Chapter 3
Code Composer Studio and the DSK

Learning Objectives

- Introduction to Code Composer Studio (CCS).
- Installation and setup of CCS.
- Introduction to the DSK.
- Laboratory.

Code Composer Studio

- The Code Composer Studio (CCS) application provides an integrated environment with the following capabilities:
  - Integrated development environment with an editor, debugger, project manager, profiler, etc.
  - 'C/C++' compiler, assembly optimiser and linker (code generation tools).
  - Simulator.
  - Real-time operating system (DSP/BIOS™).
  - Real-Time Data Exchange (RTDX™) between the Host and Target.
  - Real-time analysis and data visualisation.

CCS Installation and Setup

(A) Install the CCS Software.

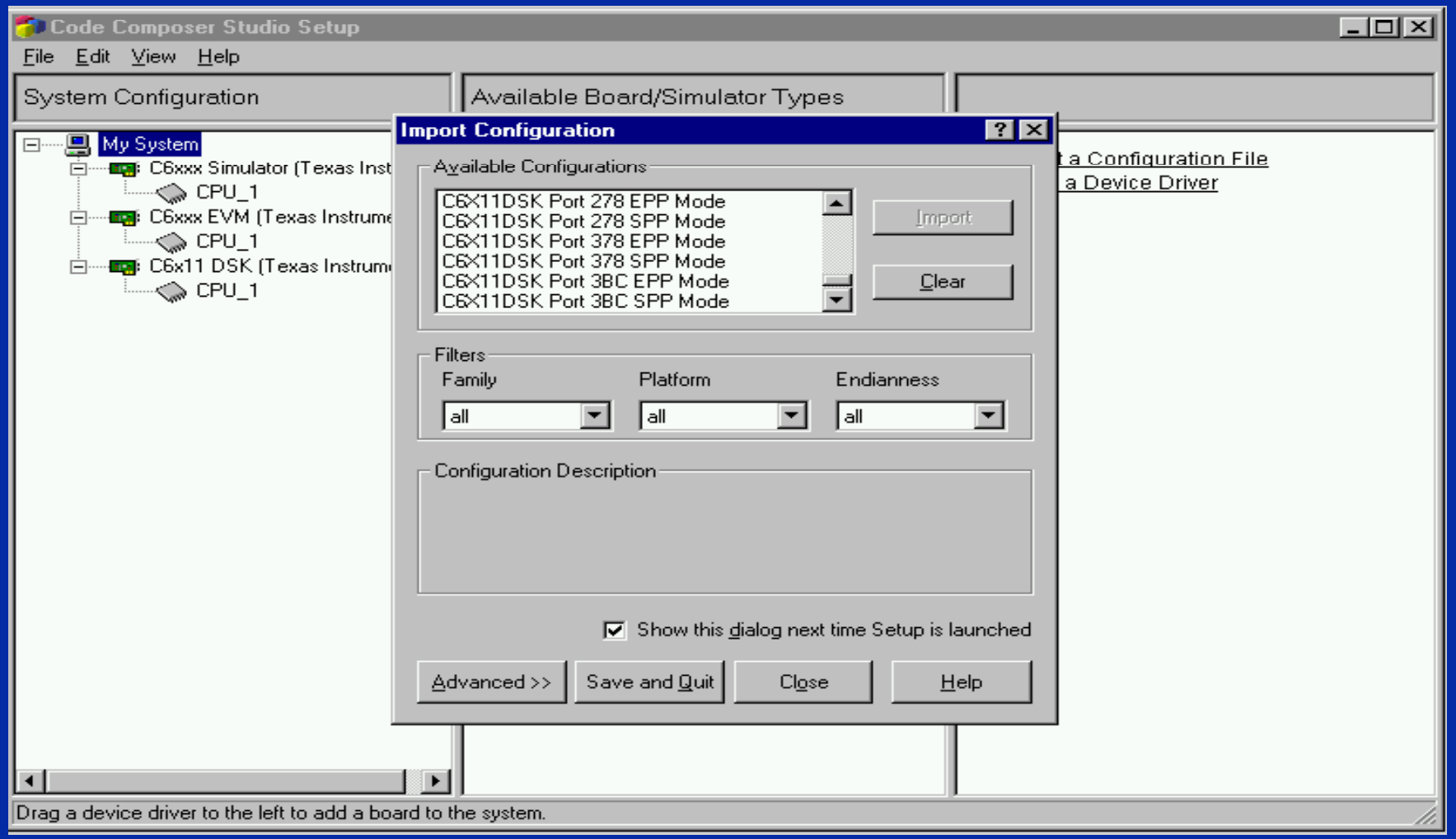
(B) Run CCS Setup:

