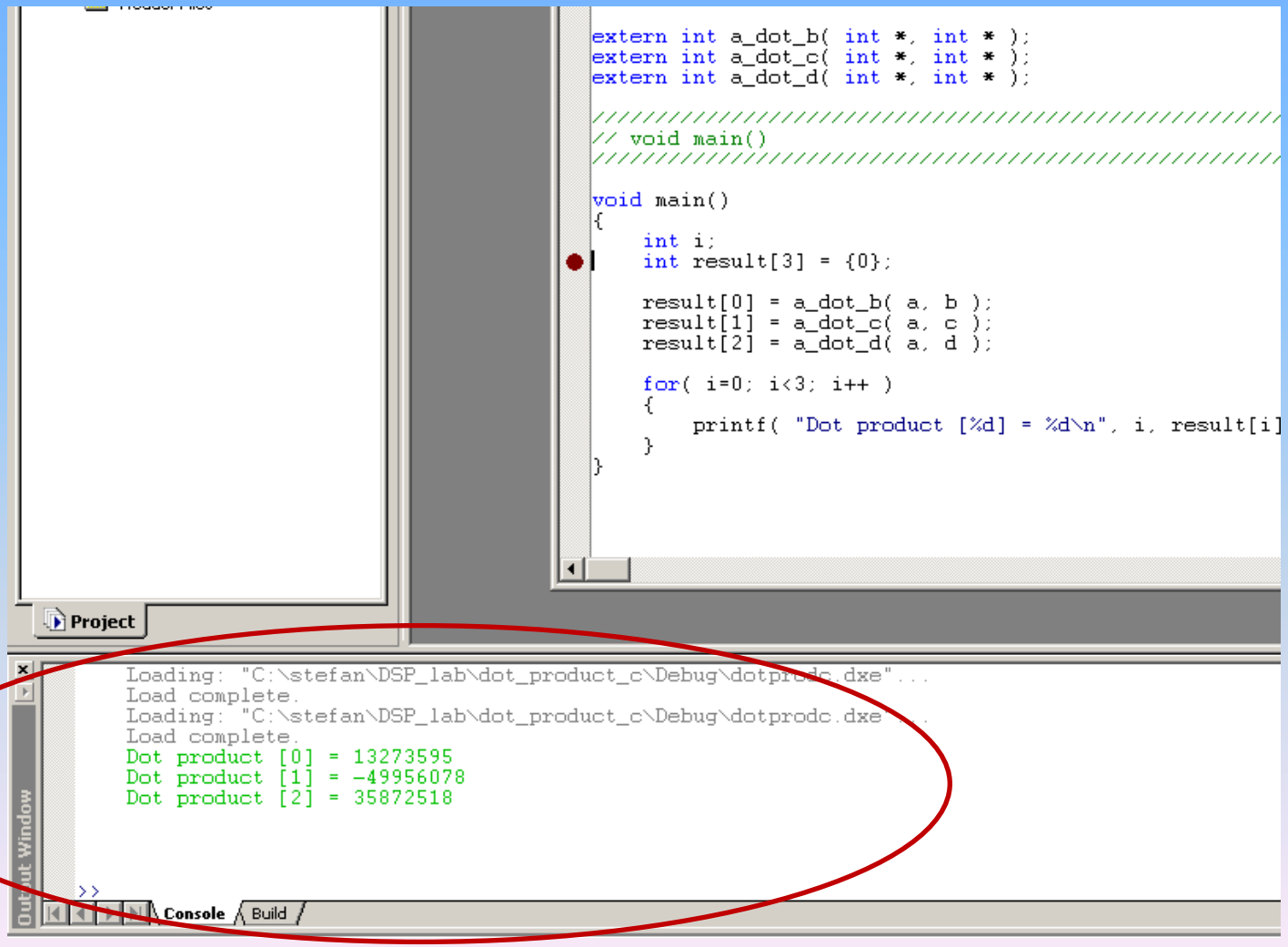
Building the dotprodc Project

- Upon successful build (compile and link steps) example program can be run.

Upon compilation you should see



Hit Run (F5)


extern int a_dot_b(int *, int *);
extern int a_dot_c(int *, int *);
extern int a_dot_d(int *, int *);
// void main()
void main()
{
int i;
int result[3] = {0};
result[0] = a_dot_b(a, b);
result[1] = a_dot_c(a, c);
result[2] = a_dot_d(a, d);
for(i=0; i<3; i++)
{
printf("Dot product [%d] = %d\n", i, result[i]
}
}
Project
Loading: "C:\stefan\DSP_lab\dot_product_c\Debug\dotprodc.dxe"...
Load complete.
Loading: "C:\stefan\DSP_lab\dot_product_c\Debug\dotprodc.dxe"...
Load complete.
Dot product [0] = 13273595
Dot product [1] = -49956078
Dot product [2] = 35872518
Output Window
Console
Build

Dot product

- the result is displayed in the output window
- step by step simulation possible (press F11 or Debug -> Step Into)

Dot product

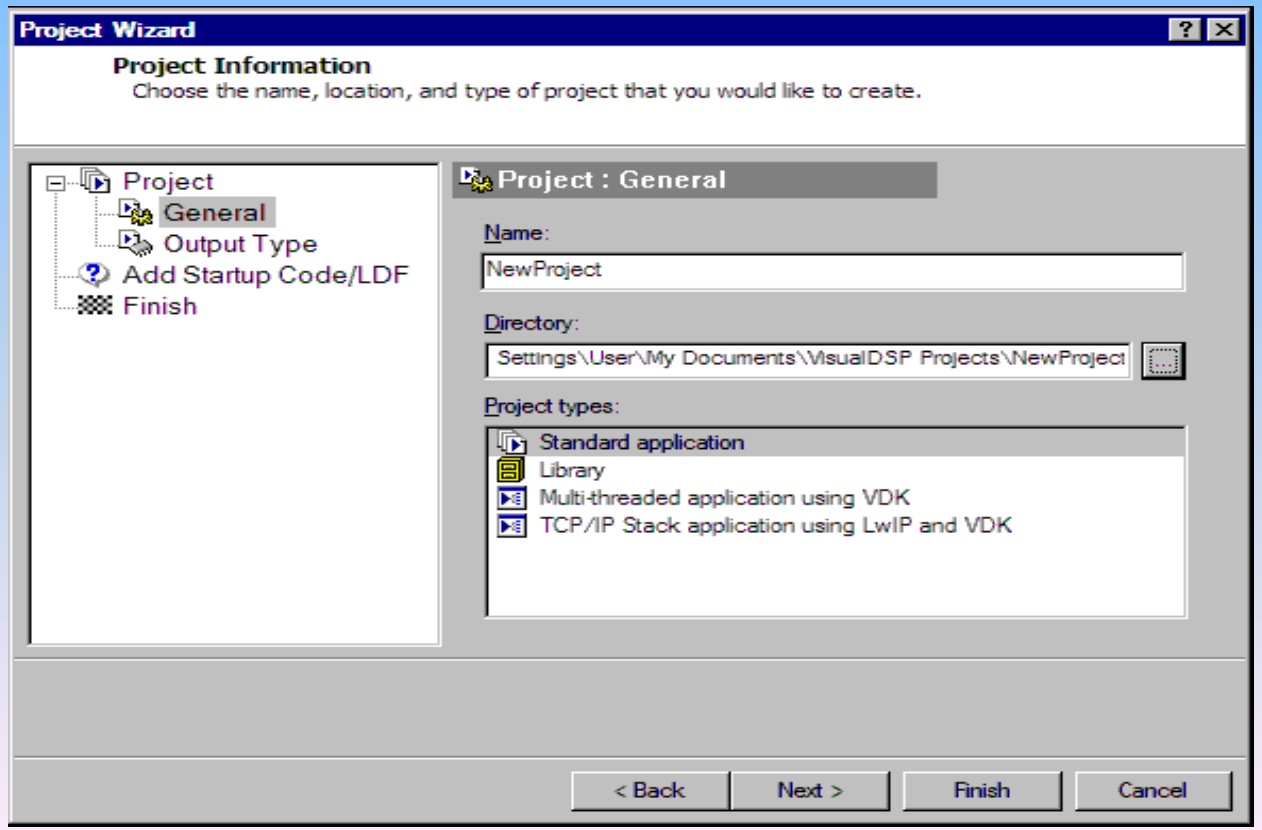
- close the project
  - File => Close => Project dotprodc.dpj

2) Mixing assembler and C. Using the linker

Modifying a C Program Code to Call an Assembly Routine

dot product asm

- From the **File** menu, choose **New** and then **Project** to open the **Project Wizard**



dot product asm

- In the Name field, type dot_product_asm
- From campus download the archive :
  - dotprod_asm.zip
- Unpack it to your own folder, *e.g.*
  - d:\your_name_GrXX\dotprod_asm

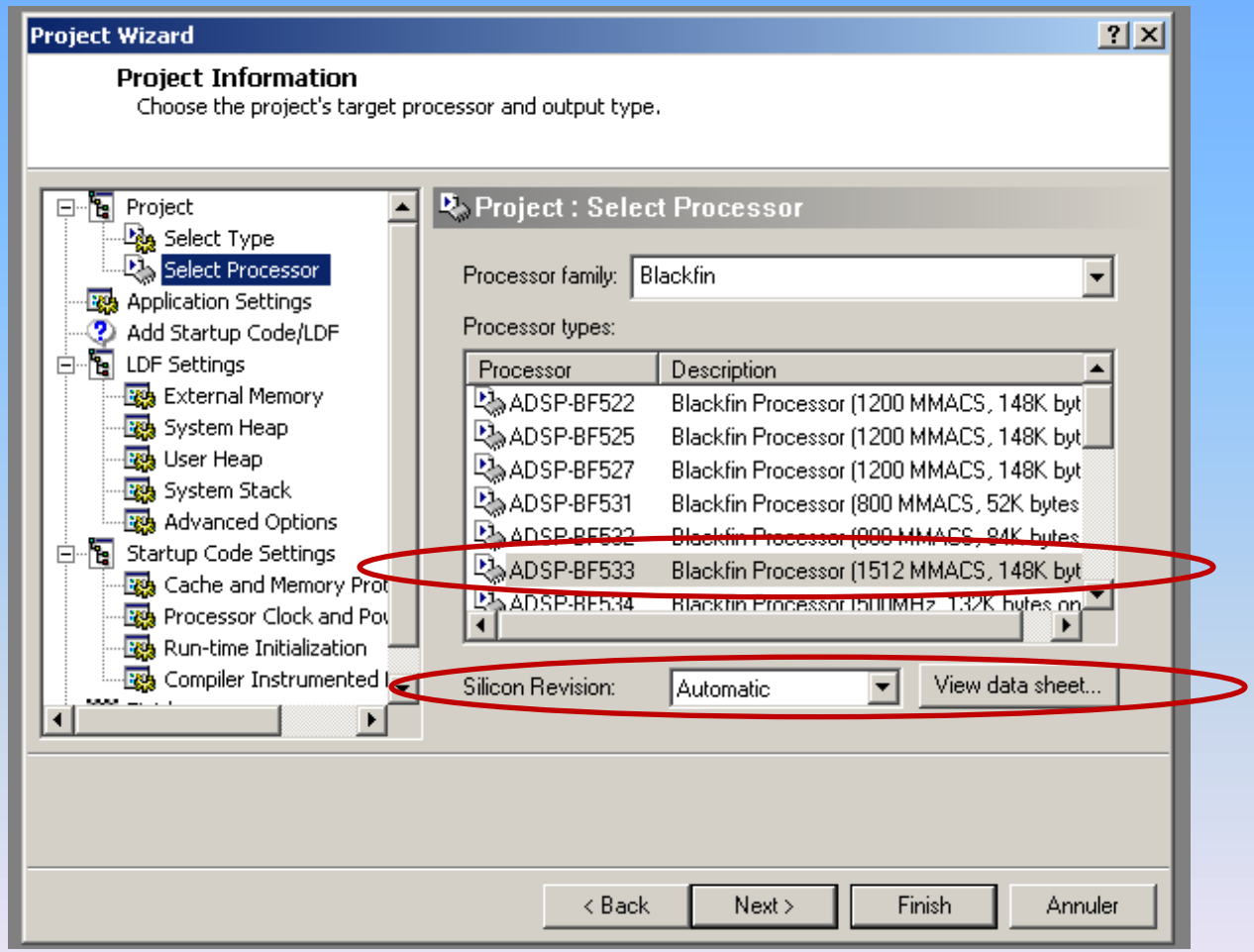
Copy the files

- you should have the files:
  - dotprod.c
  - dotprod_func.asm
  - dotprod_main.c

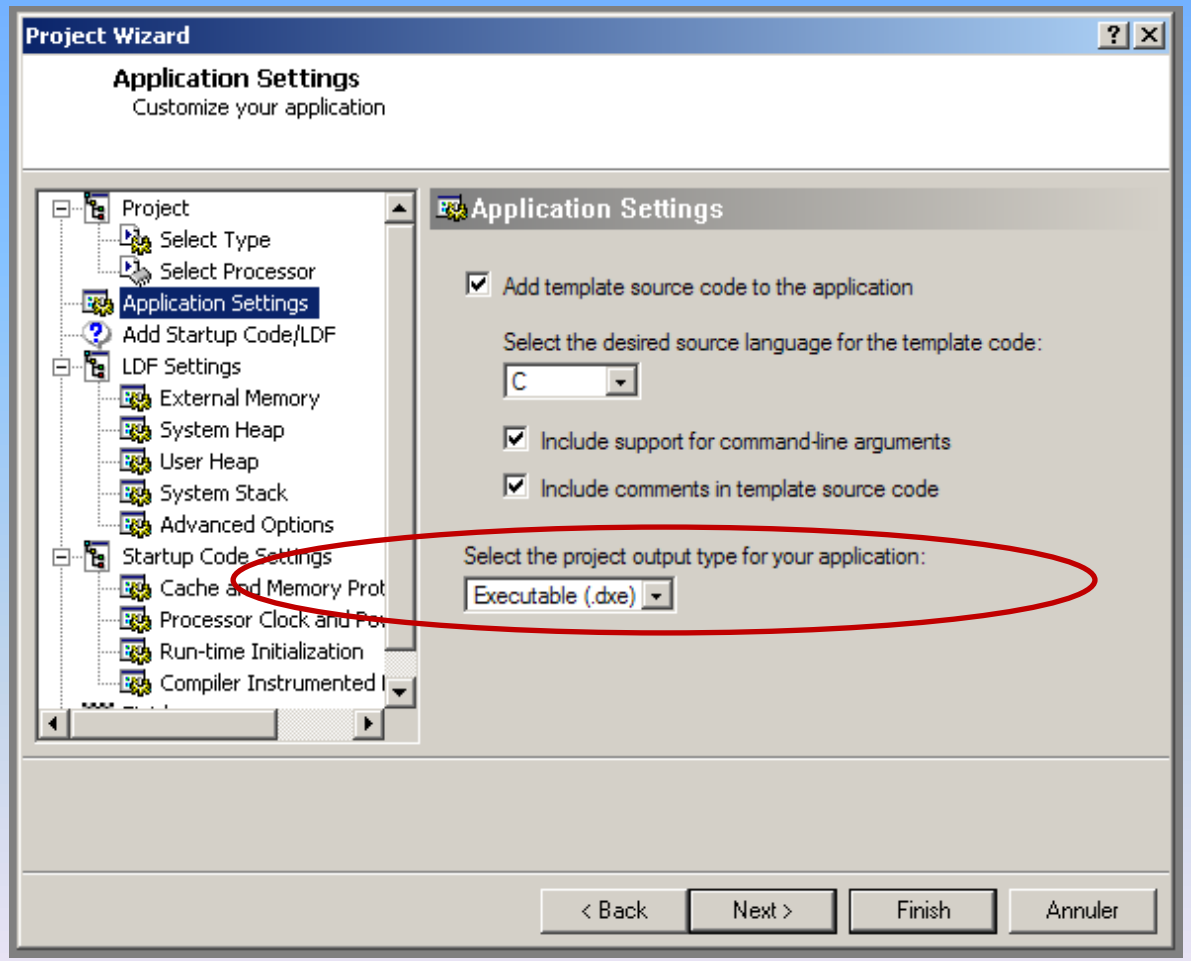
dot product asm

- Click Next to bring up the Output Type page.
- Verify that the Processor type is ADSP-BF533, the Silicon Revision is Automatic, and the Project output file is Executable file.

