


Signal Processing

© Dr. D. M. Akbar Hussain




Lecture 3

1

Signal Processing

© Dr. D. M. Akbar Hussain



Q Format

Range & Precision

Format	Largest positive value	Least negative value	Precision
Q0.15	0.999969482421875	-1	0.00003051757813
Q1.14	1.99993896484375	-2	0.00006103515625
Q2.13	3.9998779296875	-4	0.00012207031250
Q3.12	7.999755859375	-8	0.00024414062500
Q4.11	15.99951171875	-16	0.00048828125000
Q5.10	31.9990234375	-32	0.00097656250000
Q6.9	63.998046875	-64	0.00195312500000
Q7.8	127.99609375	-128	0.00390625000000
Q8.7	255.9921875	-256	0.00781250000000
Q9.6	511.984375	-512	0.01562500000000
Q10.5	1023.96875	-1,024	0.03125000000000
Q11.4	2047.9375	-2,048	0.06250000000000
Q12.3	4095.875	-4,096	0.12500000000000
Q13.2	8191.75	-8,192	0.25000000000000
Q14.1	16383.5	-16,384	0.50000000000000
Q15.0	32,767	-32,768	1.00000000000000

2

Converting Fractional Number in any Q Format to Integer Value (Recognized by Assembler)

1. Normalize the fractional number to the range determined by the desired Q format.
2. Multiply the result by 2^N , N is the total number of fractional bits.
3. Result is rounded to the nearest integer.

3

Scaling Factor for Q-Formats

Format	Scaling factor (2^n)	Range in Hex (Decimal value)
Q0.15	$2^{15} = 32,768$	7FFFh (0.99) → 8000h (-1)
Q1.14	$2^{14} = 16,384$	7FFFh (1.99) → 8000h (-2)
Q2.13	$2^{13} = 8,192$	7FFFh (3.99) → 8000h (-4)
Q3.12	$2^{12} = 4,096$	7FFFh (7.99) → 8000h (-8)
Q4.11	$2^{11} = 2,048$	7FFFh (15.99) → 8000h (-16)
Q5.10	$2^{10} = 1,024$	7FFFh (31.99) → 8000h (-32)
Q6.9	$2^9 = 512$	7FFFh (63.99) → 8000h (-64)
Q7.8	$2^8 = 256$	7FFFh (127.99) → 8000h (-128)
Q8.7	$2^7 = 128$	7FFFh (255.99) → 8000h (-256)
Q9.6	$2^6 = 64$	7FFFh (511.99) → 8000h (-512)
Q10.5	$2^5 = 32$	7FFFh (1023.99) → 8000h (-1,024)
Q11.4	$2^4 = 16$	7FFFh (2047.99) → 8000h (-2,048)
Q12.3	$2^3 = 8$	7FFFh (4095.99) → 8000h (-4,096)
Q13.2	$2^2 = 4$	7FFFh (8191.99) → 8000h (-8,192)
Q14.1	$2^1 = 2$	7FFFh (16383.99) → 8000h (-16,384)
Q15.0	$2^0 = 1$ (Integer)	7FFFh (32,767) → 8000h (-32,768)

4

Floating Point Format

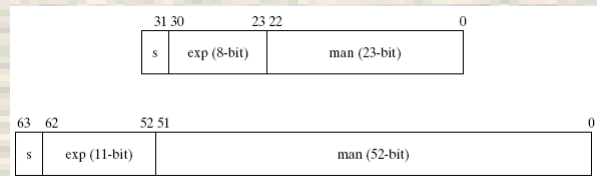
It has large dynamic range and reduces the problem of overflow.

- Three basic things (fields) for representation:

Sign Bit, Exponent Bits, Mantissa Bits.

IEEE Single Precision fp format is:

$x = -1^s \times 2^{(\text{exp} - 127)} \times 1.\text{man}$ (sign bit is b_{31} , exponent 8 bits from b_{30} to b_{23} and mantissa 23 bits from b_{22} to b_0).



5

Floating Point Addition & Multiplication

For addition, fp processor needs to adjust the exponent of the smaller number to match the bigger number, e.g. :


$$x = -1^{s_x} \times 2^{(\text{exp}_x - 127)} \times 1.\text{man}_x,$$

$$y = -1^{s_y} \times 2^{(\text{exp}_y - 127)} \times 1.\text{man}_y,$$

$$z = x + y = \begin{cases} \left[-1^{s_x} \times 1.\text{man}_x + (-1^{s_y} \times 1.\text{man}_y \times 2^{-(\text{exp}_x - \text{exp}_y)}) \right] \times 2^{-(\text{exp}_x - 127)} & \text{if } |x| \geq |y| \\ \left[-1^{s_y} \times 1.\text{man}_y + (-1^{s_x} \times 1.\text{man}_x \times 2^{-(\text{exp}_y - \text{exp}_x)}) \right] \times 2^{-(\text{exp}_y - 127)} & \text{if } |y| > |x| \end{cases}$$