- Start CCS setup utility by using the following desktop icon:

- Alternatively:
  - Windows Start Menu -> Programs -> Texas Instruments -> Code Composer Studio 2 (‘C6000) -> Setup Code Composer Studio.
  - Run cc_setup.exe located in: c:\ti\cc\bin\

CCS Installation and Setup

- **You should now see a screen similar to this:**



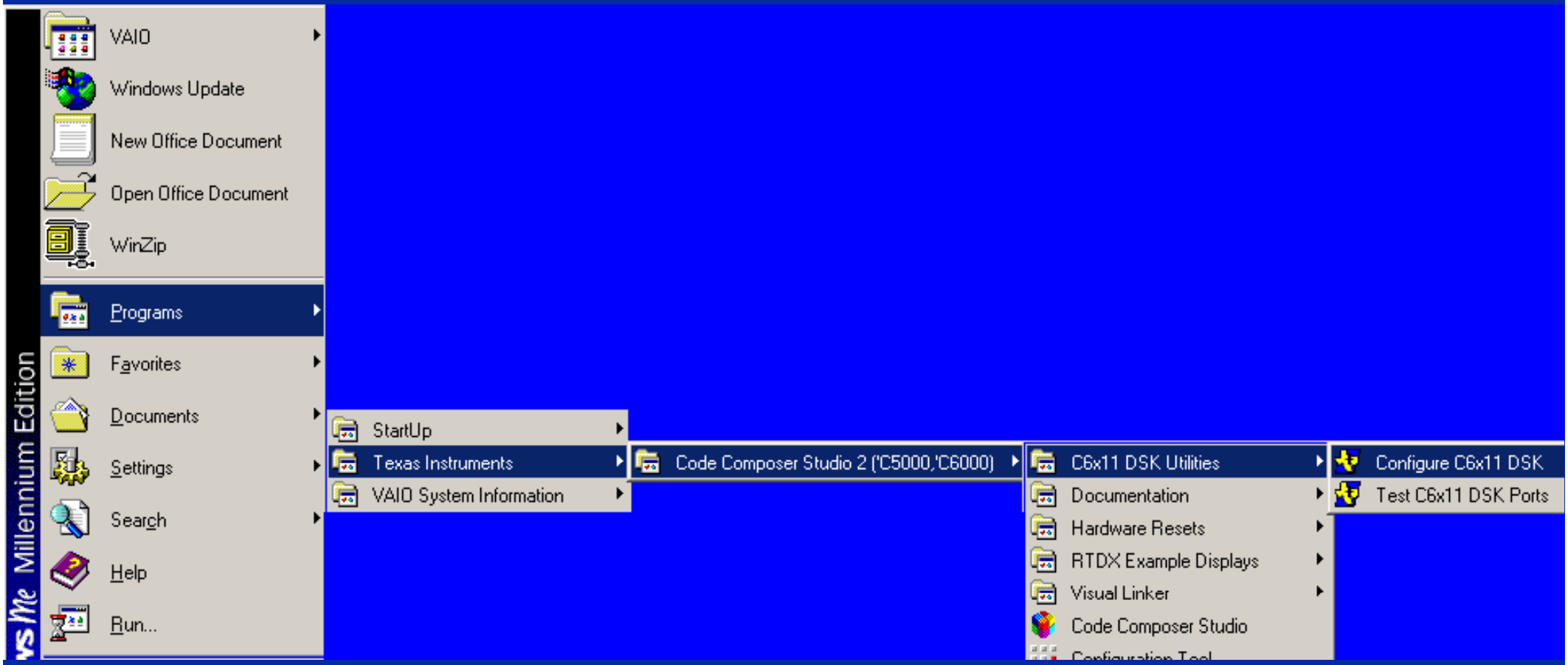
Note: If you don't see the Import Configuration dialog box you should open it from the menu using: File:Import.