dot product asm



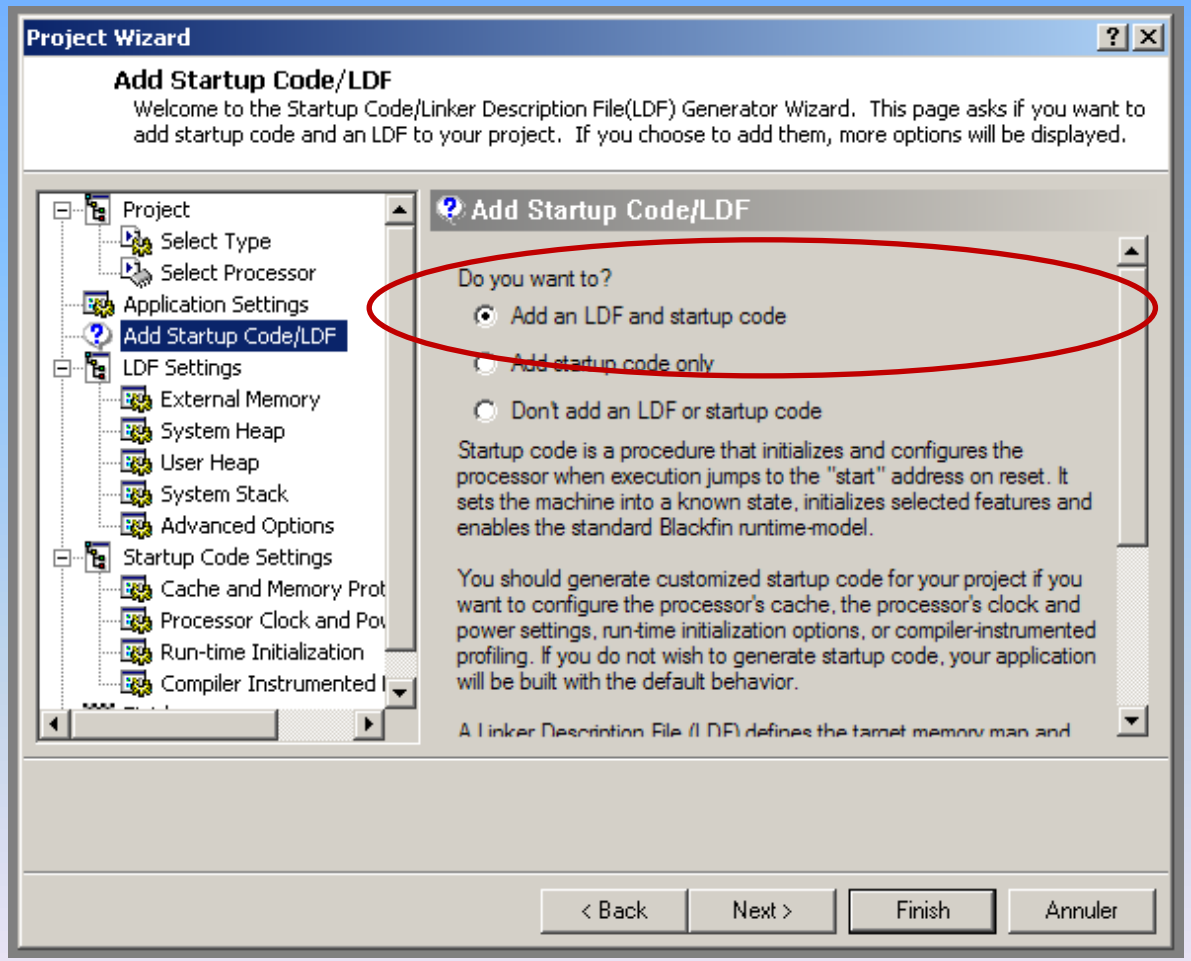
dot product asm



dot product asm

- Click Next to display the Add Startup Code/LDF page.

dot product asm

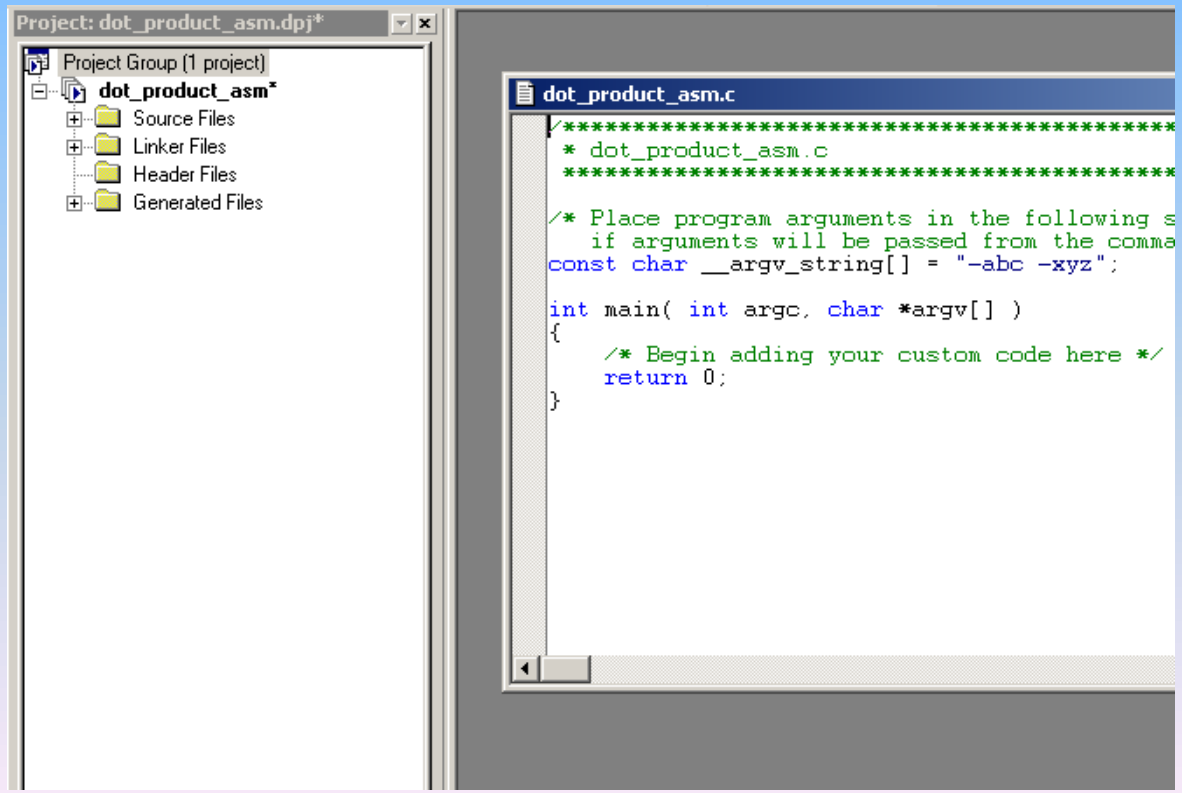


dot product asm

- Select the Add an LDF and startup code option. When this project is created:
  - Startup code that initializes and configures the processor will be added to the project, as well as
  - a Linker Description File that defines the target memory map and the placement of program sections within processor memory.
- Make sure the Add an LDF and startup code option is selected, and click Finish.

dot product asm

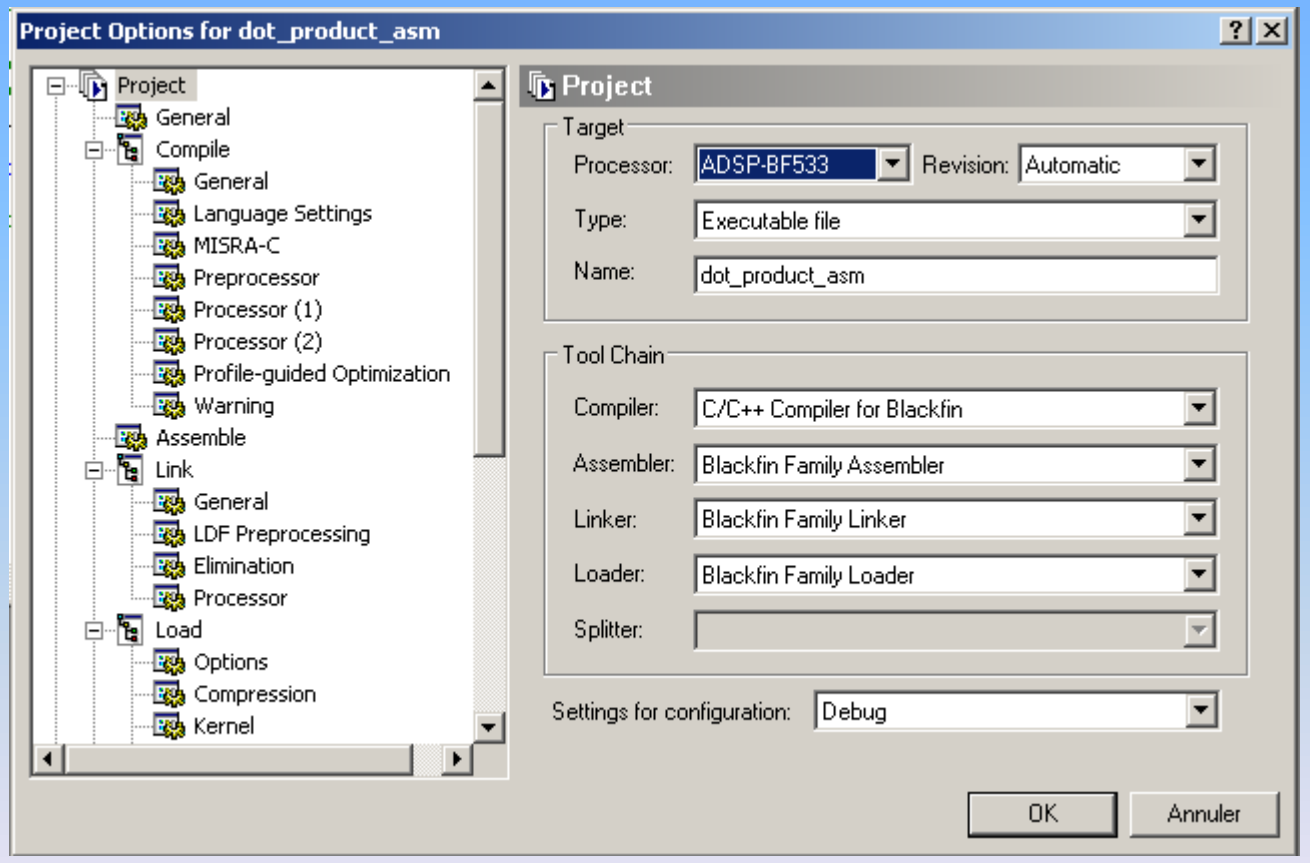
- The new project is created and is shown in the Project window of the IDE.



dot product asm

- From the **Project** menu click the **Project Options** command to display the **Project Options** dialog box

dot product asm



dot product asm

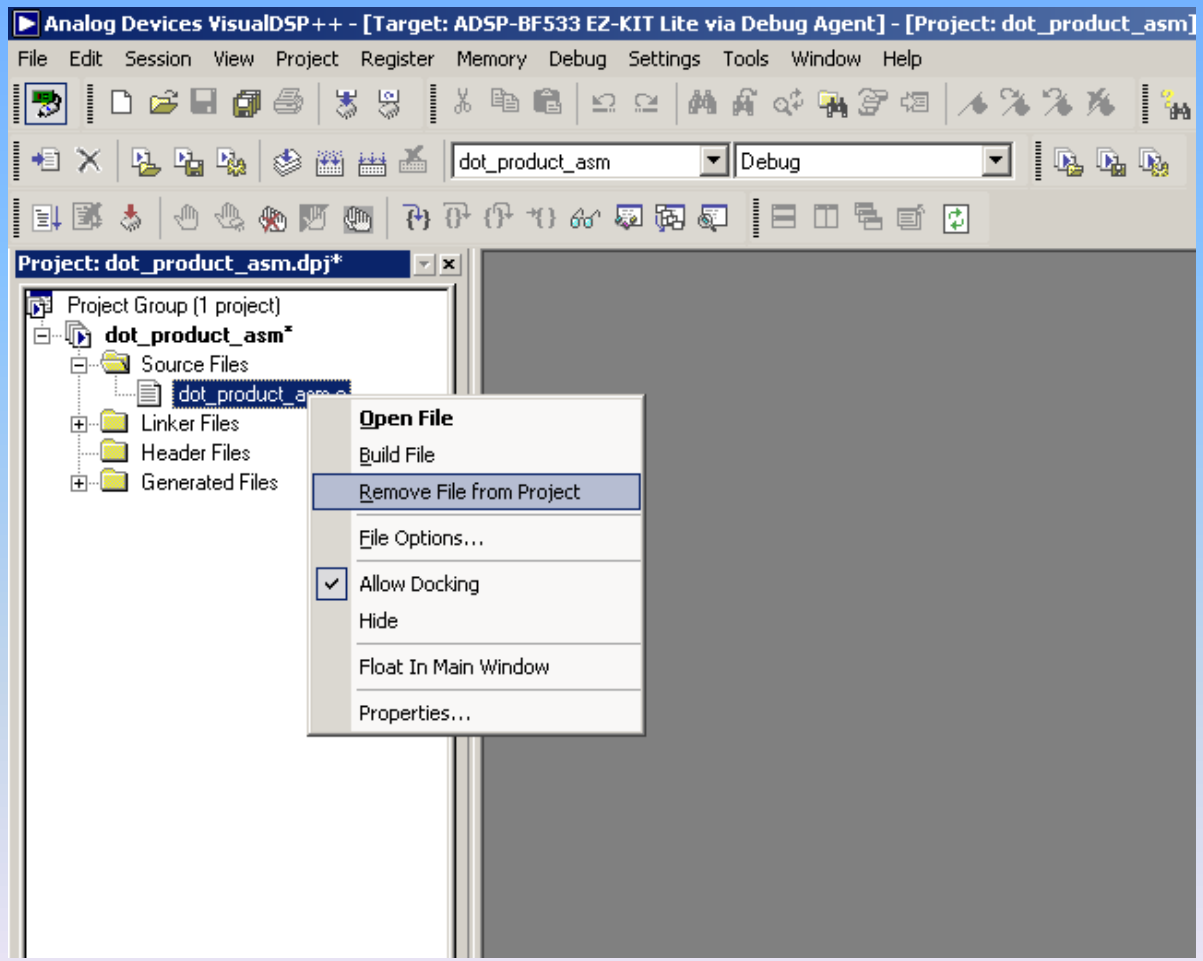
- On the **Project** page (see previous slide) verify that the values shown in the following table (or image of the previous slide) are entered.

Field	Value
Processor	ADSP-BF533
Revision	Automatic
Type	Executable file
Name	dot_product_asm
Settings for Configuration	Debug

dot product asm

- Specify these settings in the Code Generation group box:
  - Select the Enable optimization check box to enable optimization.
  - Select the Generate debug information check box, if it is not already selected, to enable debug information for the C source.
- Click OK to apply changes