6

Signal Processing



© Dr. D. M. Akbar Hussain

Example Floating Point Addition

For addition, fp processor needs to adjust the exponent of the smaller number to match the bigger number, e.g. :

$$x = -1^{s_x} \times 2^{(\text{exp}_x - 127)} \times 1.\text{man}_x,$$


$$y = -1^{s_y} \times 2^{(\text{exp}_y - 127)} \times 1.\text{man}_y,$$

$$z = 2.44 + (-12.16)$$

$$= \begin{cases} \left[-1^{s_x} \times 1.\text{man}_x + \left(-1^{s_y} \times 1.\text{man}_y \times 2^{-(\text{exp}_x - \text{exp}_y)} \right) \right] \times 2^{(\text{exp}_x - 127)} & \text{if } |x| \geq |y| \\ \left[-1^{s_y} \times 1.\text{man}_y + \left(-1^{s_x} \times 1.\text{man}_x \times 2^{-(\text{exp}_y - \text{exp}_x)} \right) \right] \times 2^{(\text{exp}_y - 127)} & \text{if } |y| > |x| \end{cases}$$

7

Signal Processing



© Dr. D. M. Akbar Hussain

Example Floating Point Multiplication

For addition, fp processor needs to adjust the exponent of the smaller number to match the bigger number, e.g. :

$$x = -1^{s_x} \times 2^{(\text{exp}_x - 127)} \times 1.\text{man}_x,$$

$$y = -1^{s_y} \times 2^{(\text{exp}_y - 127)} \times 1.\text{man}_y,$$


$$z = 2.44 * (-12.16)$$

$$= \left[(-1^{s_x} \times 1.\text{man}_x) \times (-1^{s_y} \times 1.\text{man}_y) \right] \times 2^{(\text{exp}_x + \text{exp}_y - 254)}$$

8

Signal Processing

© Dr. D. M. Akbar Hussain




Division



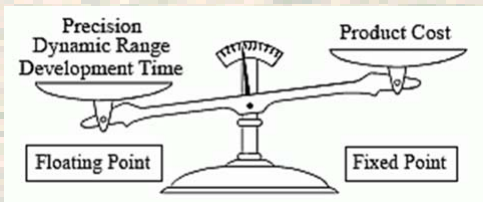
9

Signal Processing

© Dr. D. M. Akbar Hussain



Fixed Point vs Floating Point




Fixed point DSP's are generally cheaper, while floating point devices have better precision, higher dynamic range and a shorter development cycle.

The Scientist and Engineer's Guide to Digital Signal Processing
By Steven W. Smith, Ph.D.

10

Signal Processing

© Dr. D. M. Akbar Hussain




**Floating Point
Range & Precision**

	Dynamic range	Dynamic range in dB	Precision
Single precision	1.18×10^{-38} to 3.4×10^{38}	$\sim 1,530$ dB (6 dB $\times 255$)	2^{-23}
Extended single precision	1.18×10^{-38} to 3.4×10^{38}	$\sim 1,530$ dB	2^{-31}
Double precision	2^{-1022} to 2^{1024}	$6 \times (2^{11} - 1) \sim 12,282$ dB	2^{-52}

11

Signal Processing

© Dr. D. M. Akbar Hussain




Selection Guide

Fixed-point processors	Floating-point processors
16- or 24-bit devices	32-bit devices
Limited dynamic range	Large dynamic range
Overflow and quantization errors must be resolved	Easier to program since no scaling is required
Poorer C-compiler efficiency; normally programmed in assembly	Better C-compiler efficiency; can be developed in C
Long product development time	Quick time to market
Faster clock rate	Slower clock rate
Less silicon area is required; functional units are simpler	More silicon area is required; functional units are complex
Cheaper	More expensive
Lower power consumption	Higher power consumption

12

Signal Processing

© Dr. D. M. Akbar Hussain




Applications

Fixed-point processors	Floating-point processors
Disk drive and motor control	Image processing in radar, sonar, and seismic applications
Consumer audio applications such as MP3 players, multimedia gaming, and digital cameras	High-end audio applications such as ambient acoustics simulators, professional audio encoding/decoding, and audio mixing
Speech coding/decoding and channel coding	Sound synthesis in professional audio and video coding/decoding
Communication devices such as modems and cellular phones	Prototyping