CCS Installation and Setup

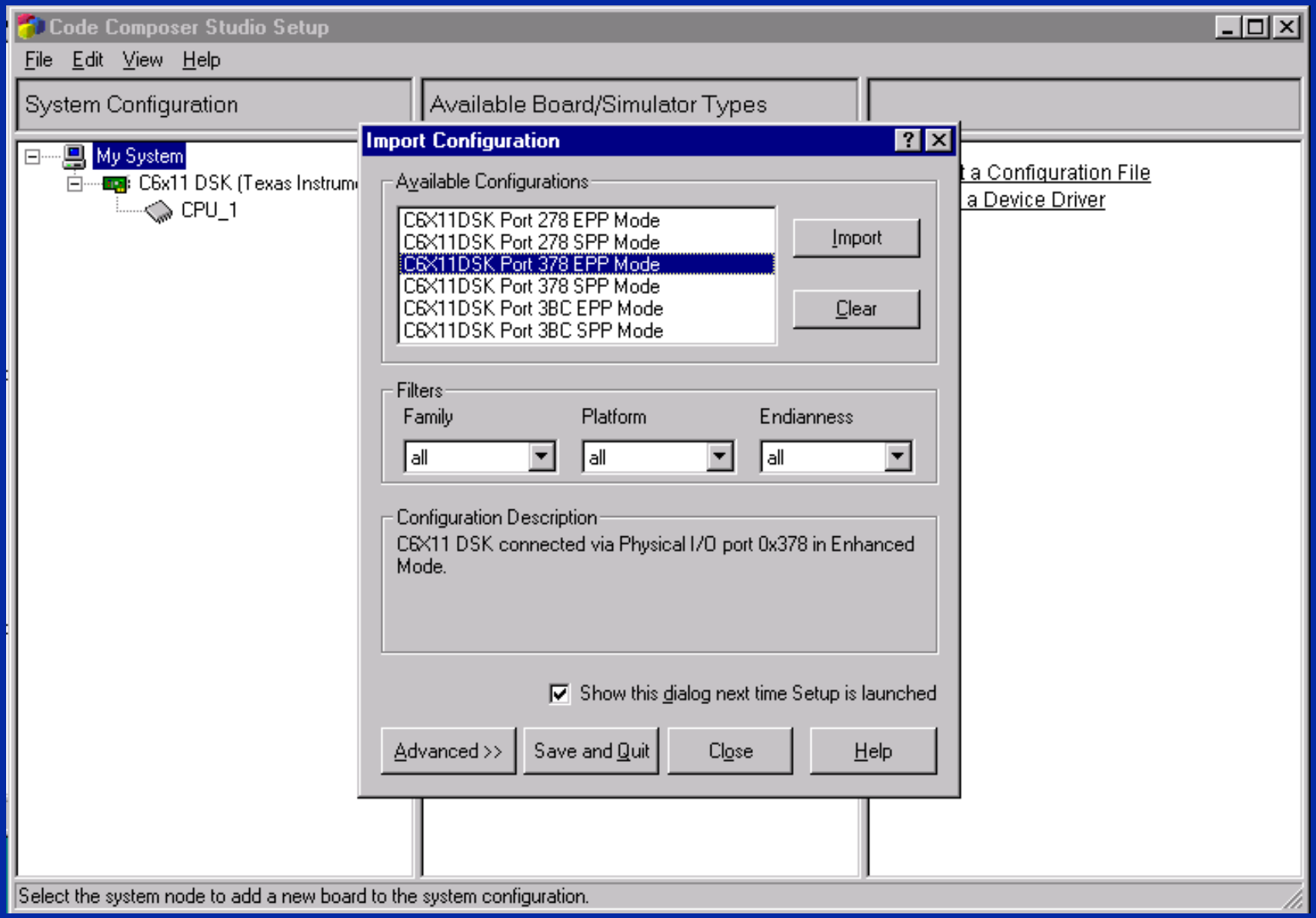
- You can clear the previous configuration by selecting the configuration you wish to clear and clicking the clear button.
- Next select a new configuration that you would like to add:
  - Select the C6x11DSK Port x y Mode.
  - The port number, x, and port mode, y, depend on your PC setup.