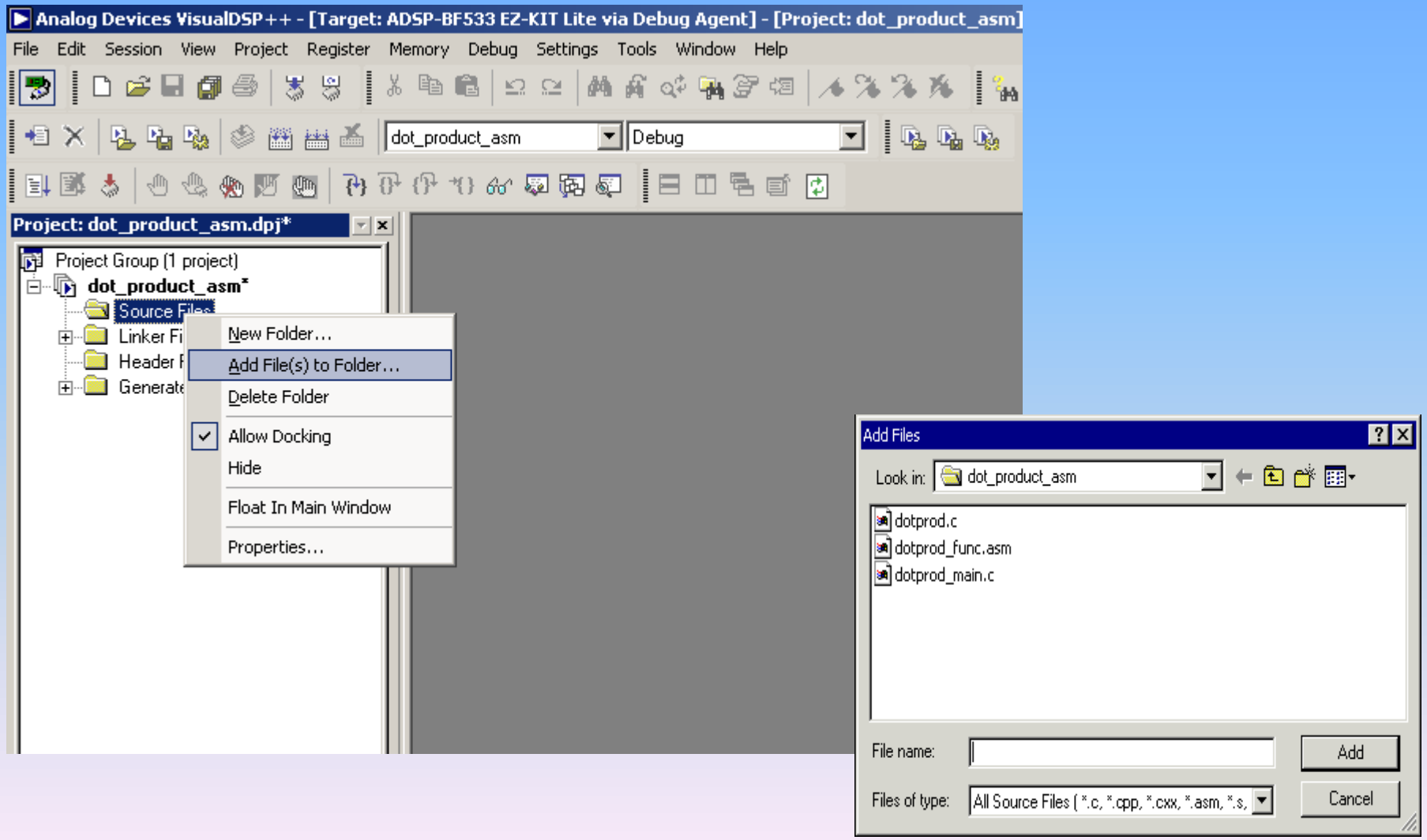
Remove the created *.c file



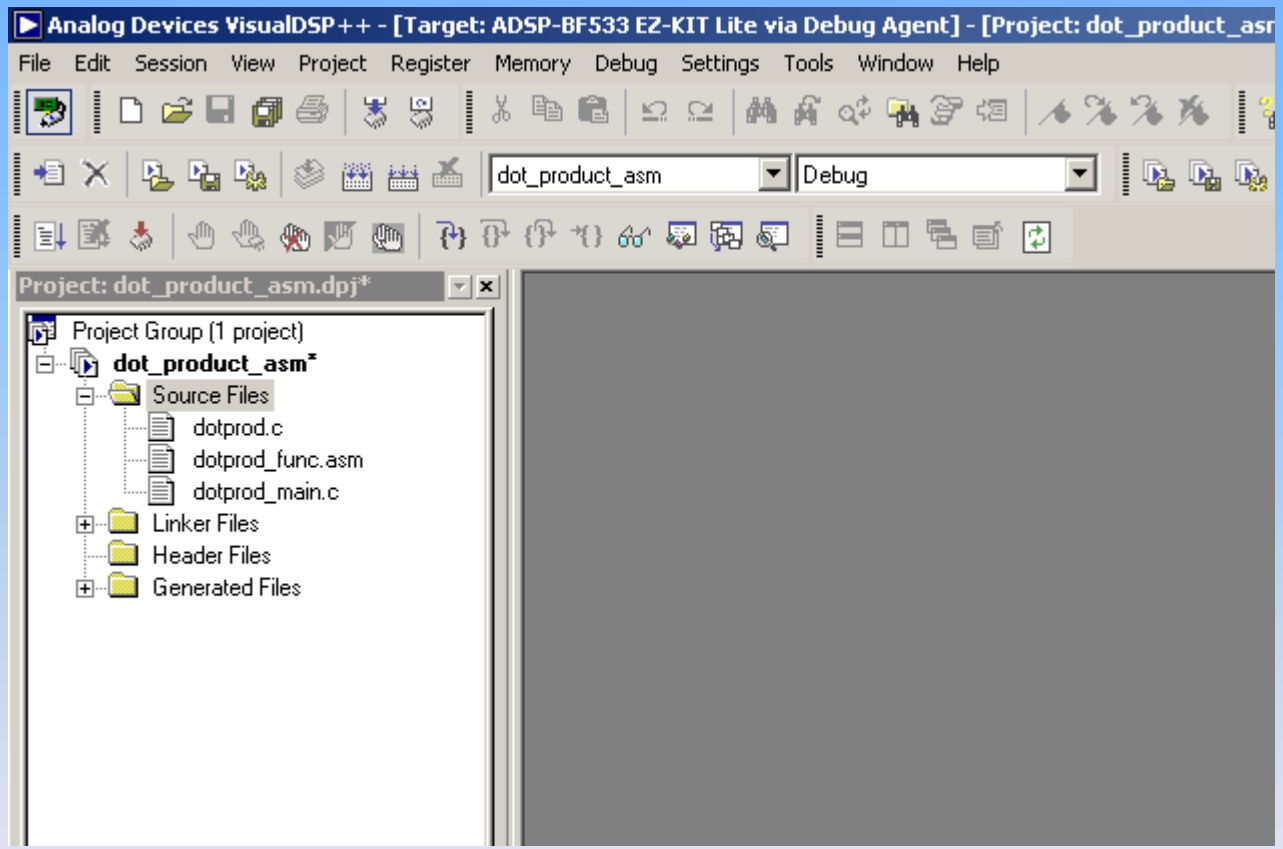
Adding Source Files to dot_product_asm

- To add the source file to the new project:
  - Click the Add File button, or
  - from the Project menu,
  - choose Add to Project, and then choose File(s).
- The Add Files dialog box appears.

Adding Source Files to dot_product_asm



You have now:


Analog Devices VisualDSP++ - [Target: ADSP-BF533 EZ-KIT Lite via Debug Agent] - [Project: dot_product_asr
File Edit Session View Project Register Memory Debug Settings Tools Window Help
dot_product_asm
Debug
Project: dot_product_asm.dpj*
Project Group (1 project)
dot_product_asm*
Source Files
dotprod.c
dotprod_func.asm
dotprod_main.c
Linker Files
Header Files
Generated Files

Modifying dotprod_main

- **`dotprod_main.c`** calls some C language routines (see next slide)

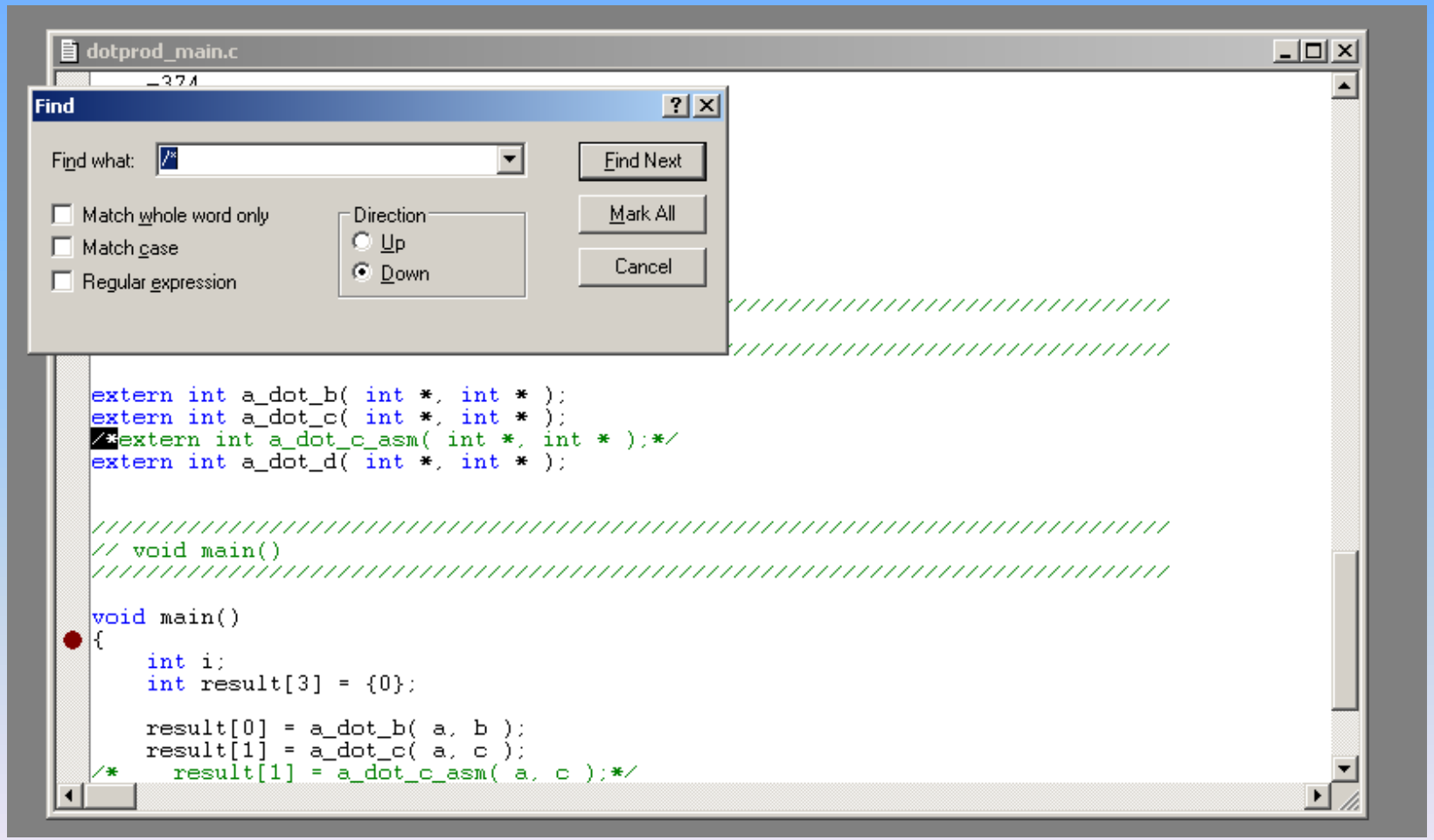
- we want to call now an assembler file

Modifying dotprod_main

dotprod_main.c

```
};

////////////////////////////////////////////////////////////////////////////////
// function prototypes
////////////////////////////////////////////////////////////////////////////////

extern int a_dot_b( int *, int * );
extern int a_dot_c( int *, int * );
/*extern int a_dot_c_asm( int *, int * );*/
extern int a_dot_d( int *, int * );


////////////////////////////////////////////////////////////////////////////////
// void main()
////////////////////////////////////////////////////////////////////////////////

void main()
{
    int i;
    int result[3] = {0};

    result[0] = a_dot_b( a, b );
    result[1] = a_dot_c( a, c );
/*    result[1] = a_dot_c_asm( a, c );*/
    result[2] = a_dot_d( a, d );

    for( i=0; i<3; i++ )
    {
        printf( "Dot product [%d] = %d\n", i, result[i] );
    }
}
```

Type Ctrl+F (find), then /*



Modify:

- uncomment the call to the assembly routine
  - `extern int a_dot_c_asm( int *, int * );`
- comment the call to the C routine:
  - `extern int a_dot_c( int *, int * );`
- save the file

Before

dotprod_main.c

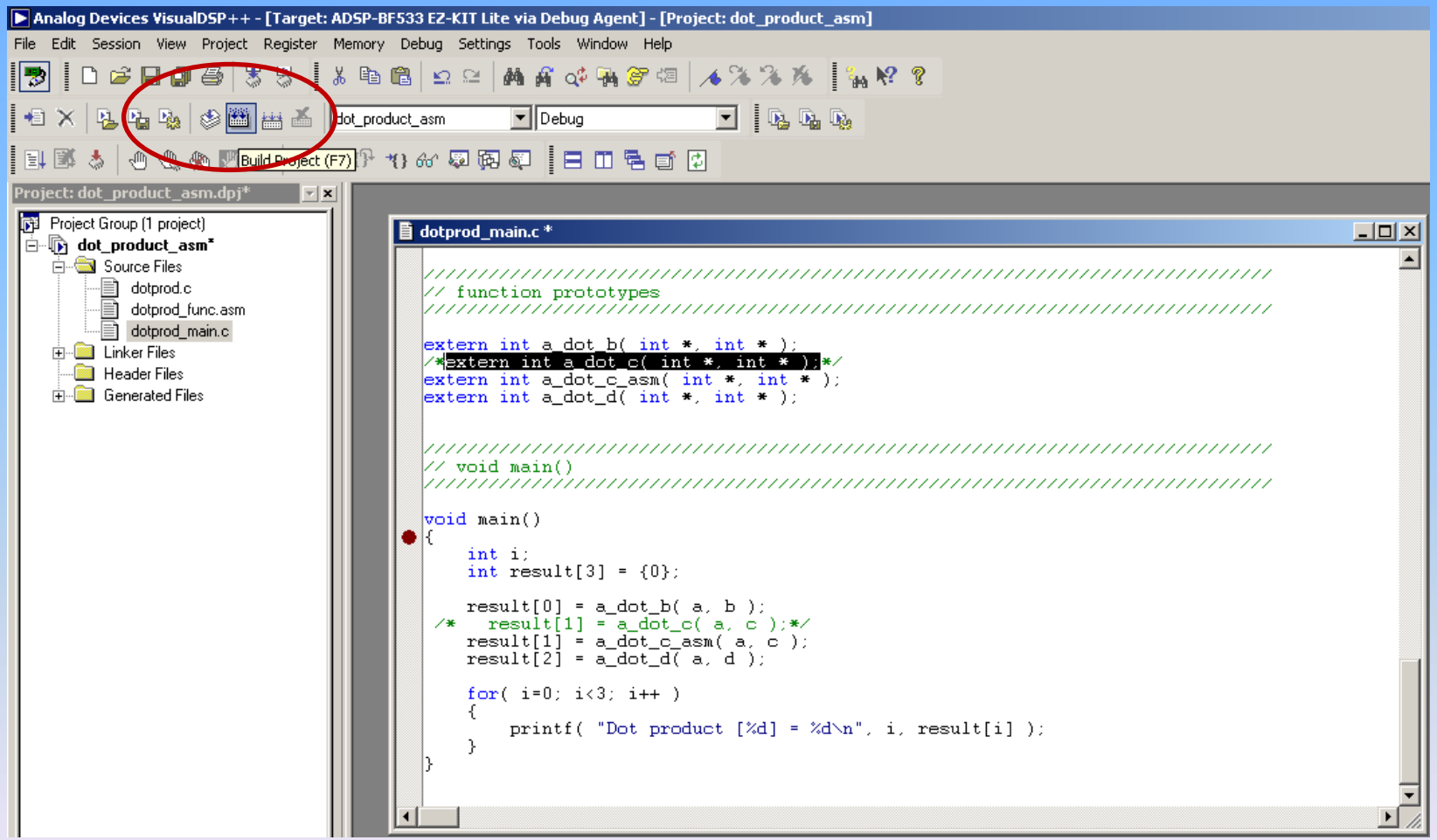
```
////////////////////////////////////////////////////////////////////////////////
// function prototypes
////////////////////////////////////////////////////////////////////////////////

extern int a_dot_b( int *, int * );
extern int a_dot_c( int *, int * );
/*extern int a_dot_c_asm( int *, int * );*/
extern int a_dot_d( int *, int * );


////////////////////////////////////////////////////////////////////////////////
// void main()
////////////////////////////////////////////////////////////////////////////////

void main()
{
    int i;
    int result[3] = {0};

    result[0] = a_dot_b( a, b );
    result[1] = a_dot_c( a, c );
/*    result[1] = a_dot_c_asm( a, c );*/
    result[2] = a_dot_d( a, d );

    for( i=0; i<3; i++ )
    {
        printf( "Dot product [%d] = %d\n", i, result[i] );
    }
}
```

After

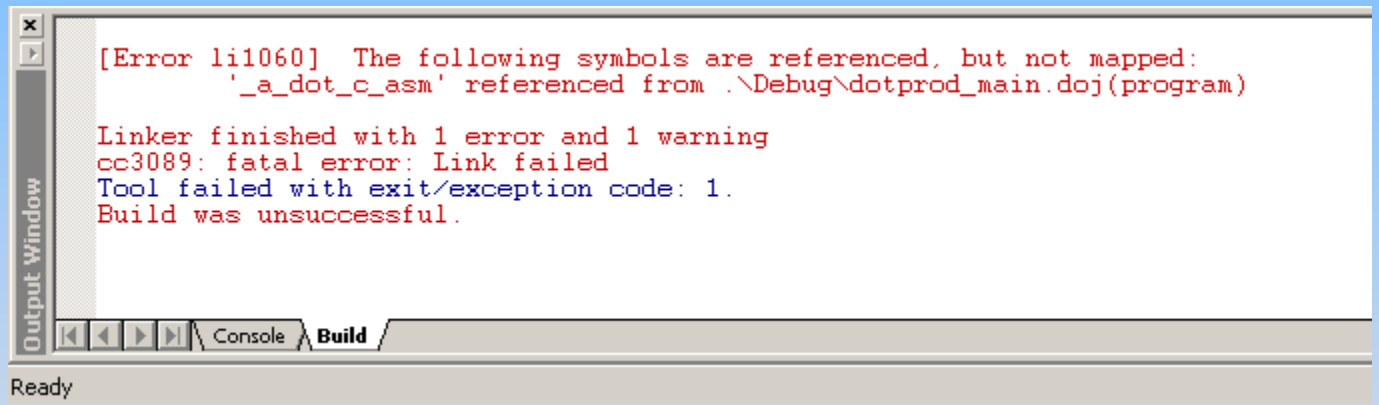
dotprod_main.c *

```
////////////////////////////////////////////////////////////////////////////////
// function prototypes
////////////////////////////////////////////////////////////////////////////////

extern int a_dot_b( int *, int * );
/*extern int a_dot_c( int *, int * );*/
extern int a_dot_c_asm( int *, int * );
extern int a_dot_d( int *, int * );


////////////////////////////////////////////////////////////////////////////////
// void main()
////////////////////////////////////////////////////////////////////////////////

void main()
{
    int i;
    int result[3] = {0};

    result[0] = a_dot_b( a, b );
/*  result[1] = a_dot_c( a, c );*/
    result[1] = a_dot_c_asm( a, c );
    result[2] = a_dot_d( a, d );

    for( i=0; i<3; i++ )
    {
        printf( "Dot product [%d] = %d\n", i, result[i] );
    }
}
```

Build the project



Failure…

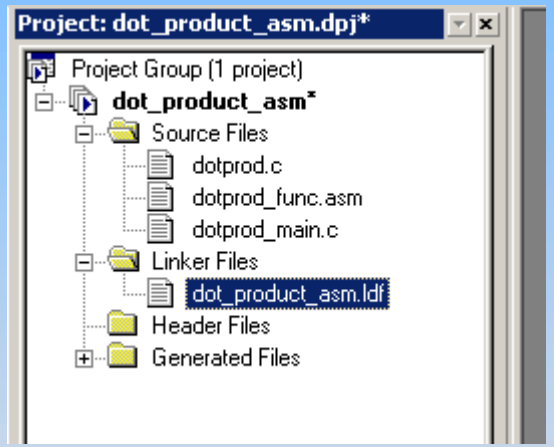

Output Window
[Error li1060] The following symbols are referenced, but not mapped:
'_a_dot_c_asm' referenced from .\Debug\dotprod_main.doj(program)
Linker finished with 1 error and 1 warning
cc3089: fatal error: Link failed
Tool failed with exit/exception code: 1.
Build was unsuccessful.
Console
Build
Ready