13

Signal Processing

© Dr. D. M. Akbar Hussain

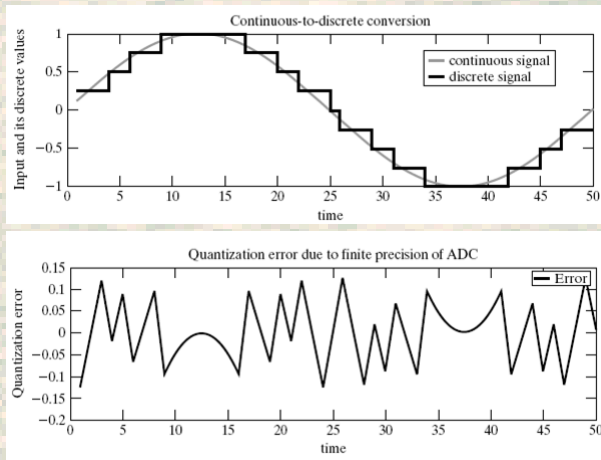


Recap: Finite Word Length Effect

- **Input Quantization (input analogue signal is converted to digital values).**
- **Limited precision in representing the system coefficients.**
- **Limited dynamic range so over flow can occur.**
- **Rounding and truncating.**

14

Input Quantization



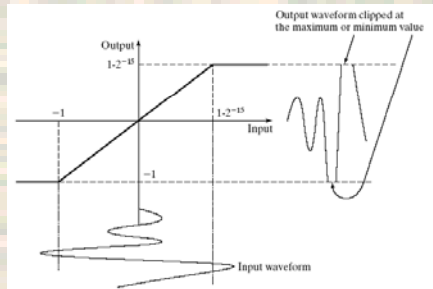
Coefficient Quantization

Signal Processing



© Dr. D. M. Akbar Hussain

- ❑ **Overflow**
- ❑ **Saturation Mode**
- ❑ **Scaling Input Signal**
- ❑ **Guard Bits in Accumulator**



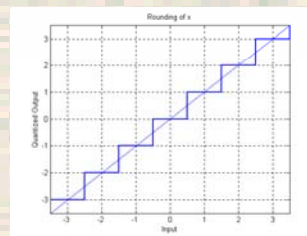
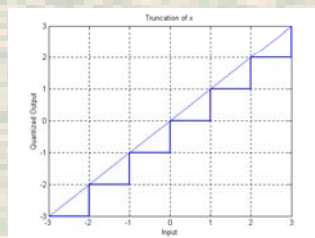
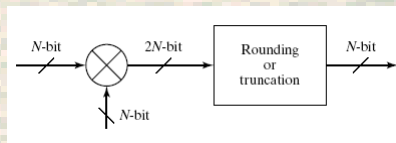
17

Signal Processing



© Dr. D. M. Akbar Hussain

Rounding & Truncating



18

Signal Processing

© Dr. D. M. Akbar Hussain

AALBORG UNIVERSITET
ESBJERG

System Considerations

Interfacing

Performance

Power

Size

Ease-of Use

- Programming
- Interfacing
- Debugging

Cost

- Device cost
- System cost
- Development cost
- Time to market

Integration

- Memory
- Peripherals

19

Signal Processing

© Dr. D. M. Akbar Hussain

AALBORG UNIVERSITET
ESBJERG

Different Needs? Multiple Families

C2000
(C20x/24x/28x)
•C1x •C2x

C5000
(C54x/55x/OMAP)
•C5x

C6000
(C62x/64x/67x)
•C3x •C4x •C8x

Lowest Cost

Control Systems

- Segway
- Motor Control
- Storage
- Digital Ctrl Systems

Efficiency

Best MIPS per Watt / Dollar / Size

- Wireless phones
- Internet audio players
- Digital still cameras
- Modems
- Telephony
- VoIP

Max Performance with Best Ease-of-Use

- Multi Channel and Multi Function App's
- Wireless Base-stations
- DSL
- Imaging & Video
- Home Theater
- Performance Audio
- Multi-Media Servers
- Digital Radio

20

Signal Processing



© Dr. D. M. Akbar Hussain

Hardware & Software (TMS320C6416)

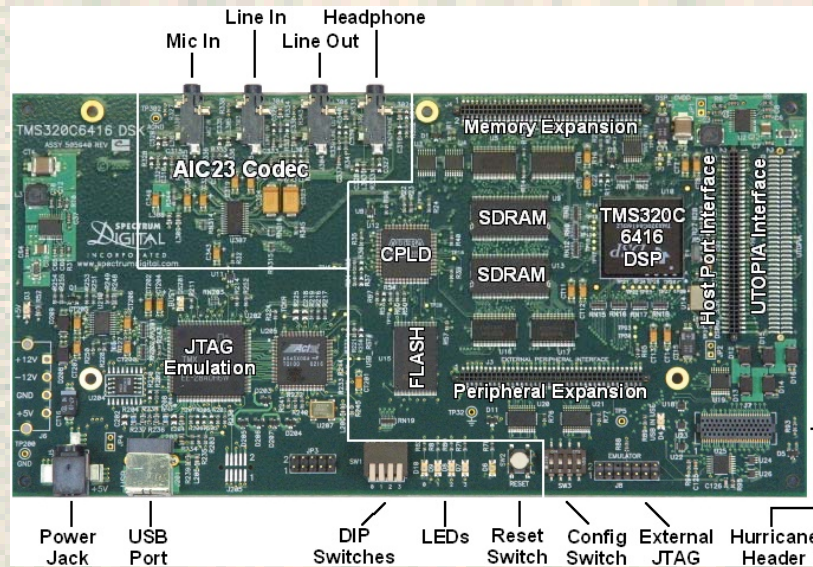
- ◆ 600 MHz 'C6416 DSP
- ◆ AIC23 Stereo Codec
- ◆ External Memory
 - 16M Bytes SDRAM
 - 512K Bytes Flash ROM
- ◆ 4 user accessible LED's and DIP Switches
- ◆ Daughter card expansion
- ◆ Software Board Configuration through registers implemented in CPLD
- ◆ JTAG Emulation through on-board JTAG emulator with USB host interface or external emulator
- ◆ Power Supply & Parallel Port Cable