CCS Installation and Setup

- If you do not know your configuration then you can select it automatically using the “Configure C6x11 DSK” Utility.



CCS Installation and Setup



- **Finally save and quit the import configuration dialog box.**

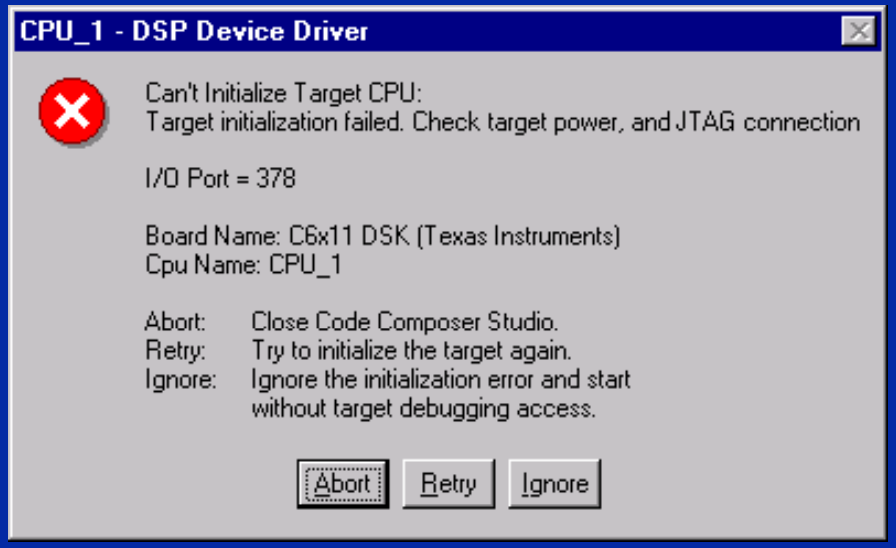
Using CCS

- **Start CCS by either:**
  - **Using the desktop icon:**

  - **Start -> Programs -> Texas Instruments -> Code Composer Studio 2 -> Code Composer Studio.**
  - **Run cc_app.exe in c:\ti\cc\bin**

Troubleshooting

- If the following window appears on your screen then:
  - Check that the DSK is connected properly and powered up.
  - Check if the port address and mode is correct (See Slide 7).



Introduction to the ‘C6711 DSK

- The ‘C6711 DSK provides a powerful, low-cost development environment.
- The DSK comes with CCS code development tools (does not include the simulator).
- A laboratory at the end of this section takes you through the DSK setup and shows you how to run the confidence test to check it is working correctly.