The linker

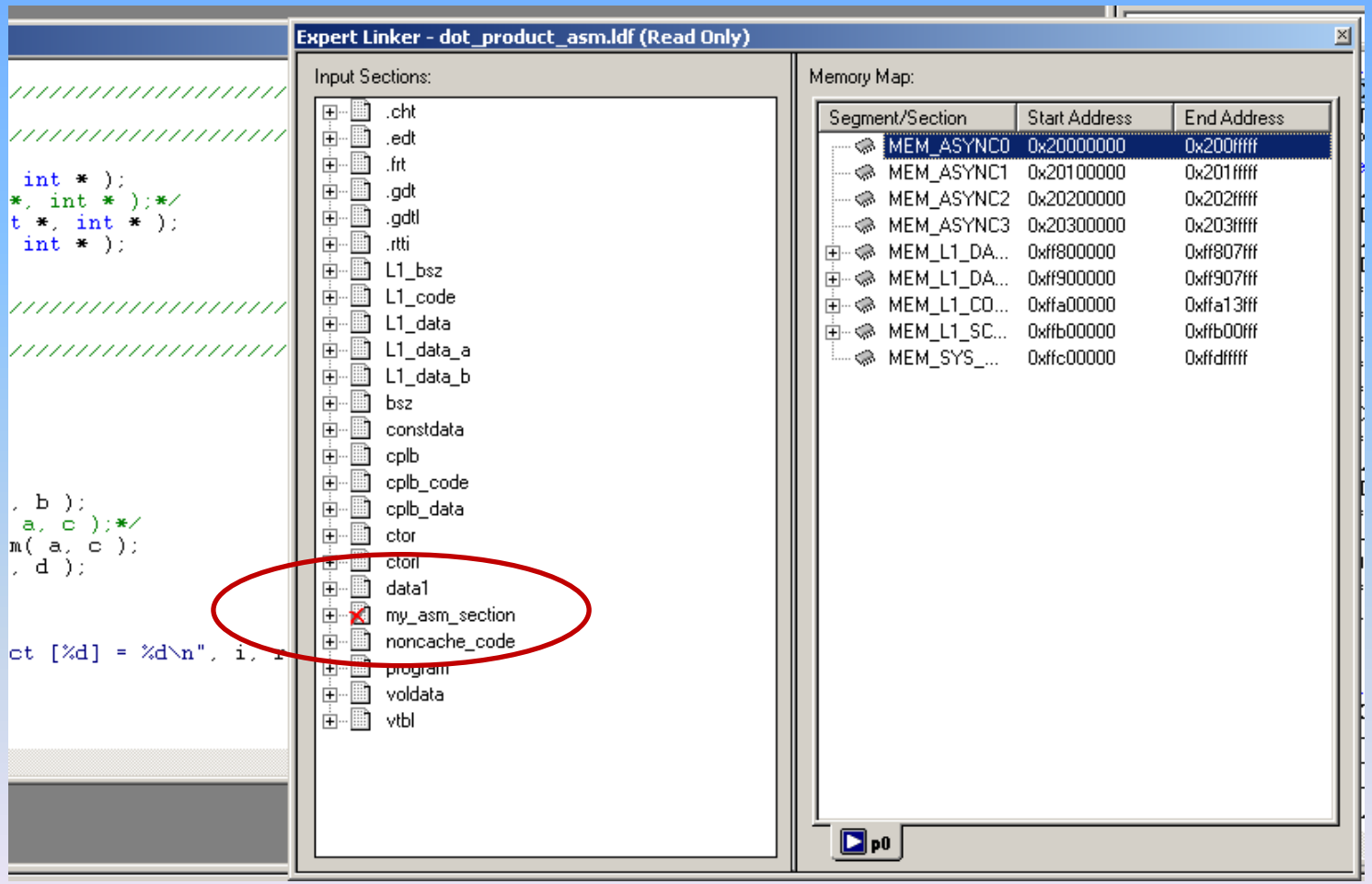
- Output Window: Linker Error

- To correct this error the do the following:

Double click the linker file:



The expert linker appears



The expert linker

Resize the window to expand the view and change the view mode.

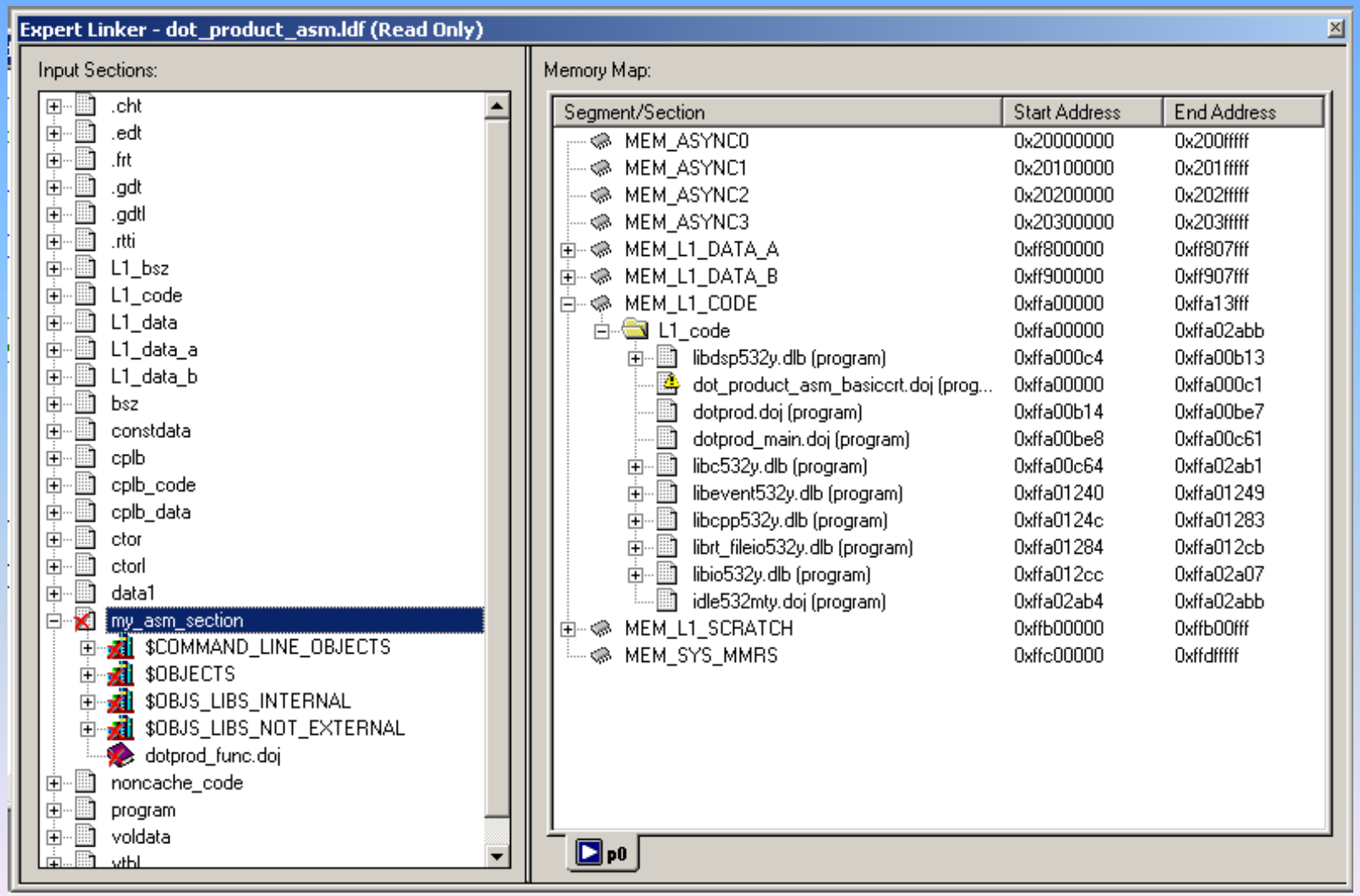
To display the tree view shown in next slide, right-click in the right pane, choose **View Mode**, and then choose **Memory Map Tree**.

The expert linker

The left pane (**Input Sections**) contains a list of the input sections that are in your project or are mapped in the .LDF file. A red X is over the icon in front of the section named "my_asm_section" because Expert Linker has determined that the section is not mapped by the .LDF file.

The right pane (**Memory Map**) contains a representation of the memory segments that Expert Linker defined when it created the .LDF file.

The expert linker



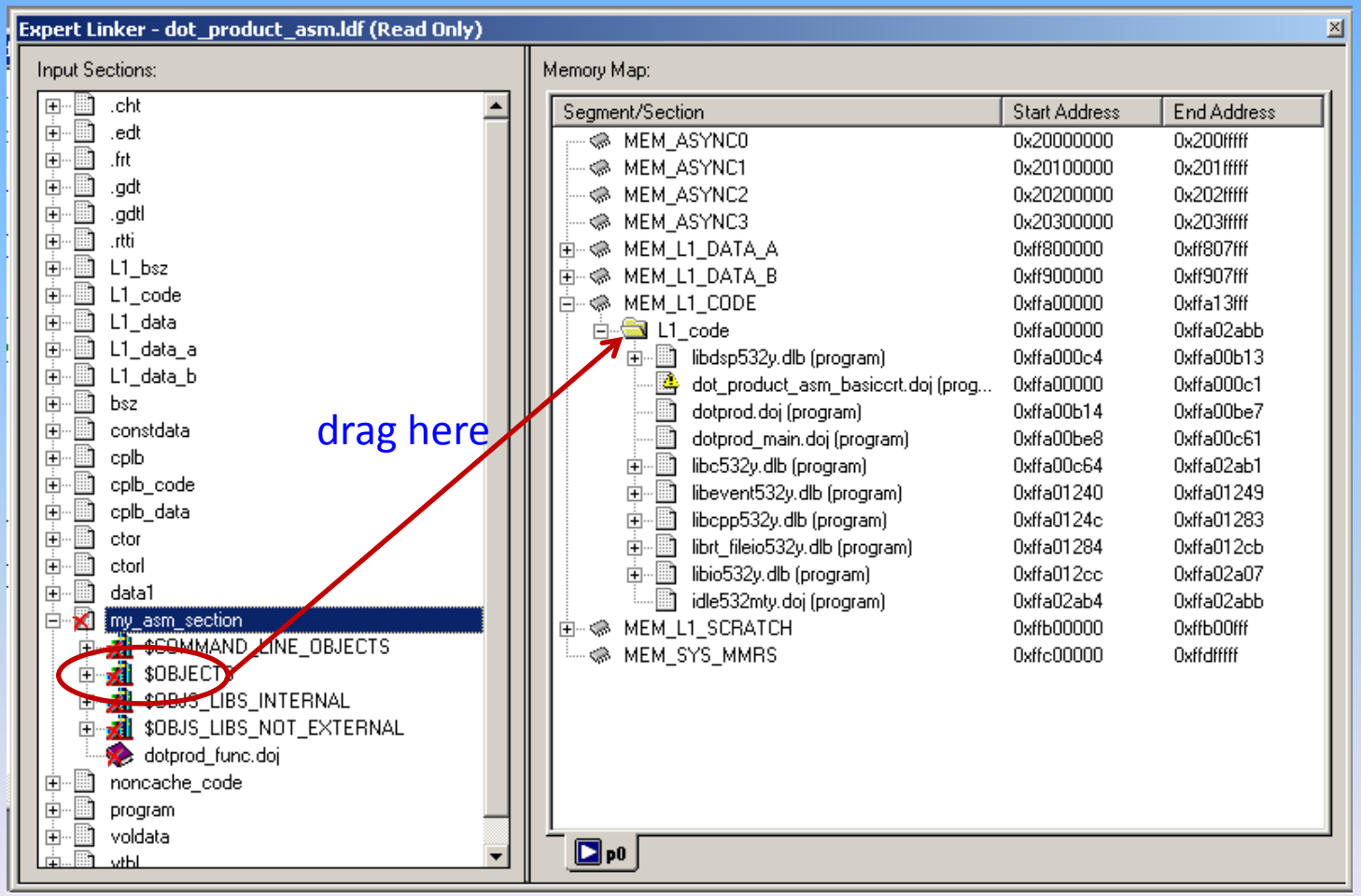
Use the Expert Linker to map

- In the Input Sections pane, open my_asm_section by clicking on the plus sign in front of it. The input section expands to show that the linker macros:
  - $COMMAND_LINE_OBJECTS and
  - $OBJECTS and
  - the object file dotprod_func.doj have a section that has not been mapped.

Use the Expert Linker

- In the Memory Map pane, expand MEM_L1_CODE and drag the icon in front of $OBJECTS onto the program_ram output section under MEM_L1_CODE.

Use the Expert Linker



Rebuild and Run

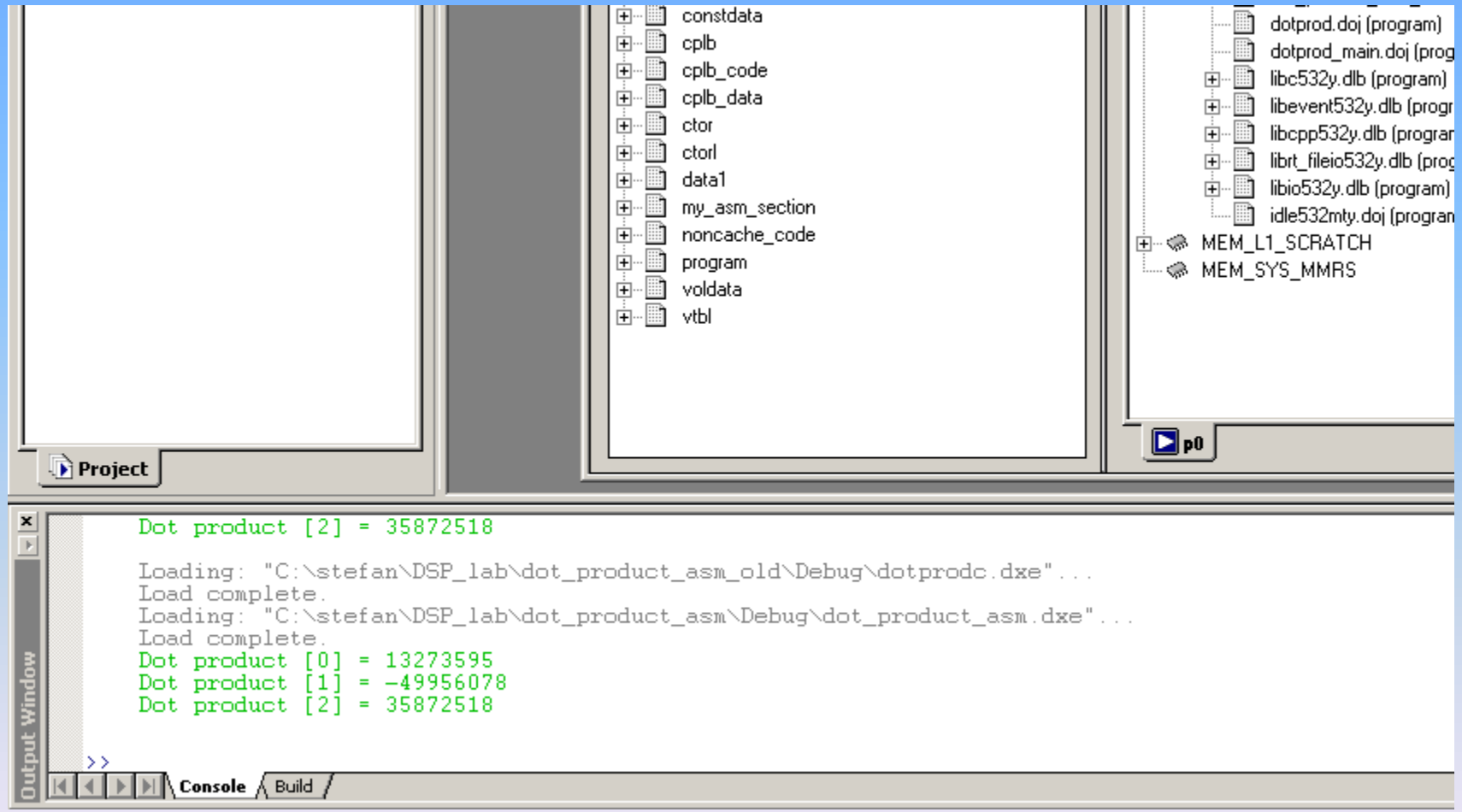
- Rebuild and Run dot_prod_asm

- At the end of the build, the Output window should display this message in the Build view:
  - “Build completed successfully”

Warning

- sometimes the my_asm_section does not appear
- make "Clean Project" and build again
- repeat this until it appears