21

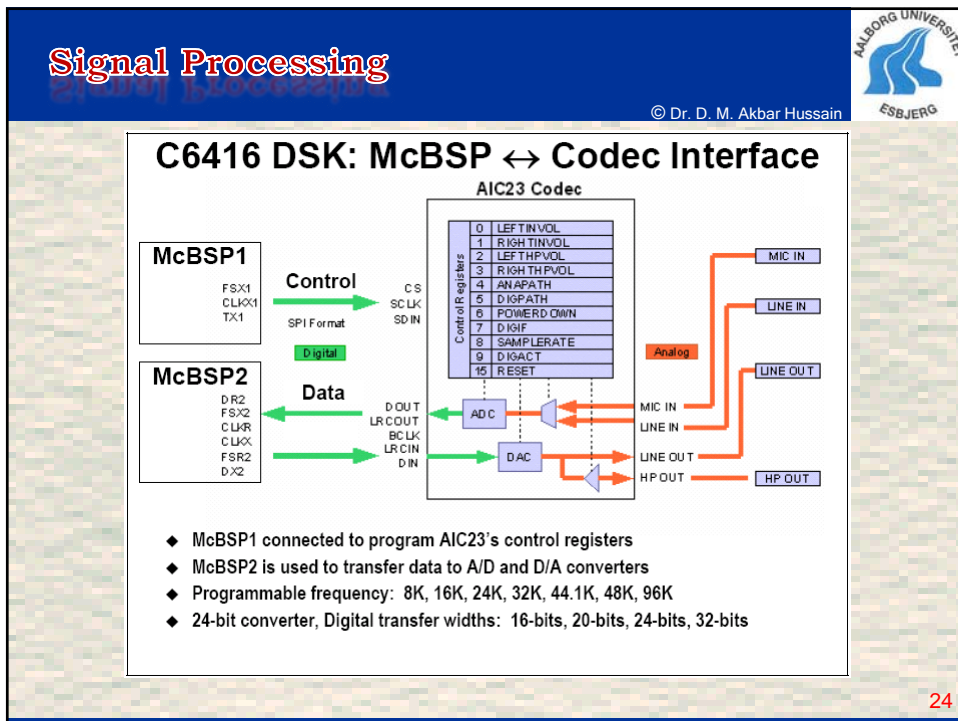
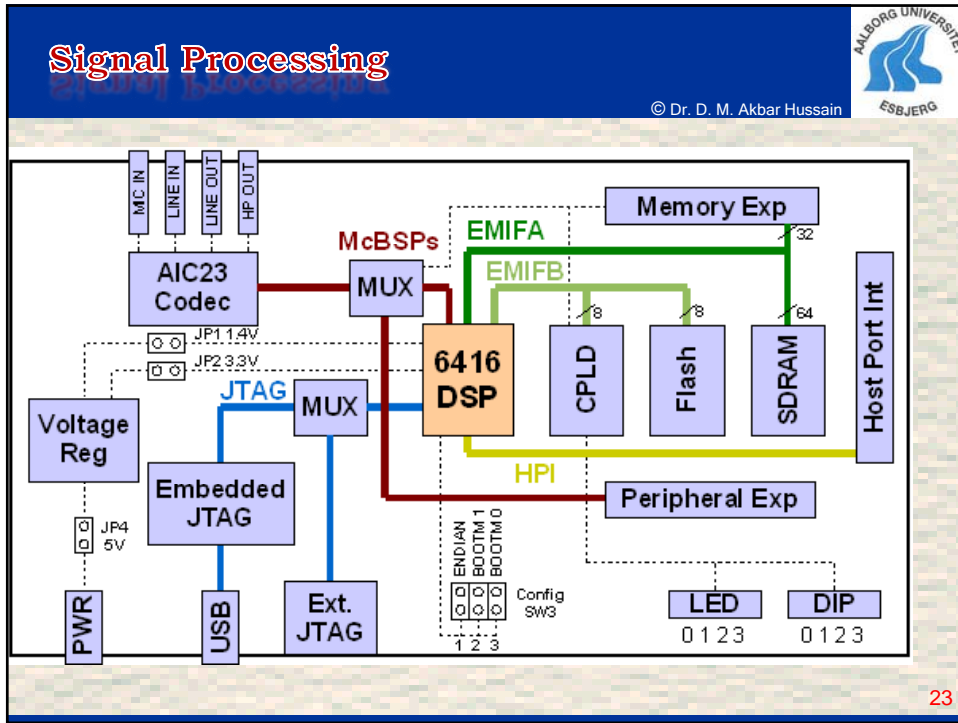
Signal Processing