DSK Contents

Hardware (Schematics)

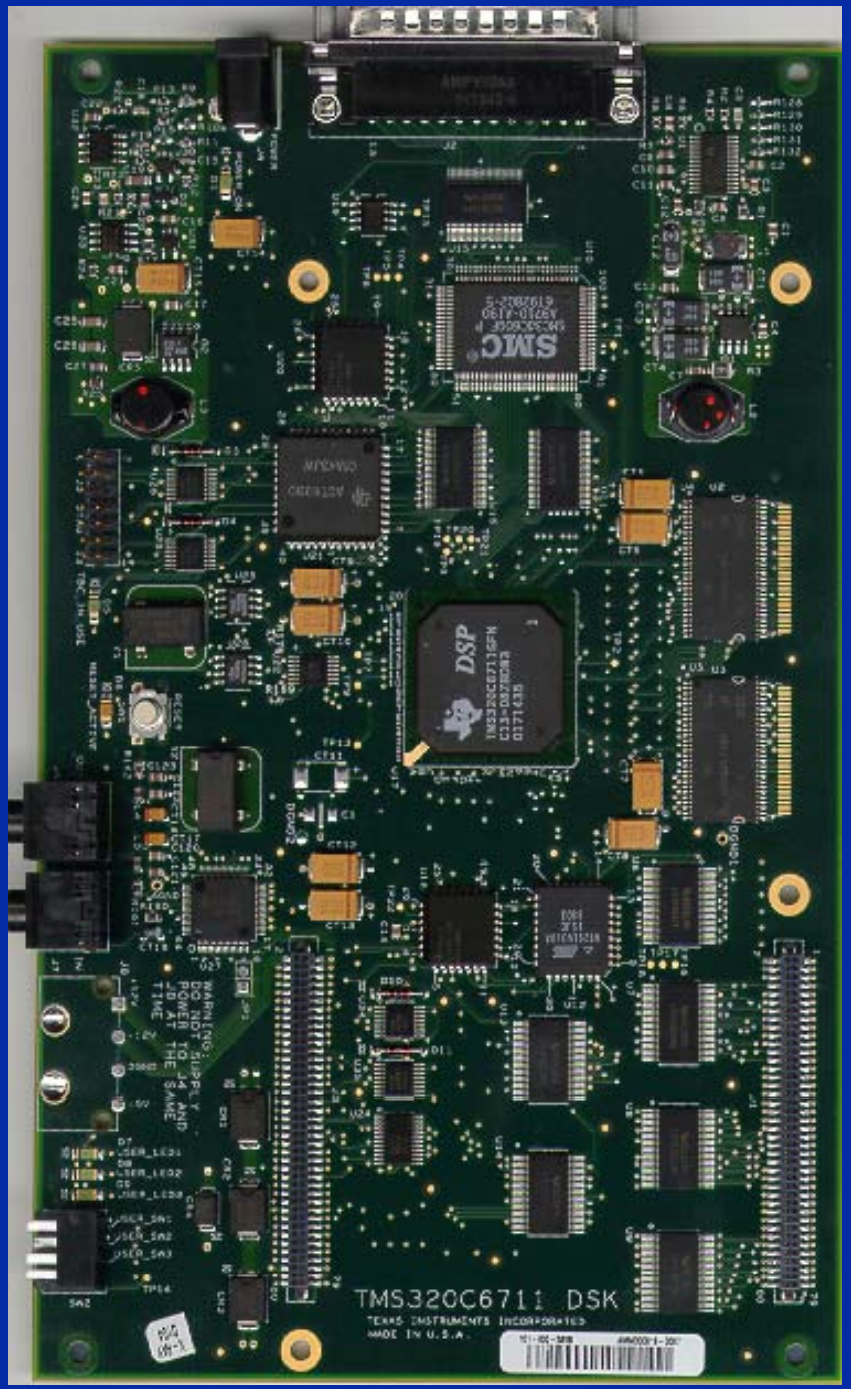
- 150 MHz ‘C6711 DSP
- TI 16-bit A/D Converter (‘AD535)
- External Memory
  - 16M Bytes SDRAM
  - 128K Bytes Flash ROM
- LED’s
- Daughter card expansion
- Power Supply & Parallel Port Cable