Rebuild and Run



dot product asm

- when you're done, close the project

FIR Filter example

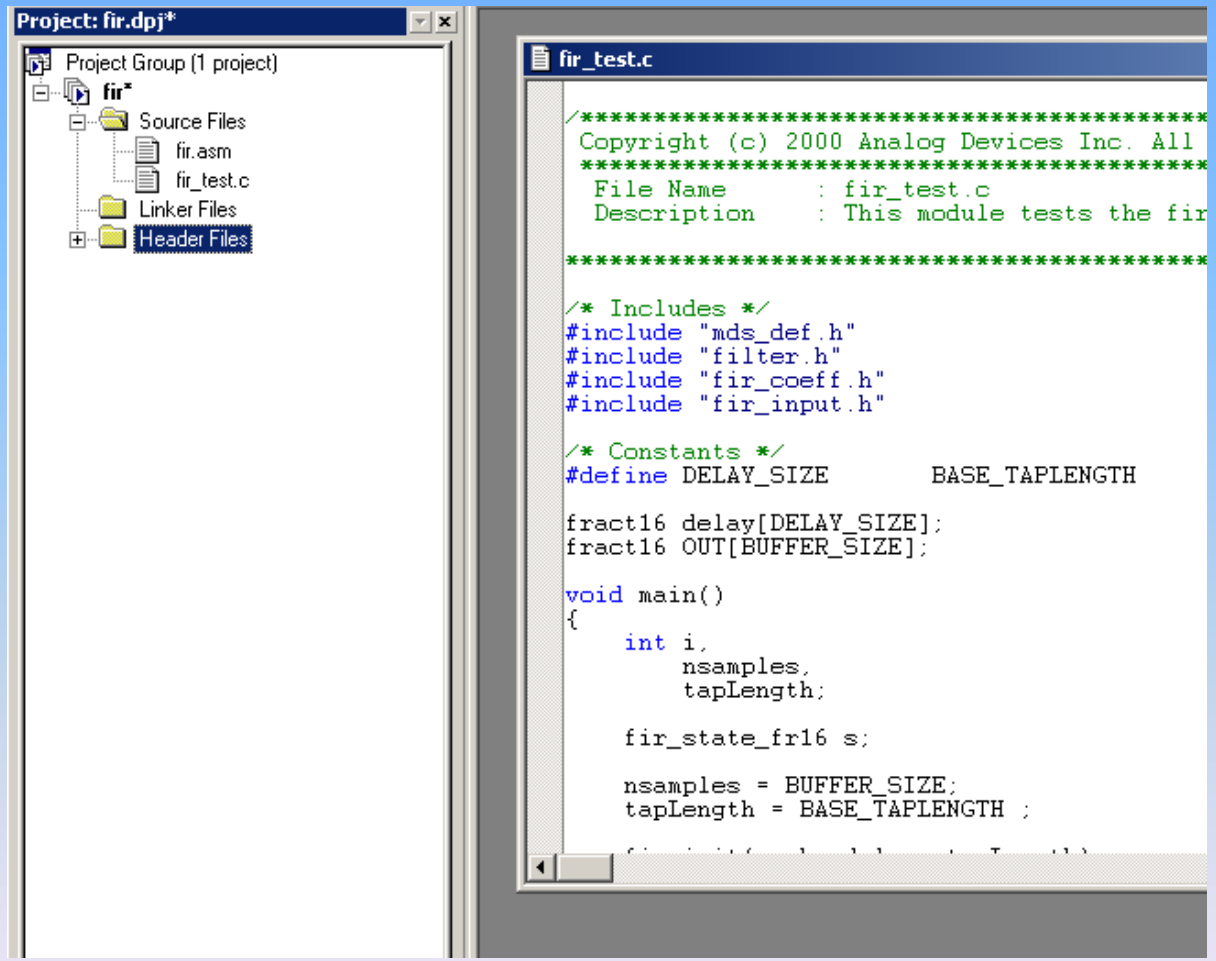
Opening a FIR project and displaying the input and output data

FIR Filter example

- Unzip and copy the files from fir.zip into your working directory (e.g. D:\GrXX\YourName\FIR)

- Open the FIR project

You should see:

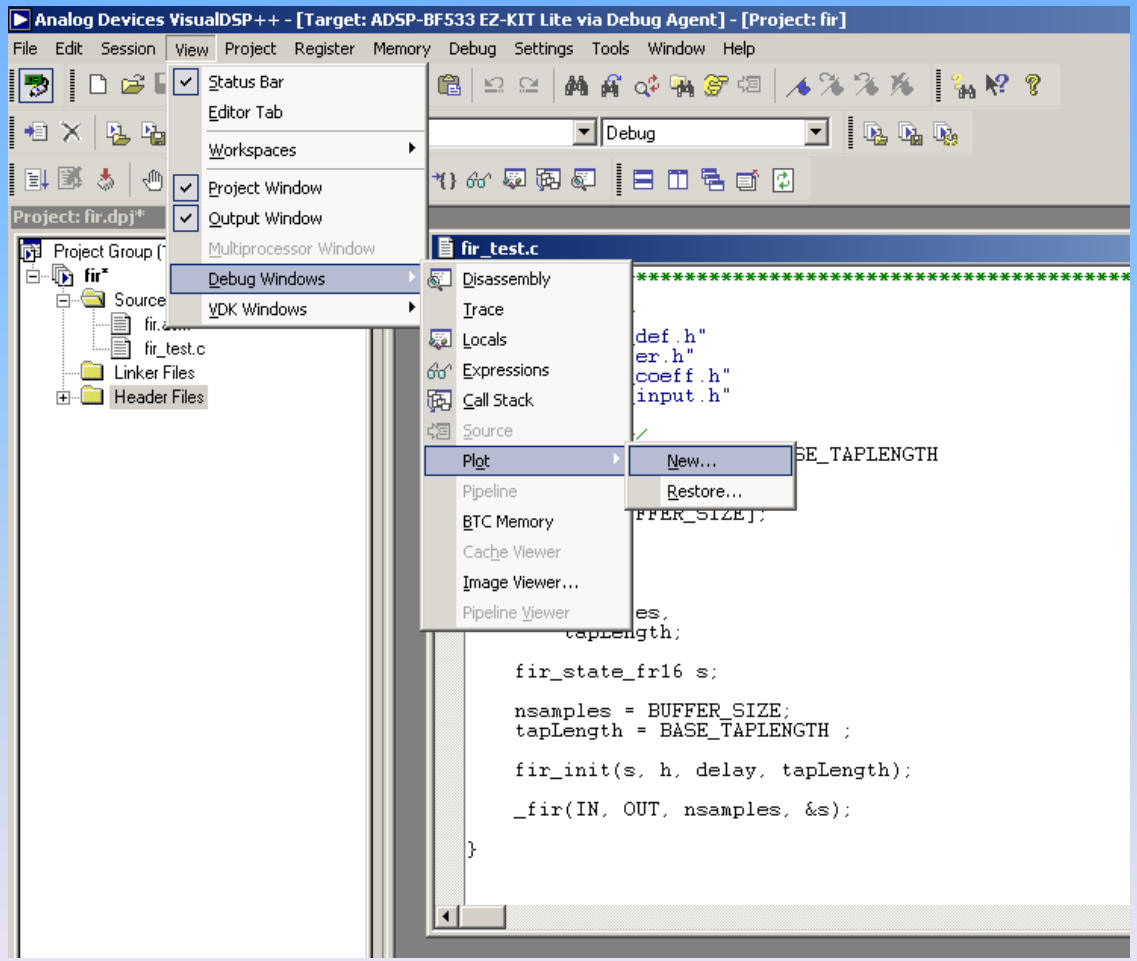


Starting to plot

- From the **View** menu, choose **Debug Windows** and **Plot**. Then choose **New** to open the **Plot Configuration** dialog box, shown in the next slide Figure.

  Here you add the data sets that you want to view in a plot window.

Add a new graphic:



A new window appears:

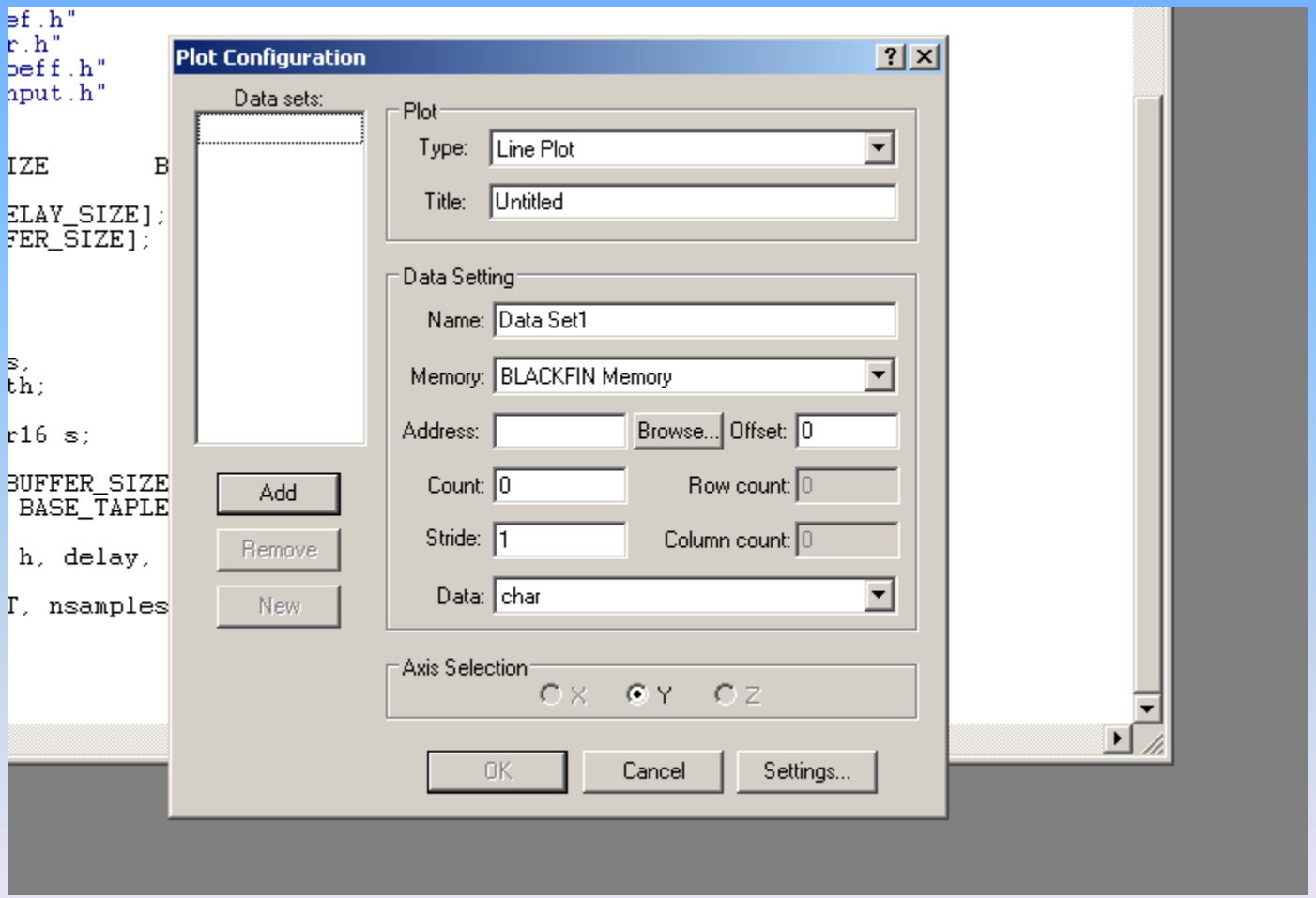


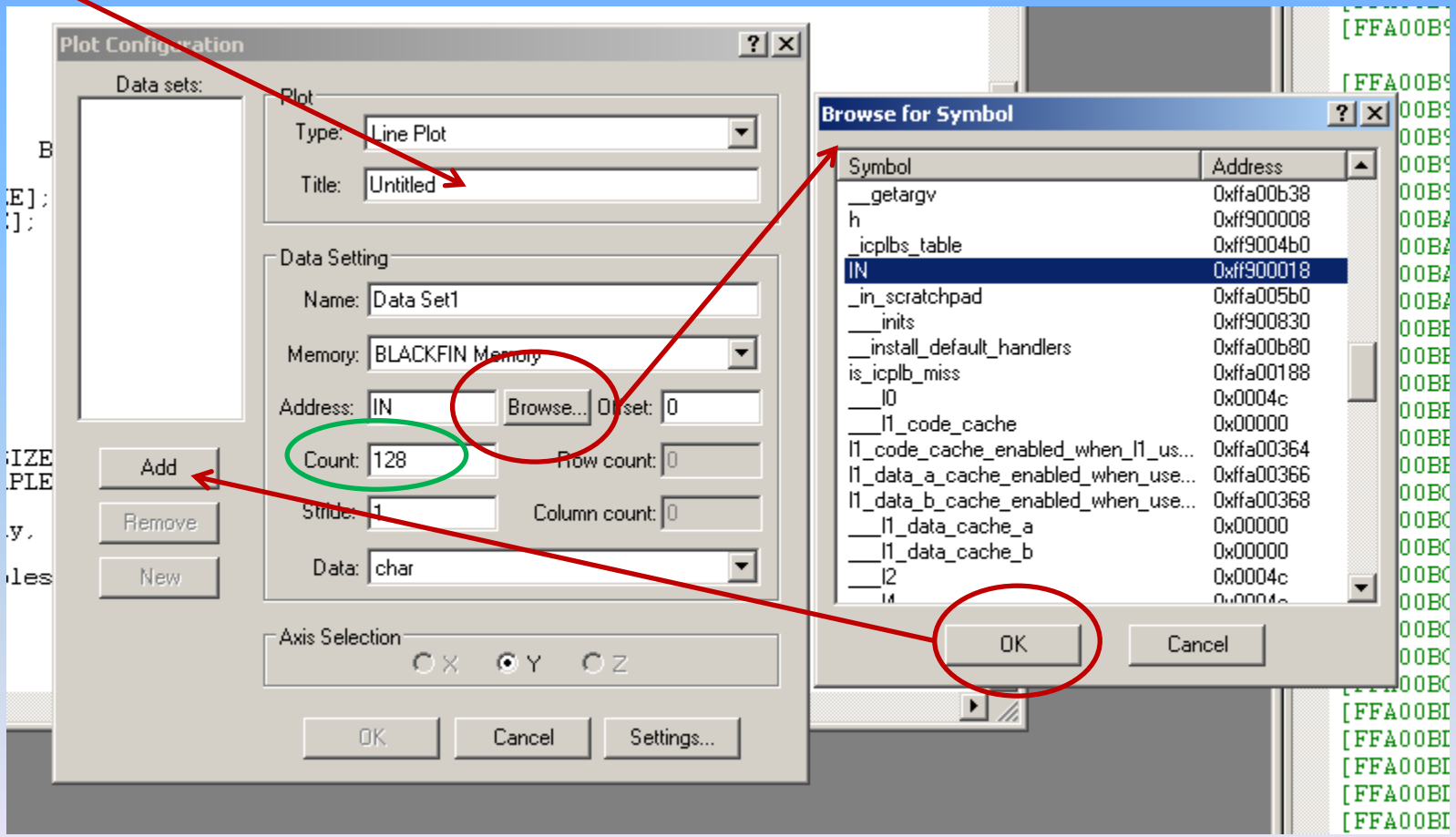
Table Specifying 2 Data sets: Input & Output

Box	Input Data Set	Output Data Set	Description
Name	Input	Output	Data set
Memory	BLACKFIN Memory	BLACKFIN Memory	Data memory
Address	IN	OUT	The address of this data set is that of the Input or Output array. Click Browse to select the value from the list of loaded symbols
Count	128	128	The array is 260 elements long, but you are plotting the first 128 elements
Stride	1	1	The data is contiguous in memory
Data	short	short	Input & Output are arrays of **int** values

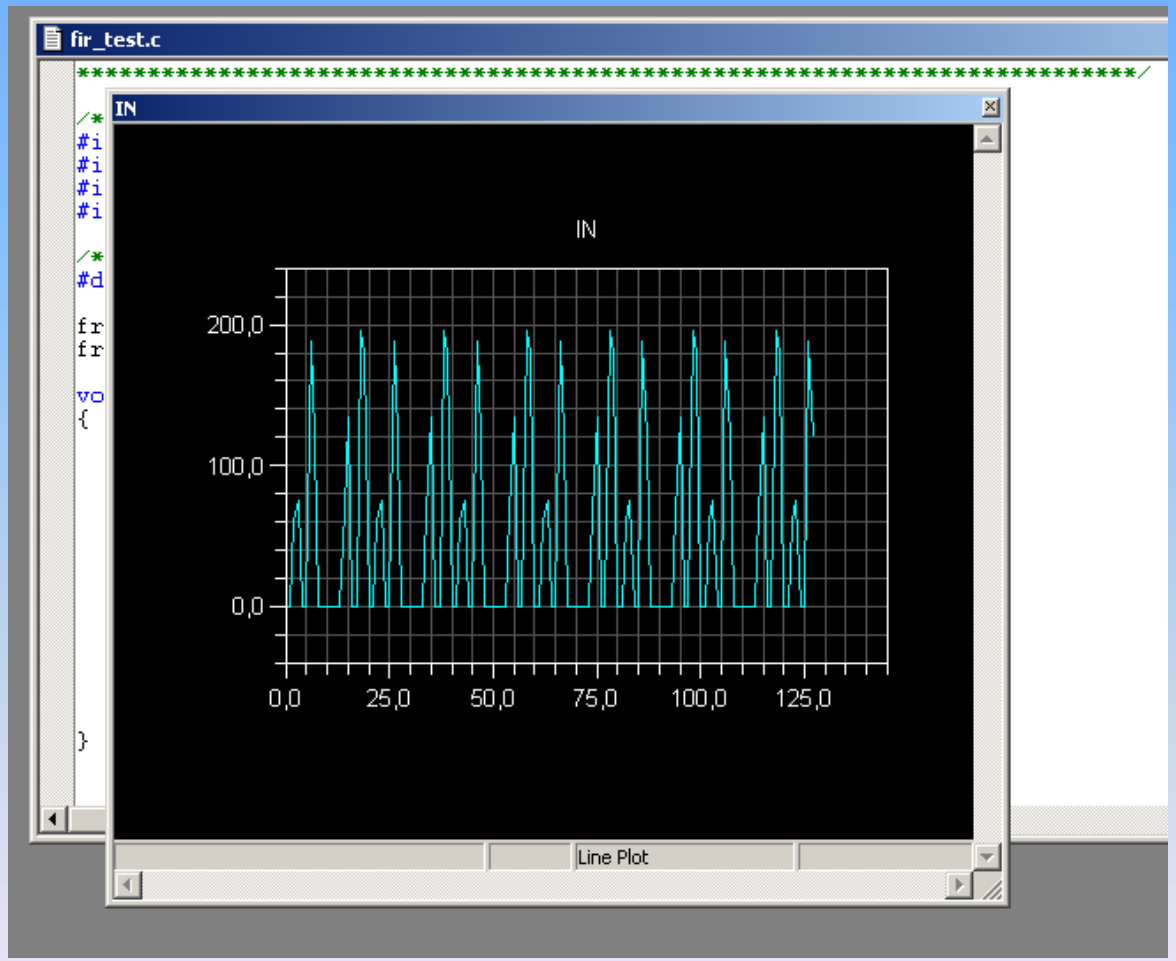
- After entering each data set, click **Add** to add the data set to the **Data sets** list on the left of the dialog box.
- The **Plot Configuration** dialog box should now look like the one in the Figure presented in the next slide.