© Dr. D. M. Akbar Hussain




22



Signal Processing

© Dr. D. M. Akbar Hussain

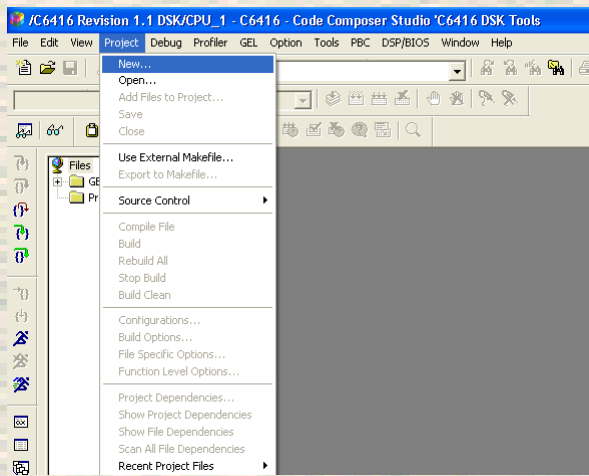



Project Creation Steps For Code Composer

25

Signal Processing

© Dr. D. M. Akbar Hussain



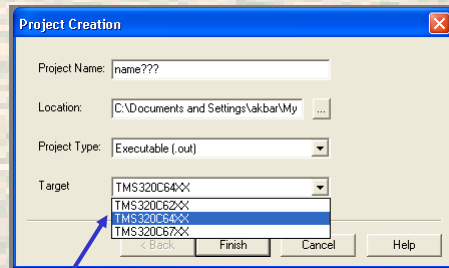
The screenshot shows the Code Composer Studio interface. The title bar reads "/C6416 Revision 1.1 DSK/CPU_1 - C6416 - Code Composer Studio 'C6416 DSK Tools'". The menu bar includes File, Edit, View, Project, Debug, Profiler, GEL, Option, Tools, PBC, DSP/BIOS, Window, and Help. The Project menu is open, displaying options such as New..., Open..., Add Files to Project..., Save, Close, Use External Makefile..., Export to Makefile..., Source Control (with sub-options like Compile File, Build, Rebuild All, Stop Build, Build Clean), Configurations..., Build Options..., File Specific Options..., Function Level Options..., Project Dependencies..., Show Project Dependencies, Show File Dependencies, Scan All File Dependencies, and Recent Project Files.

26

Signal Processing



© Dr. D. M. Akbar Hussain



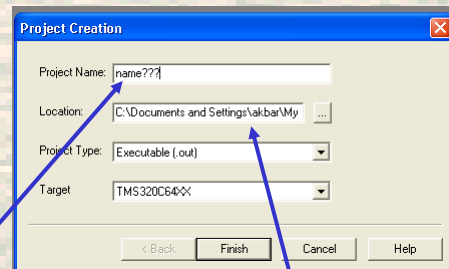
Selecting Appropriate Processor

27

Signal Processing



© Dr. D. M. Akbar Hussain



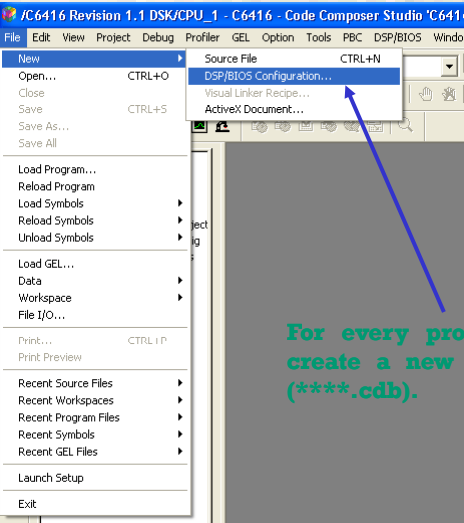
Selecting Folder where you want to create the project