Software

- Code Generation Tools

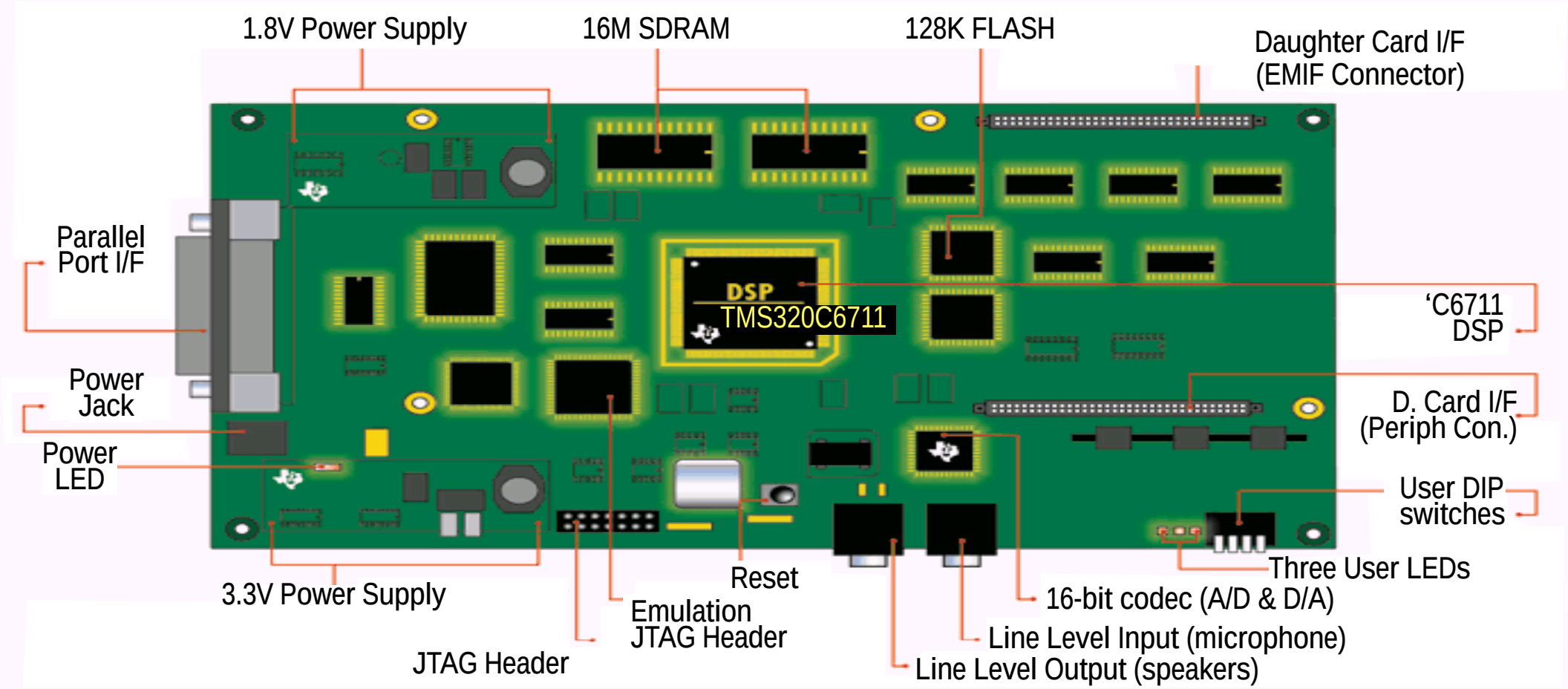
  (C Compiler, Assembler & Linker)
- Code Composer Debugger

  (256K program limitation)
- Example Programs & S/W Utilities
  - Power-on Self Test
  - Flash Utility Program
  - Board Confidence Test
  - Host access via DLL
  - Sample Program(s)