Add a first data set (IN)

type the name



You should see:

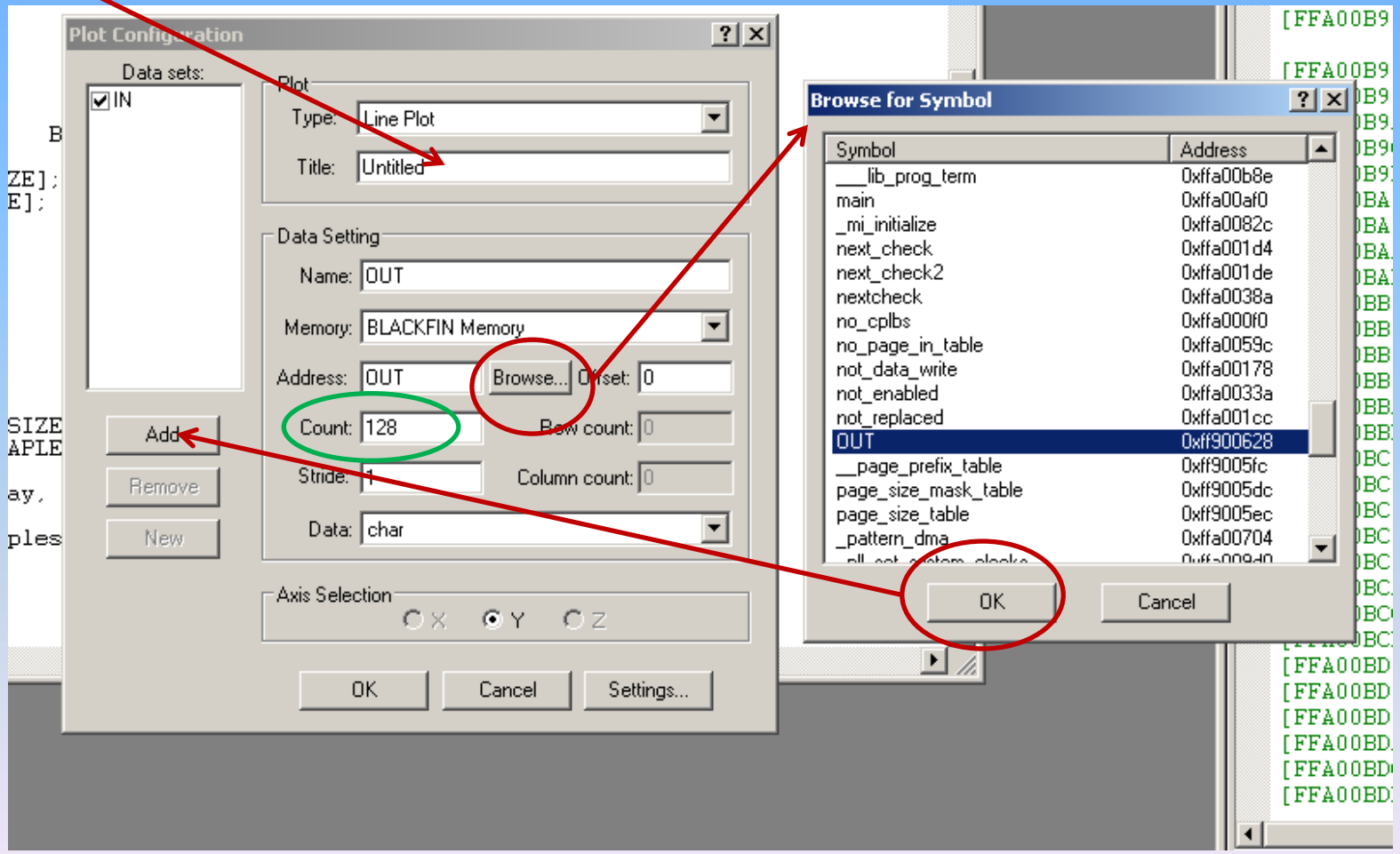


Add a second data set (OUT)

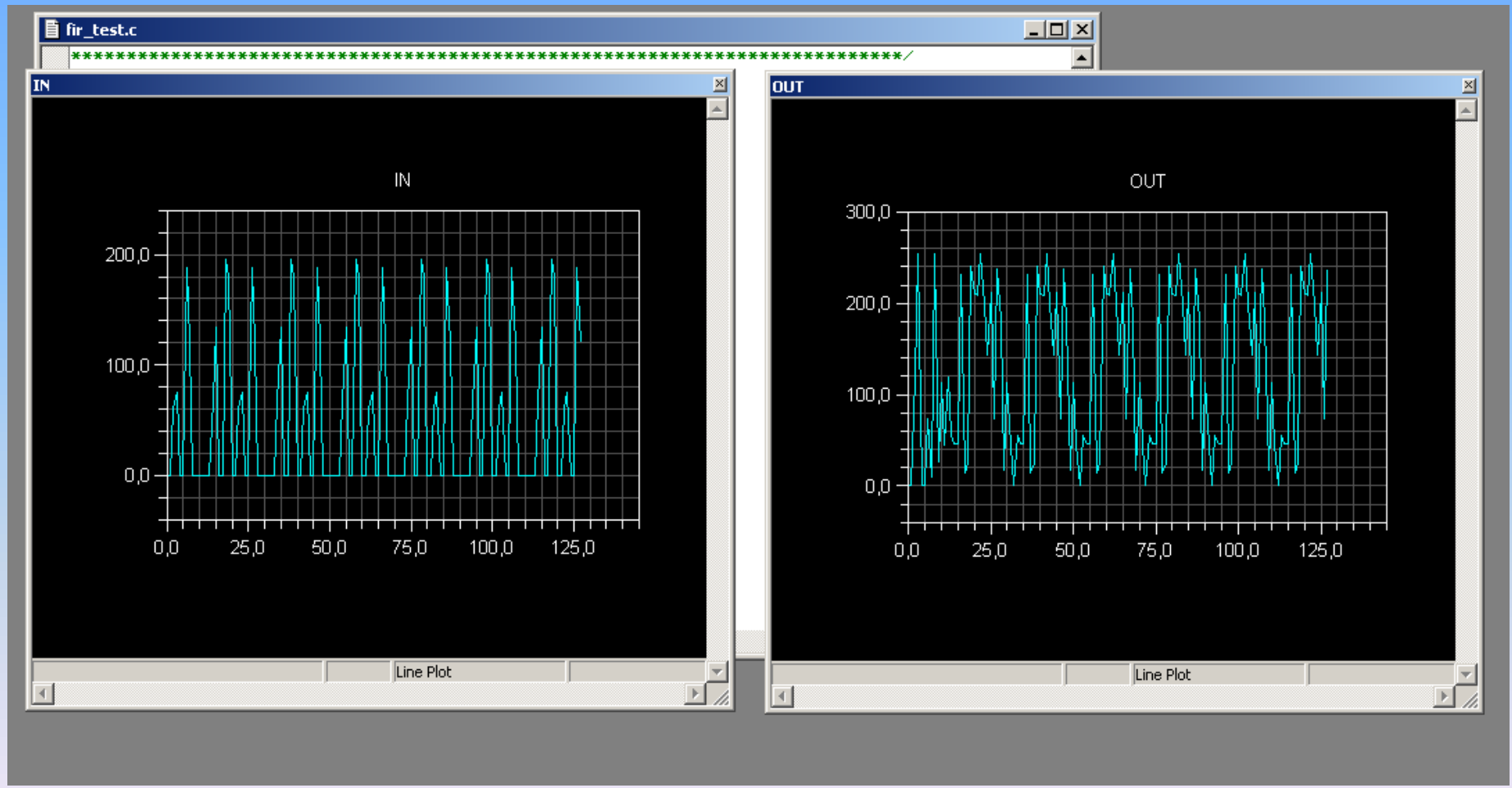
- Again: from the **View** menu, choose **Debug Windows** and **Plot**. Then choose **New** to open the **Plot Configuration** dialog box,

Add a second data set (OUT)

type the name

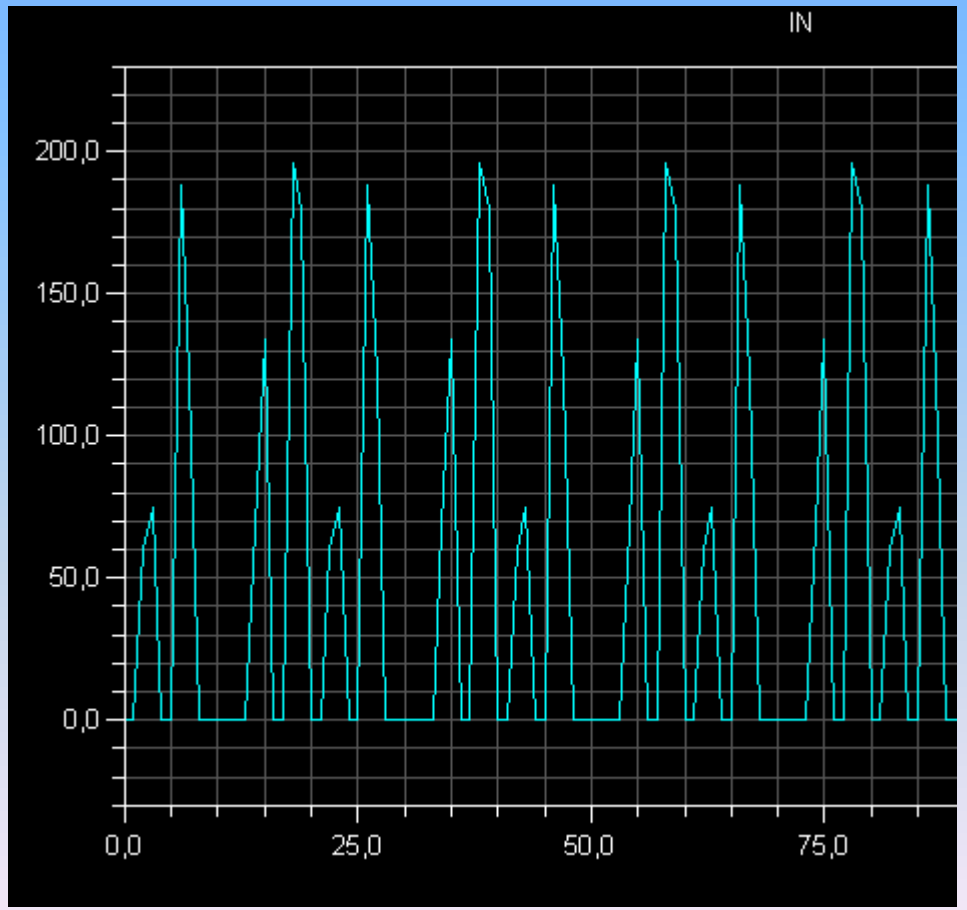


You should see:



Wait a minute…

- this is distorted data!
- something is wrong!

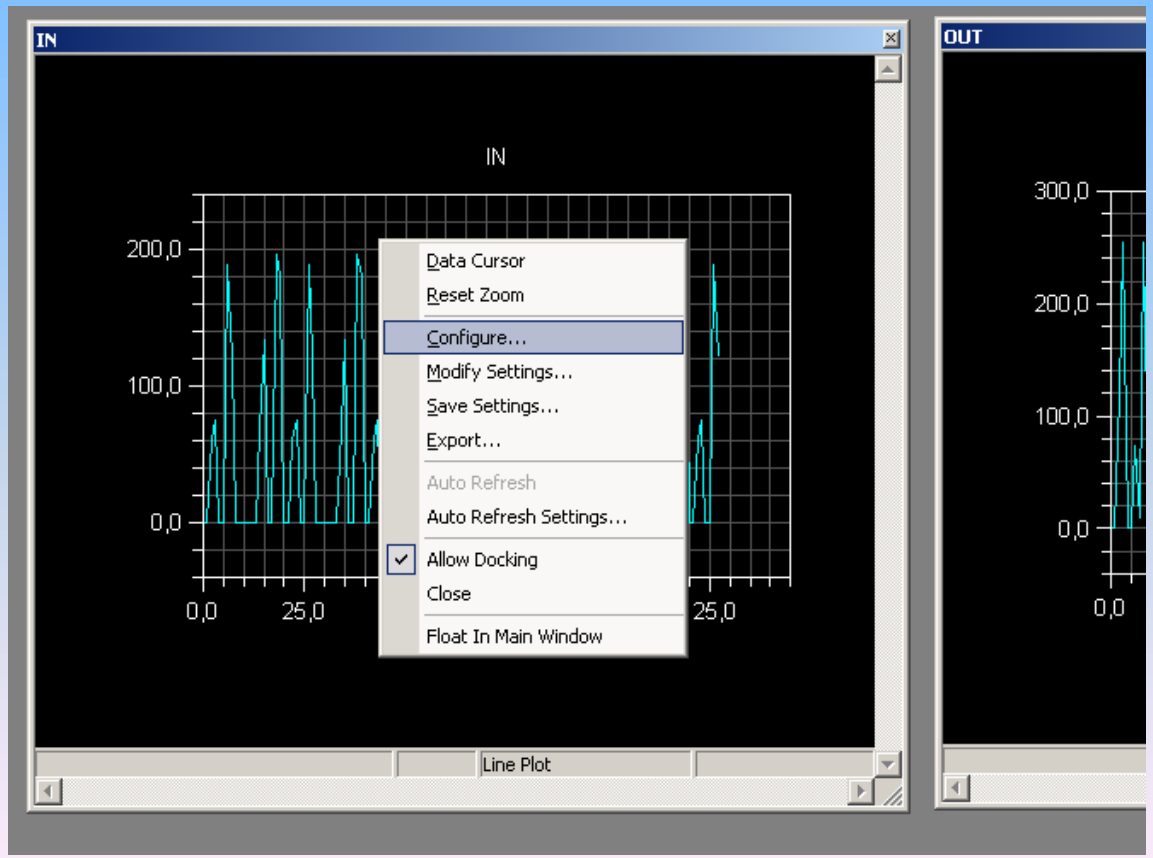


Any guesses?

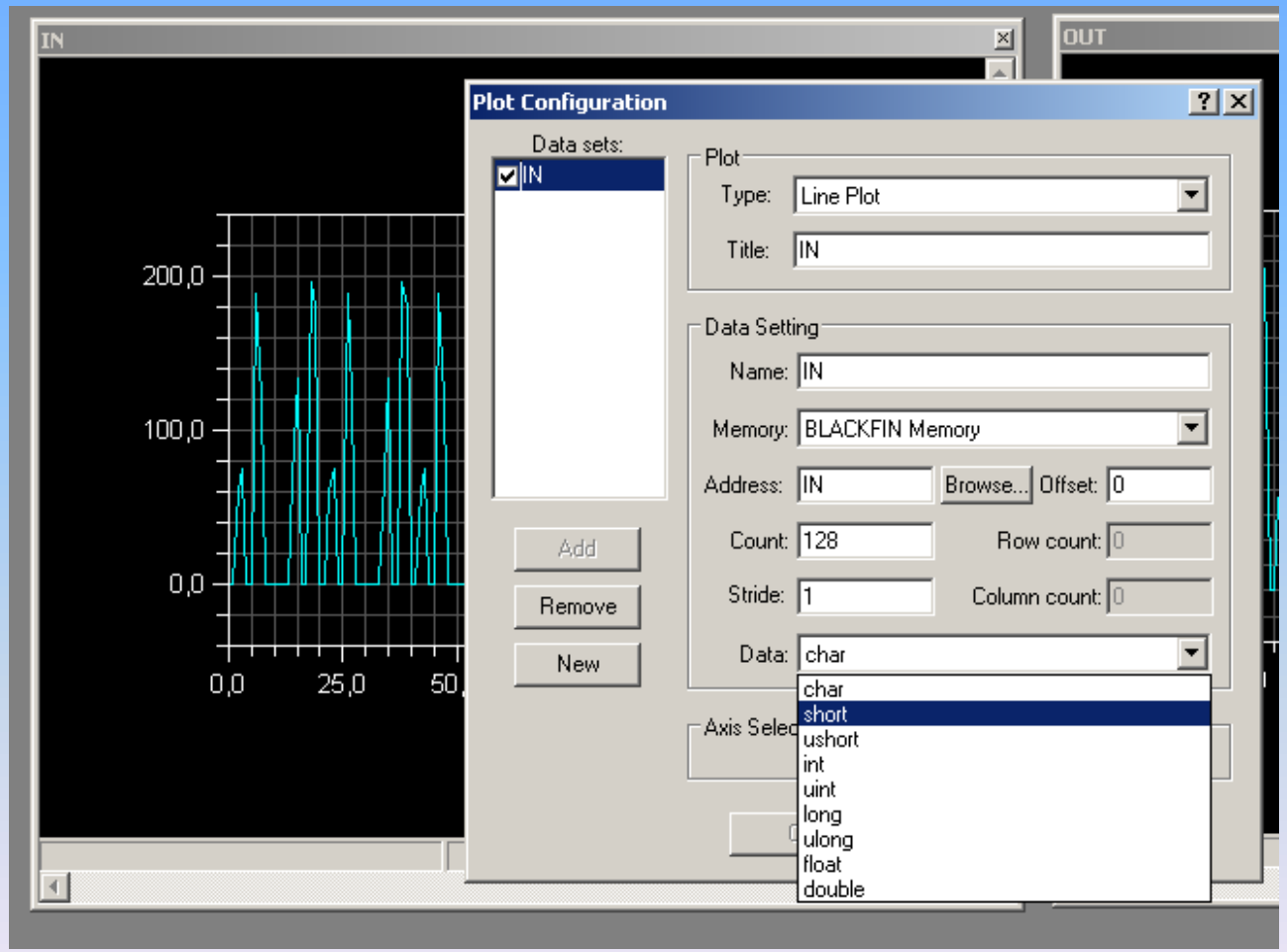
Why data is always positive and below 256?