Name of the project

28

Signal Processing

© Dr. D. M. Akbar Hussain

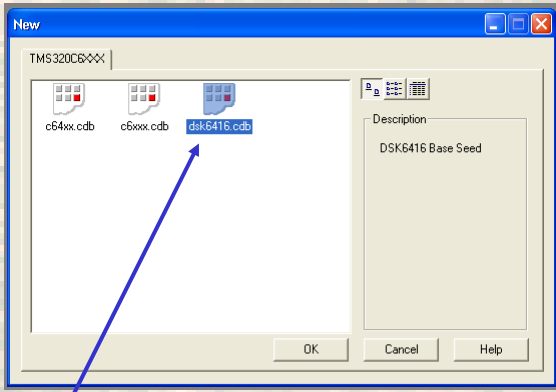


For every project you need to create a new configuration file (**.cdb).**

29

Signal Processing

© Dr. D. M. Akbar Hussain



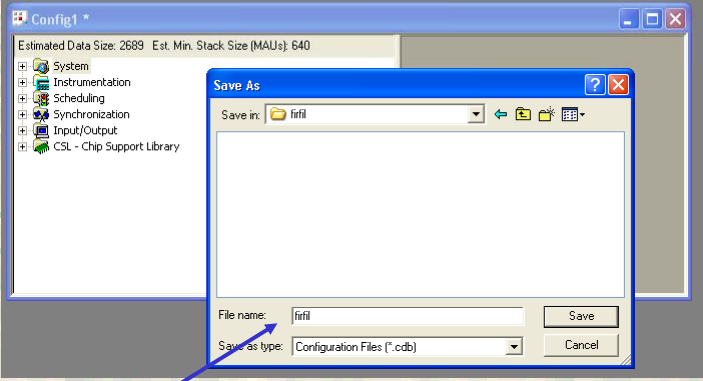
Selecting Appropriate dsk Board (In our case dsk6416.cdb)

30

Signal Processing

AALBORG UNIVERSITET
ESBJERG

© Dr. D. M. Akbar Hussain



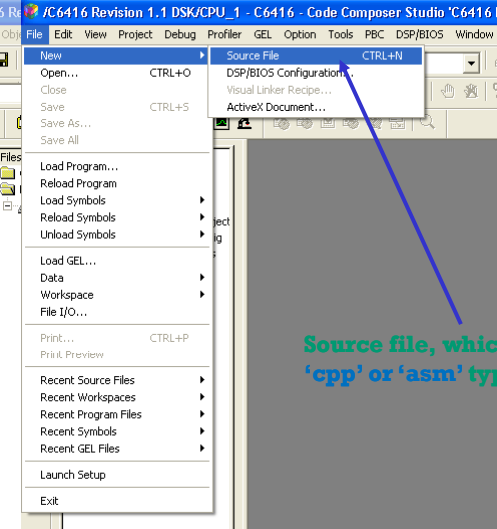
Advice: always give the same name as your project name.

31

Signal Processing

AALBORG UNIVERSITET
ESBJERG

© Dr. D. M. Akbar Hussain



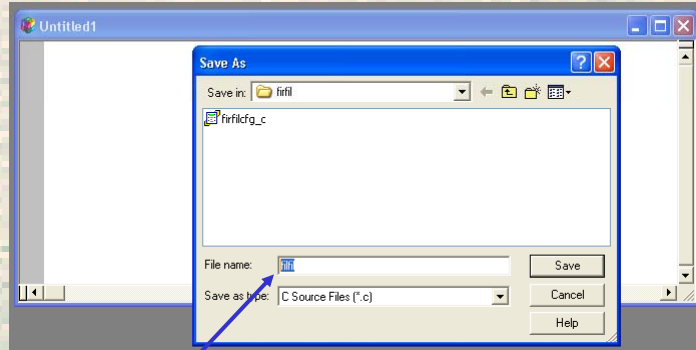
Source file, which could be a 'c', 'cpp' or 'asm' type.

32

Signal Processing



© Dr. D. M. Akbar Hussain



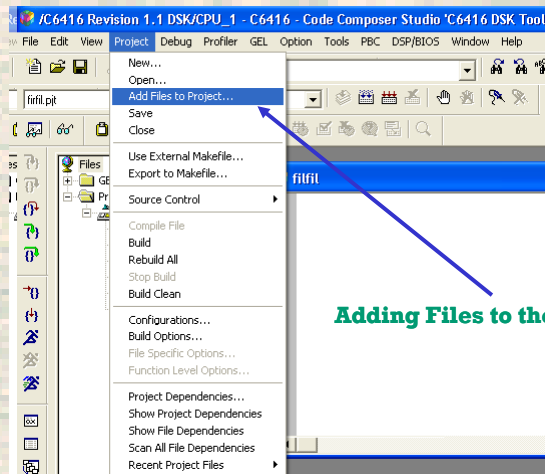
In this case we are creating a 'c' type file, so don't give the extension it automatically creates it for you.