Answer: the data type!

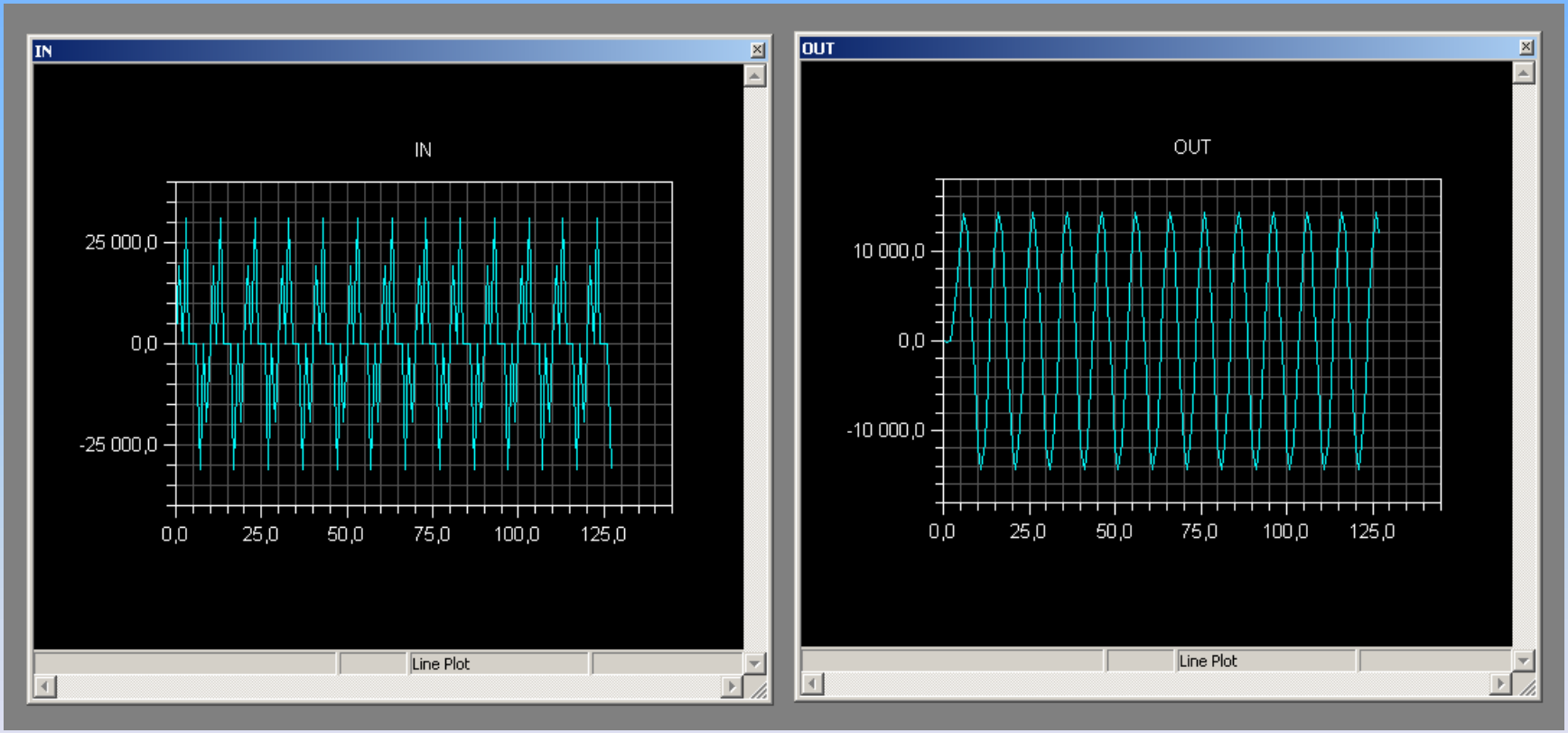
- you selected "char"
- you should have selected "short"



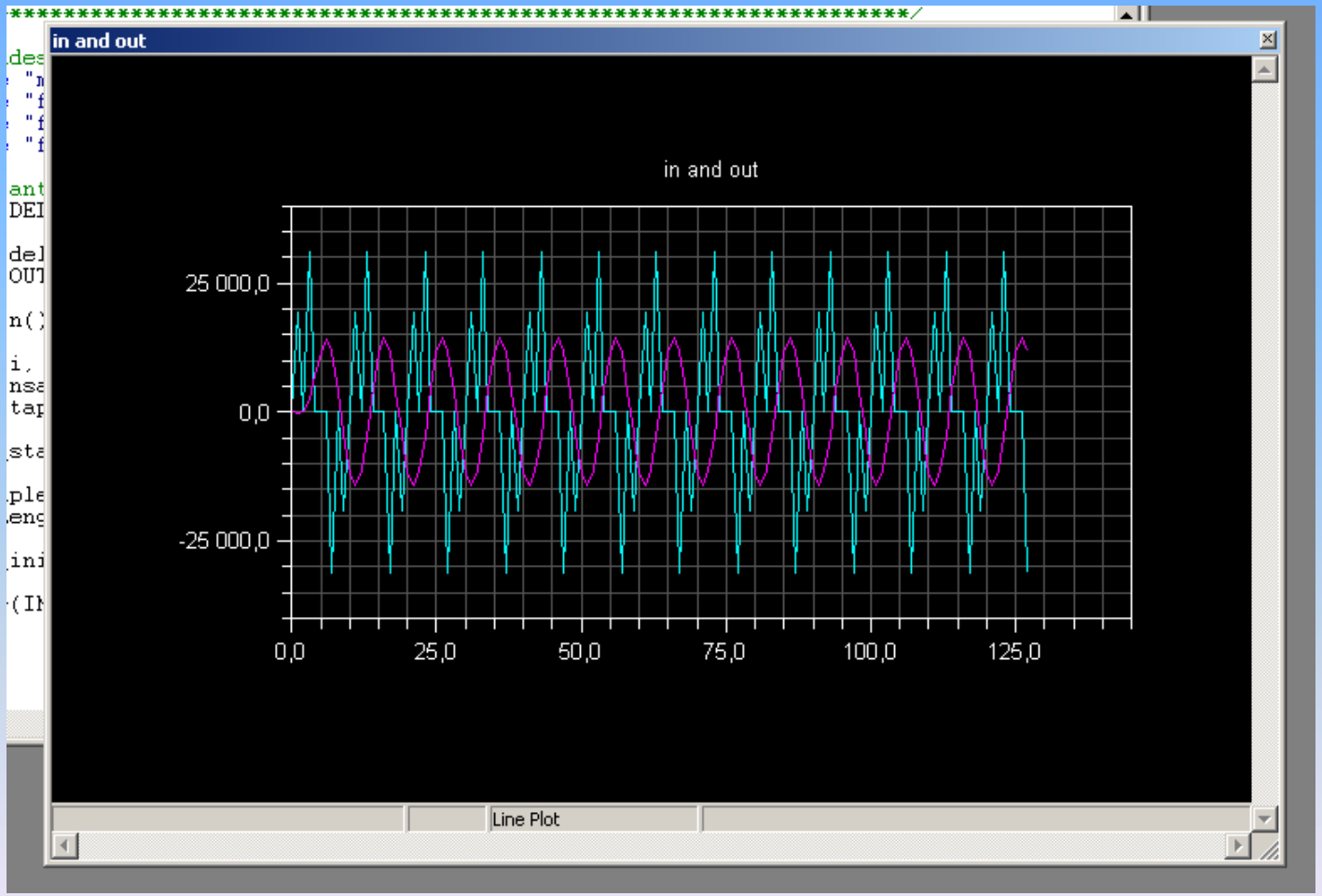
Modify the data type (for both graphics)



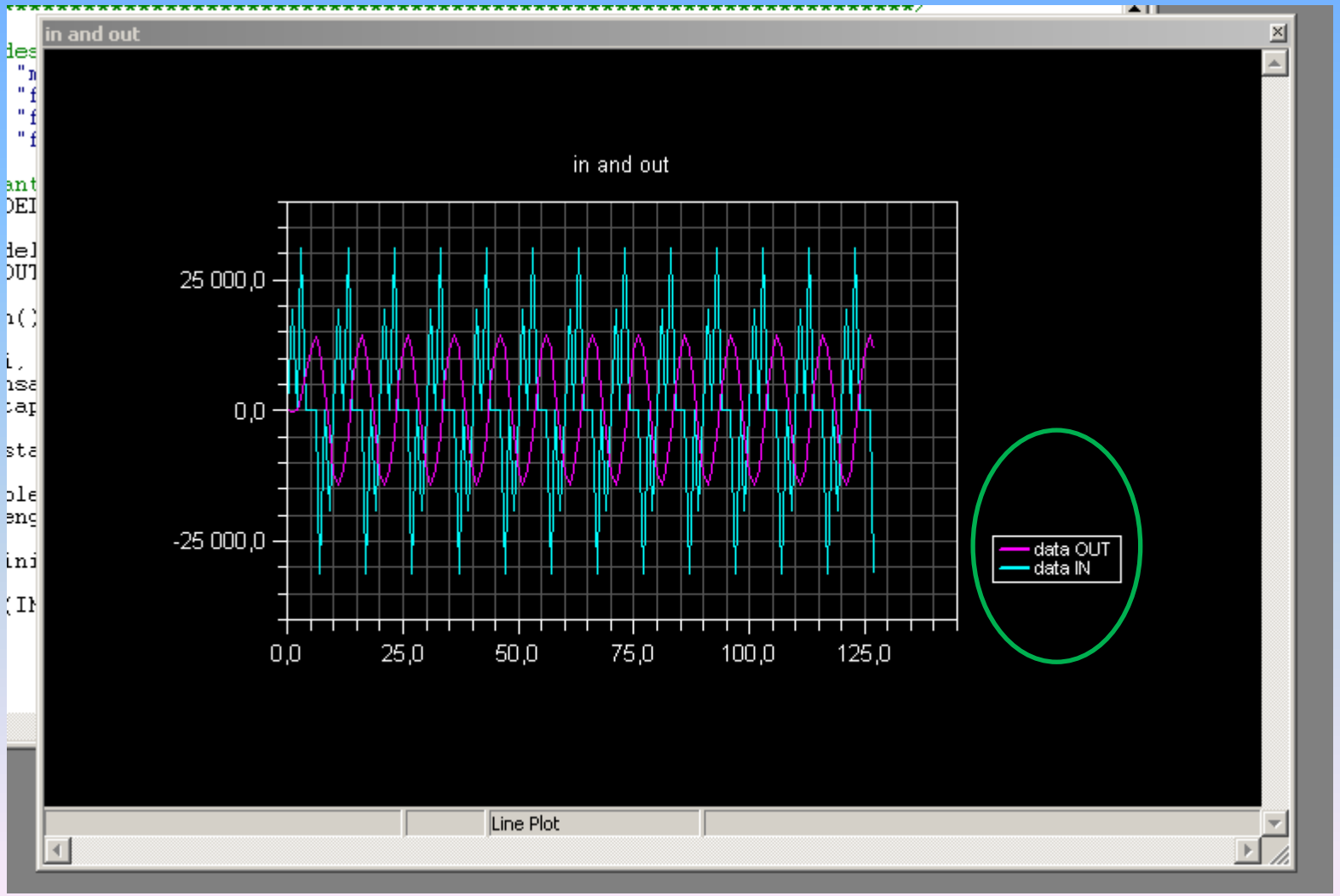
You should see now:



Now add both curves on the same graph:



Play around with the settings

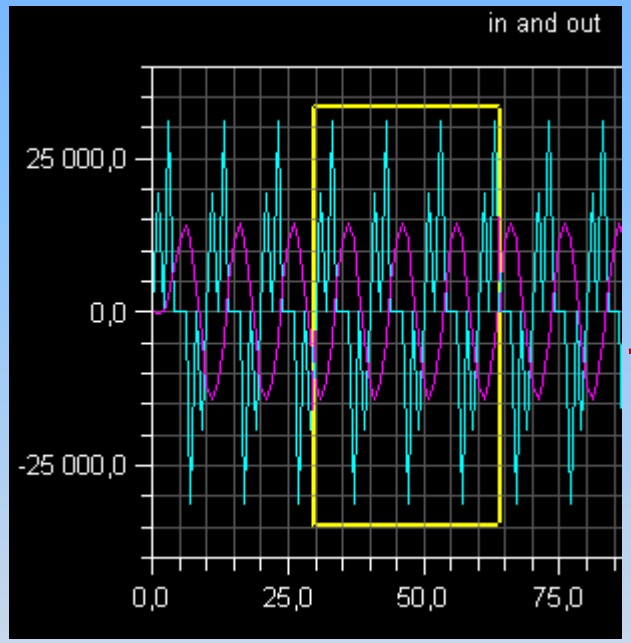


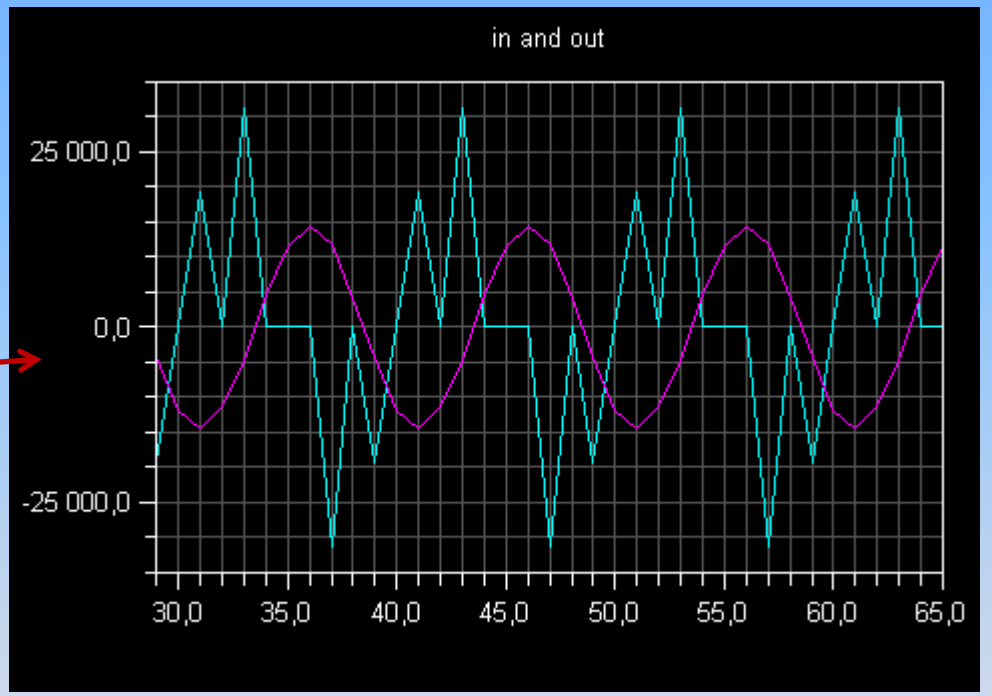
Hit F5 (run)

- you should see the real-time execution of your program

- Additional Features of the Plotting interface allows you to select the region and to zoom in.

Zooming:


in and out
25 000,0
0,0
-25 000,0
0,0
25,0
50,0
75,0


in and out
25 000,0
0,0
-25 000,0
30,0
35,0
40,0
45,0
50,0
55,0
60,0
65,0

More than a plot:

creating a FFT plot

FFT plot

- Right-click in the plot window and choose Data Cursor from the pop-up menu.