33

Signal Processing



© Dr. D. M. Akbar Hussain



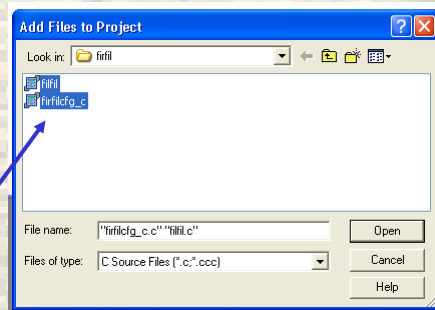
Adding Files to the Project

34

Signal Processing



© Dr. D. M. Akbar Hussain



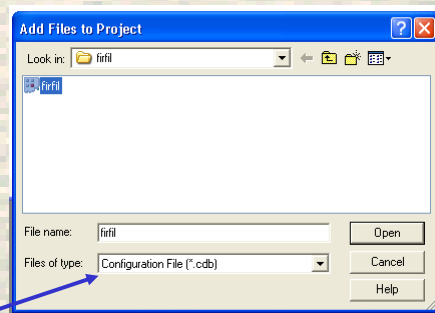
You will find two types of 'c' files one you just created as your source and the second is created by the system with the same name as your project with extension 'cfg_c'.

35

Signal Processing



© Dr. D. M. Akbar Hussain



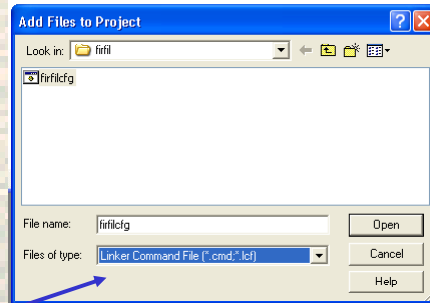
You also need to add the configuration file you created earlier.

36

Signal Processing



© Dr. D. M. Akbar Hussain



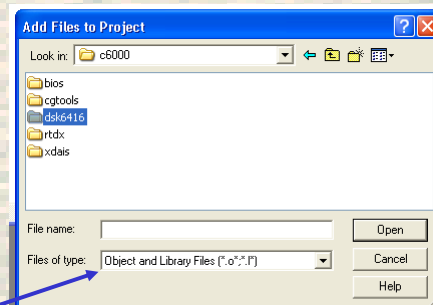
You also need to add the linker command file ????.cmd which system creates.

37

Signal Processing



© Dr. D. M. Akbar Hussain



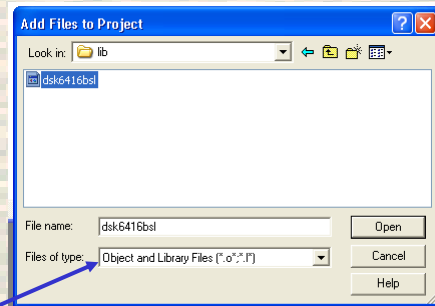
If you are using library functions then you need to add the board specific library file, which you can find at the top folder (ti) where you actually installed the code composer.

38

Signal Processing



© Dr. D. M. Akbar Hussain



In our case the board is dsk6416 therefore, we select 'dsk6416bsl.l'.

39

Signal Processing



© Dr. D. M. Akbar Hussain

Once you have added this file 'dsk6416bsl.l', please go to the source file and add the following line into that file:

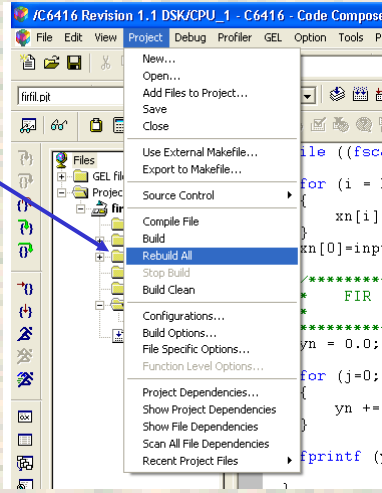
```
#include "firfilcfg.h"
```

You can see this has two parts first part is the name of the project and second is extended with `cfg.h`.

40

Signal Processing

© Dr. D. M. Akbar Hussain



Finally Compiling

The screenshot shows the Code Composer Studio interface. The 'Project' menu is open, and the 'Build' option is selected, which has opened a sub-menu where 'Rebuild All' is highlighted. A blue arrow points from the text 'Finally Compiling' to the 'Rebuild All' option. The background shows a project tree on the left and a code editor on the right with some C code.

41

Signal Processing

© Dr. D. M. Akbar Hussain

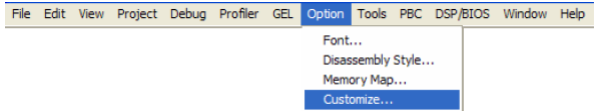
Customizing

Set CCS – Customize Options

There are a few option settings that need to be verified before we begin. Otherwise, the lab procedure may be difficult to follow.