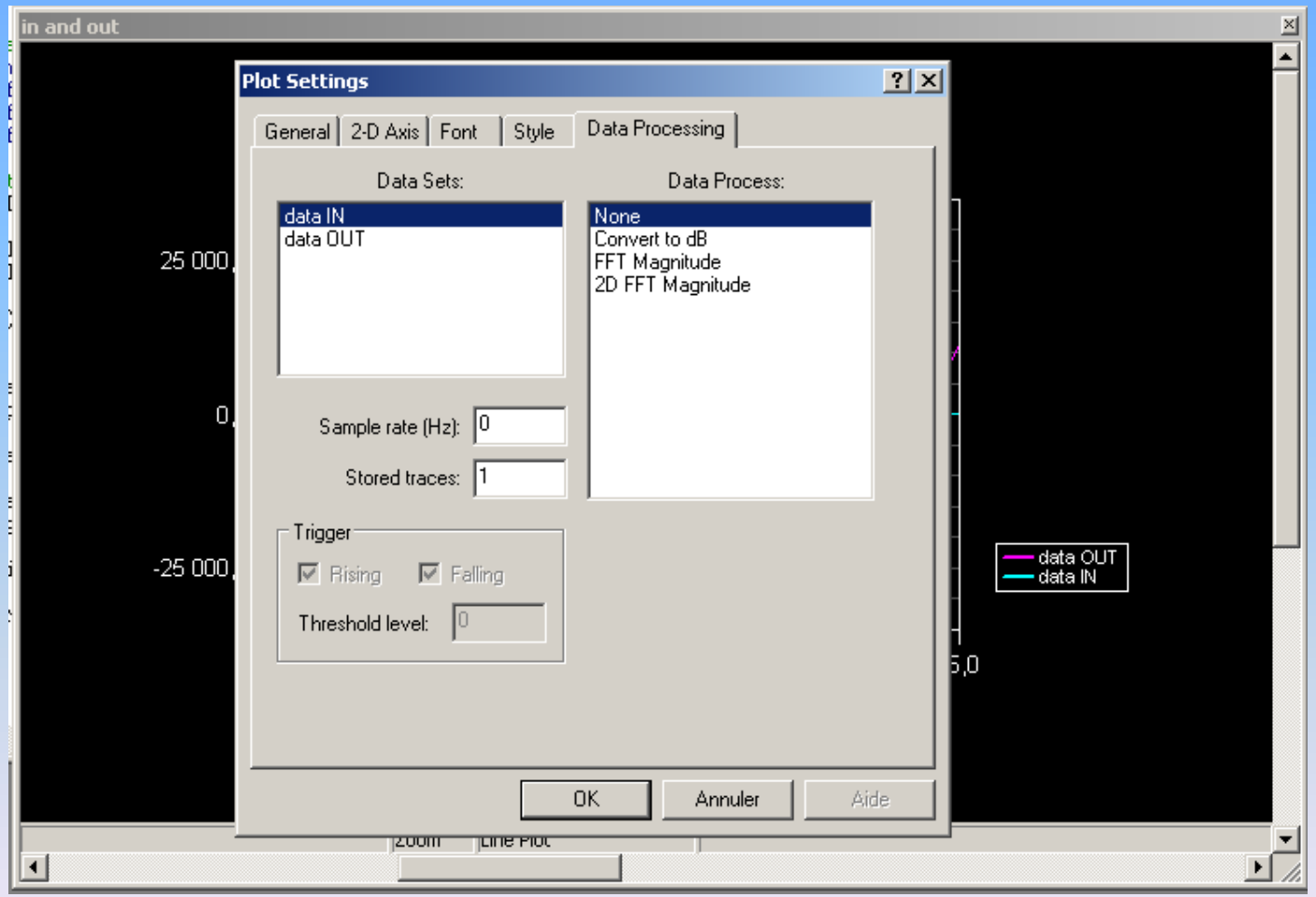
- Right-click in the plot window and choose Modify Settings to open the Plot Settings dialog box.

FFT plot

- Click the Data Processing tab to display the Data Processing page, shown in next slide.

- In the Data Sets box, ensure that Input (the default) is selected. In the Data Process box, choose FFT Magnitude.

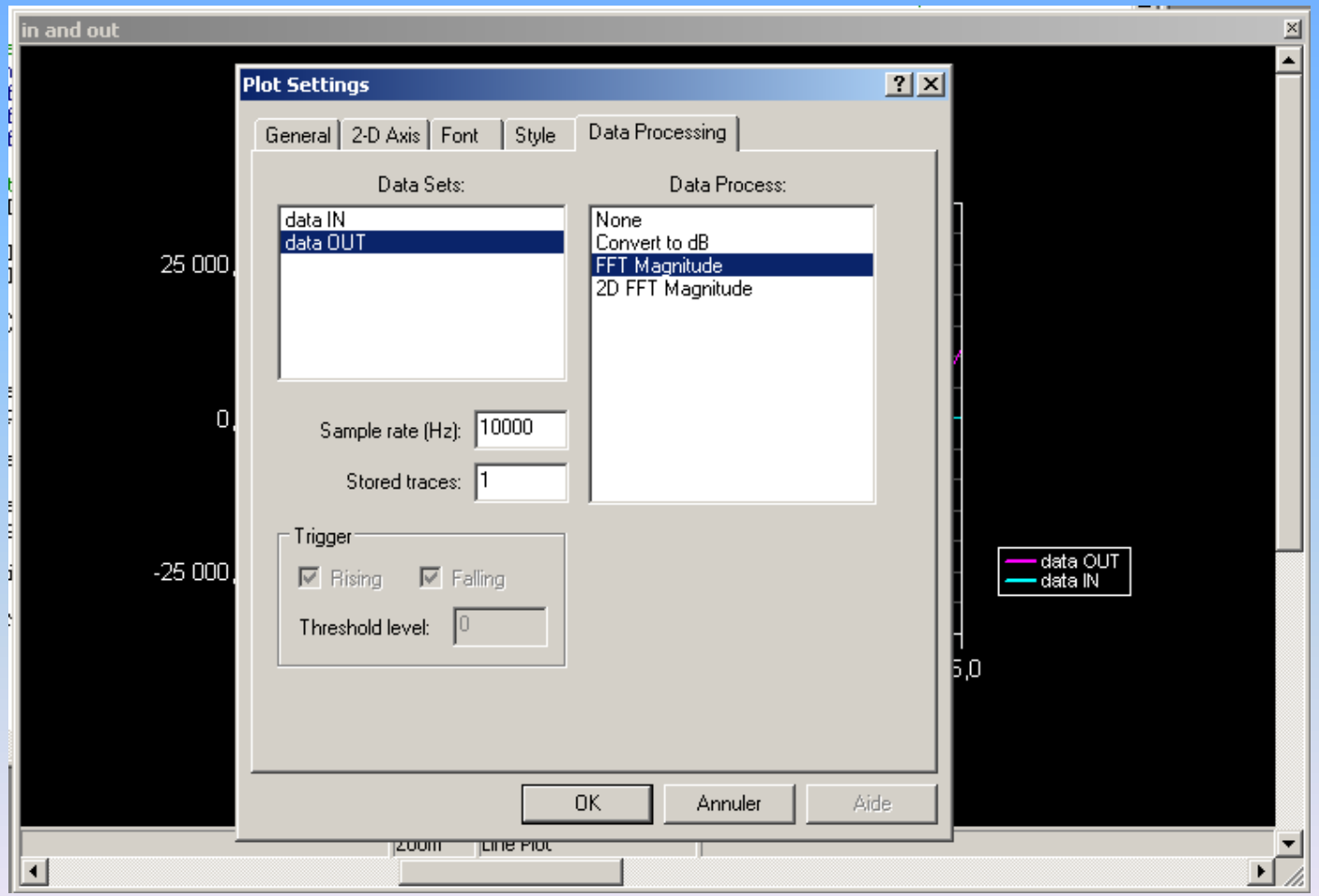
FFT plot


in and out
Plot Settings
General
2-D Axis
Font
Style
Data Processing
Data Sets:
data IN
data OUT
Data Process:
None
Convert to dB
FFT Magnitude
2D FFT Magnitude
Sample rate (Hz):
0
Stored traces:
1
Trigger
Rising
Falling
Threshold level:
0
OK
Annuler
Aide
25 000
0
-25 000
data OUT
data IN
Zoom
Line Plot

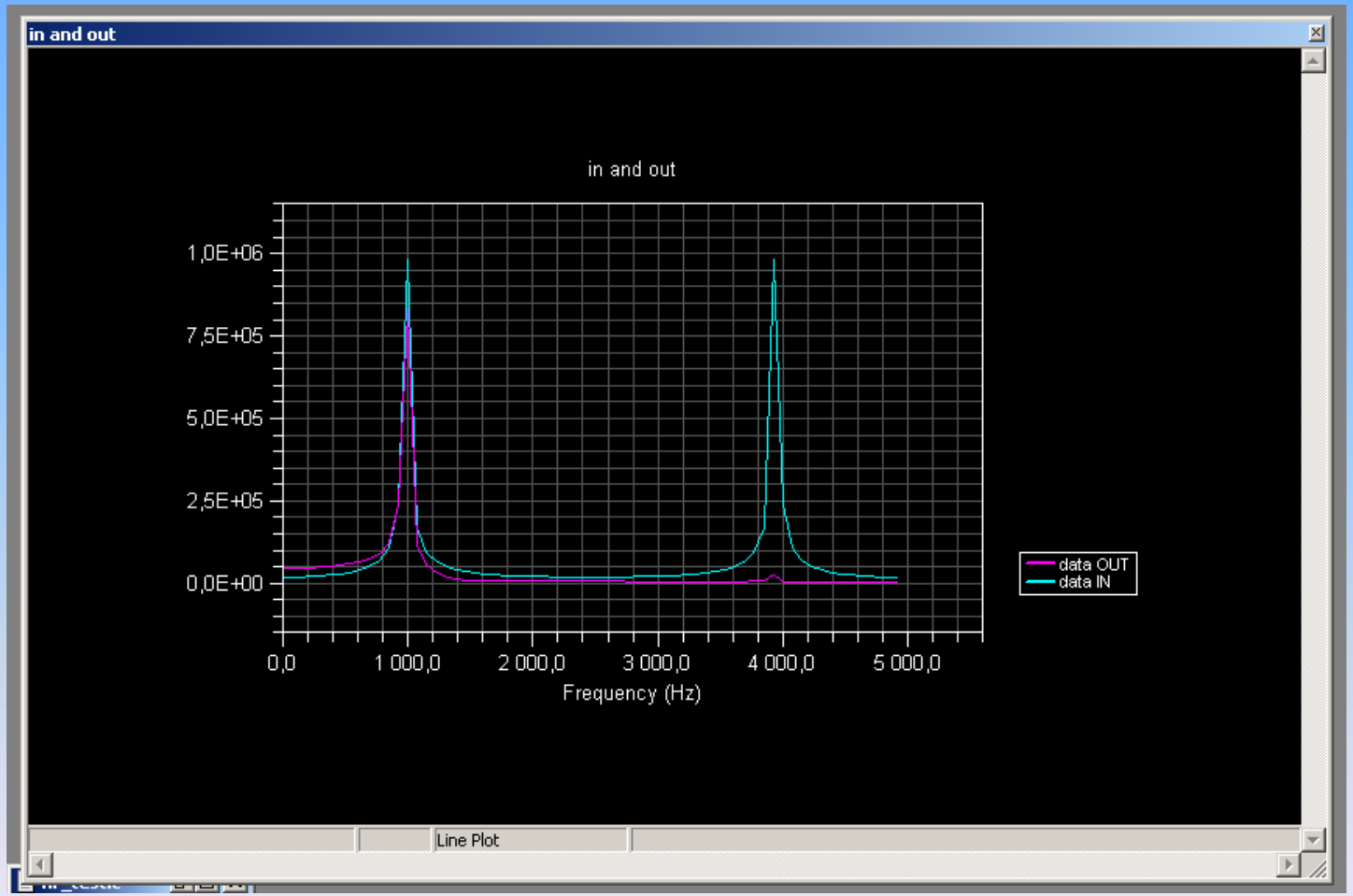
FFT plot

- In the Sample rate (Hz) box, type 10000.
- In the Data Sets box, select Output. In the Data Process box, choose FFT Magnitude
- Click OK to exit the Plot Settings dialog box.

FFT plot


in and out
Plot Settings
General
2-D Axis
Font
Style
Data Processing
Data Sets:
data IN
data OUT
Data Process:
None
Convert to dB
FFT Magnitude
2D FFT Magnitude
Sample rate (Hz):
10000
Stored traces:
1
Trigger
Rising
Falling
Threshold level:
0
OK
Annuler
Aide
25 000
0
-25 000
data OUT
data IN
Zoom
Line Plot

You should see now:



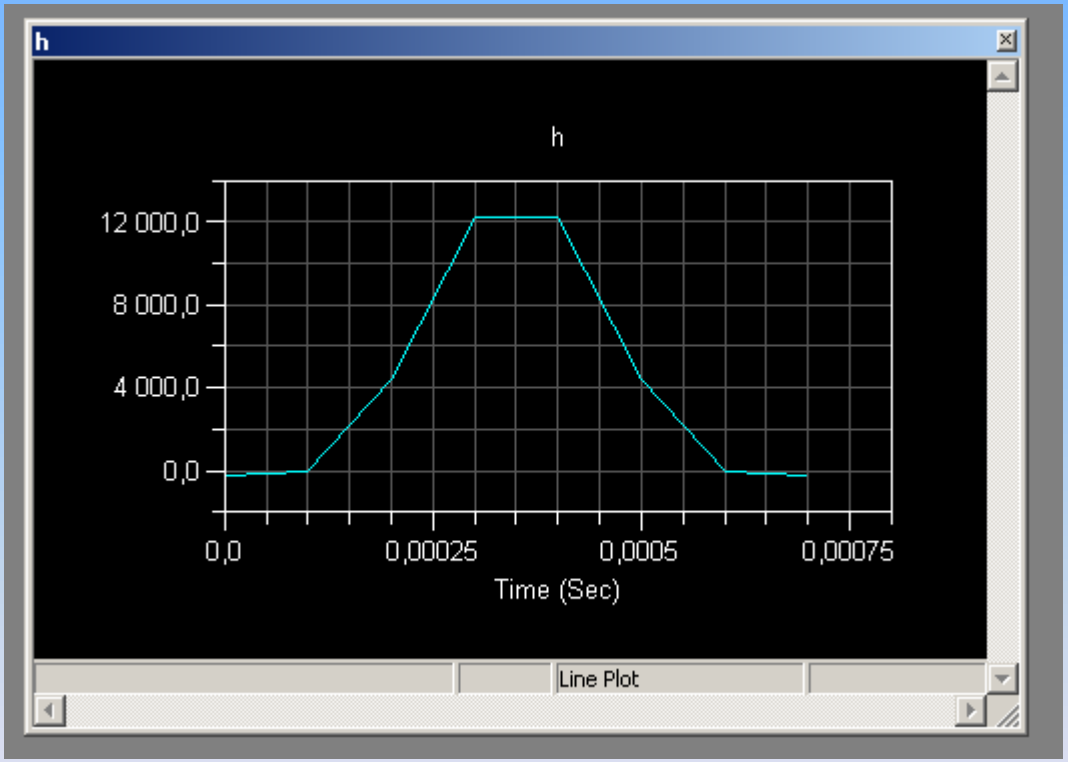
Question Set #1

- what does these Fourier transforms represent?
- from their spectrum only explain what type of signals are IN and OUT
- what type of filter is your FIR?

The FIR transfer function

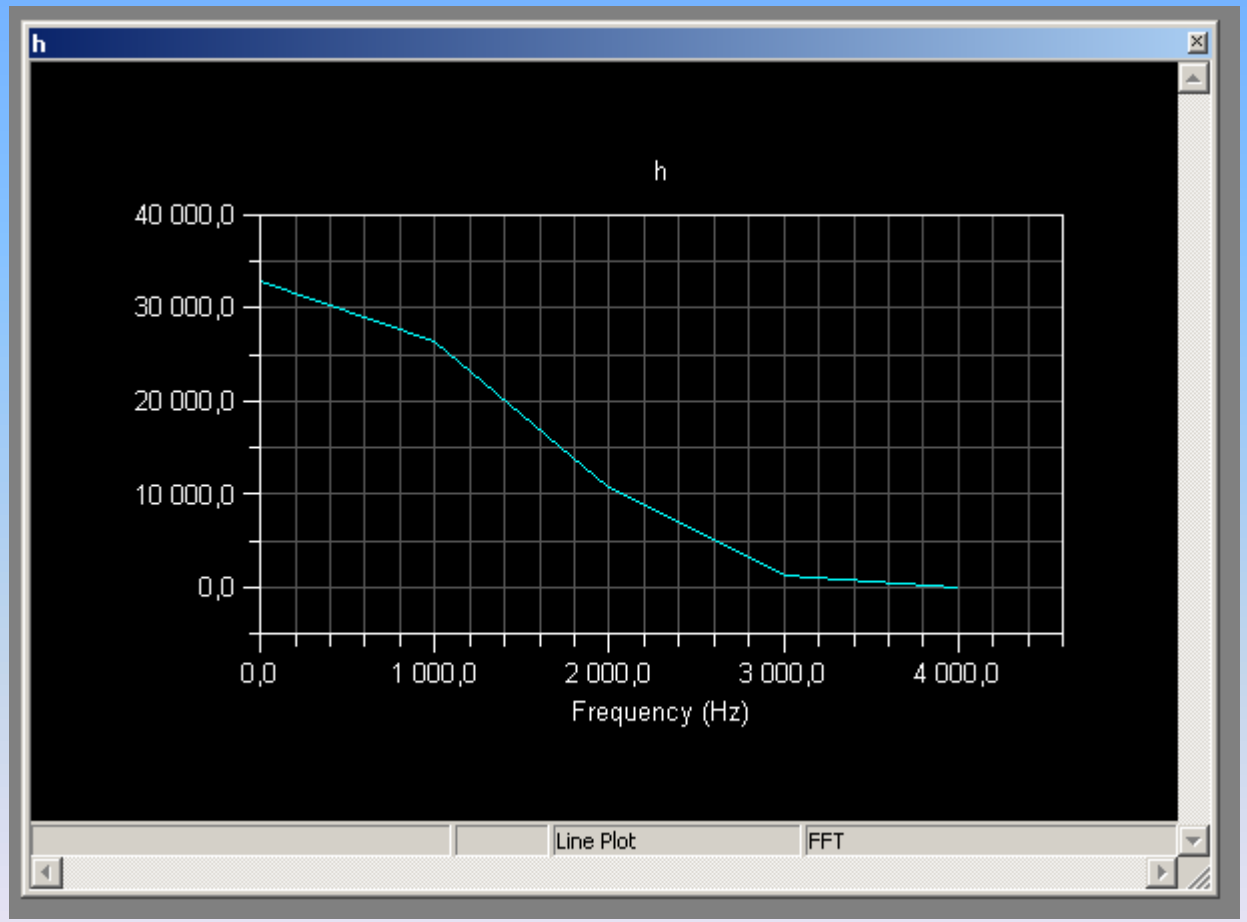
- Click the Data Processing tab to display the Data Processing page. Complete this page as follows.
- In the Data Sets box, select h.
- In the Data Process box, choose FFT Magnitude.
- In the Sample rate (Hz) box, type 10000.
- Click OK to exit the Data Processing page.

You should see:



-> now do a FFT plot of the same data

You should see now:



Question Set #2

- so what type of filter is your FIR finally?
- explain its effect on the signal IN
- could you predict beforehand the shape of the signal OUT? how?