- Disable open Disassembly Window upon load
- Go to main() after build
- Program load after build
- Clear breakpoints when loading a new program
- Set CCS Titlebar information

Open the Options→Customize Dialog.



The screenshot shows the 'Option' menu in Code Composer Studio. The 'Customize...' option is selected and highlighted in blue. Other options visible include 'Font...', 'Disassembly Style...', and 'Memory Map...'. The menu bar at the top includes 'File', 'Edit', 'View', 'Project', 'Debug', 'Profiler', 'GEL', 'Option', 'Tools', 'PBC', 'DSP/BIOS', 'Window', and 'Help'.

42

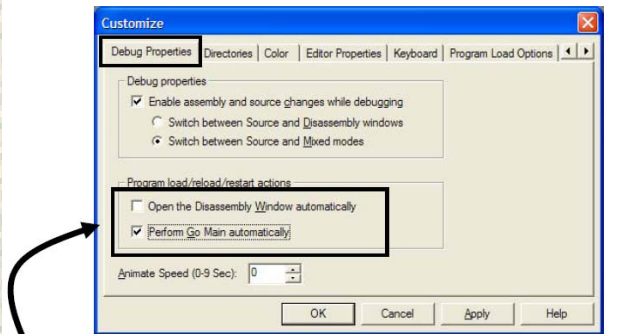
Signal Processing



© Dr. D. M. Akbar Hussain

Customizing

Set Debug Properties



- Here are a couple options that can help make debugging easier.
- Unless you want the Disassembly Window popping up every time you load a program (which annoys many folks), deselect this option.
 - Many find it convenient to choose the "Perform Go Main automatically". Whenever a program is loaded the debugger will automatically run thru the compilers initialization code to your main() function.

43

Signal Processing



© Dr. D. M. Akbar Hussain

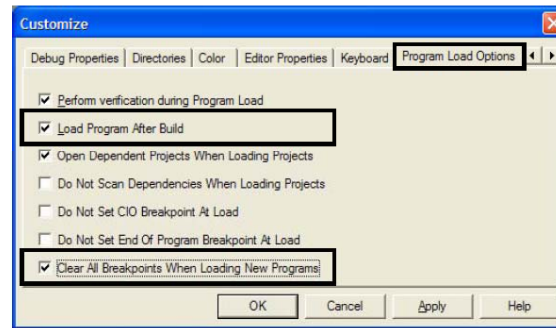
Customizing

Set Program Load Options

On the "Program Load Options" tab, select the two following options:

- Load Program After Build
- Clear All Breakpoints When Loading New Programs


By default, these options are not enabled, though a previous user of your computer may have already enabled them.



44

Signal Processing

© Dr. D. M. Akbar Hussain



Customizing

Conceptually, the CCS Integrated Development Environment (IDE) is made up of two parts:


- **Edit** (and Build) programs
Uses editor and code gen tools to create code.
- **Debug** (and Load) programs
Communicates with processor/simulator to download and run code.

The **Load Program After Build** option automatically loads the program (.out file) created when you build a project. If you disabled this automatic feature, you would have to manually load the program via the **File→Load Program** menu.

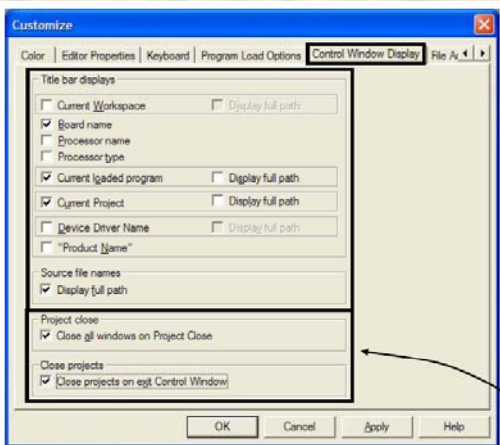
45

Signal Processing

© Dr. D. M. Akbar Hussain



Title Bar Properties



- We have chosen the "Board Name", "Current Project", and "Currently loaded program".
- The first item allows you to quickly confirm the chosen target (simulator, DSK, etc.).
- The other two let us quickly determine which project is active and what program we have loaded. Notice how these correlate to the two parts of CCS: Edit and Debug.
- For our convenience we have also enabled the remaining two features on this dialog page.

46

Signal Processing

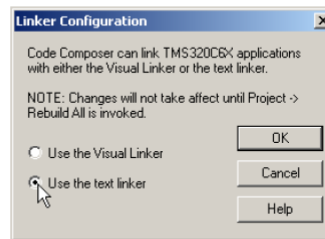


© Dr. D. M. Akbar Hussain

Text Based Linker

Open the CCS linker selection dialog.

Tools → Linker Configuration



47