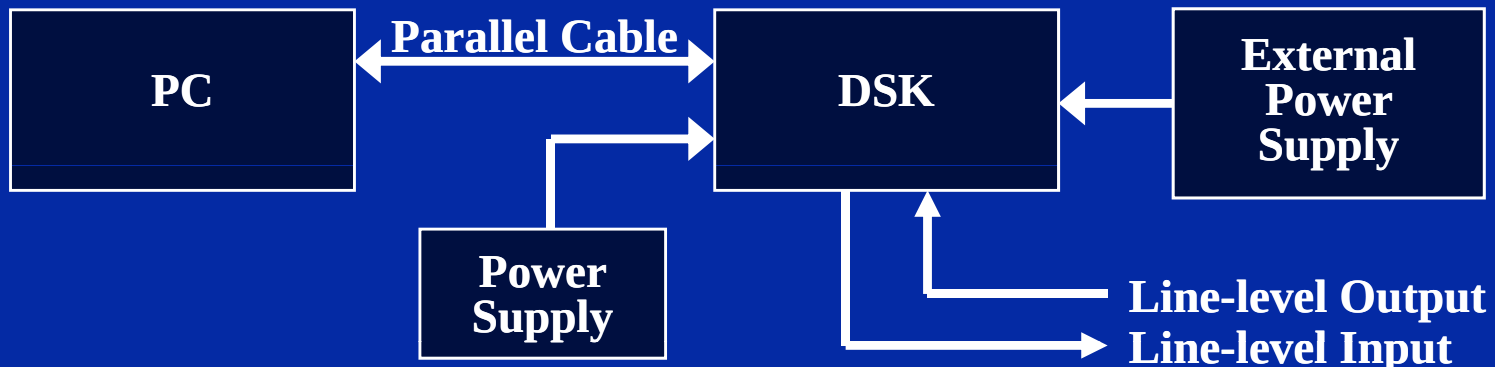
Hardware: (1) Overview

- **The daughter card interface socket provides a method for accessing most of the C6711 DSP for hardware extension.**

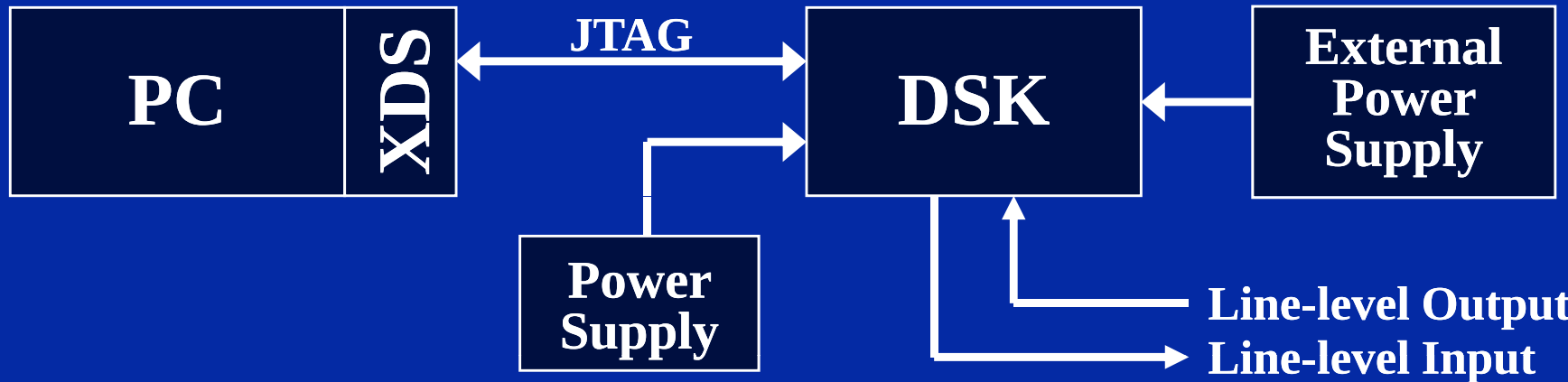


Hardware: (2) DSK Connections

(A) Parallel port: The PC's parallel port is connected to the parallel port on the DSK.



(B) JTAG: An XDS JTAG emulator connected to the PC (either internal or external) is connected to the JTAG header on the DSK.



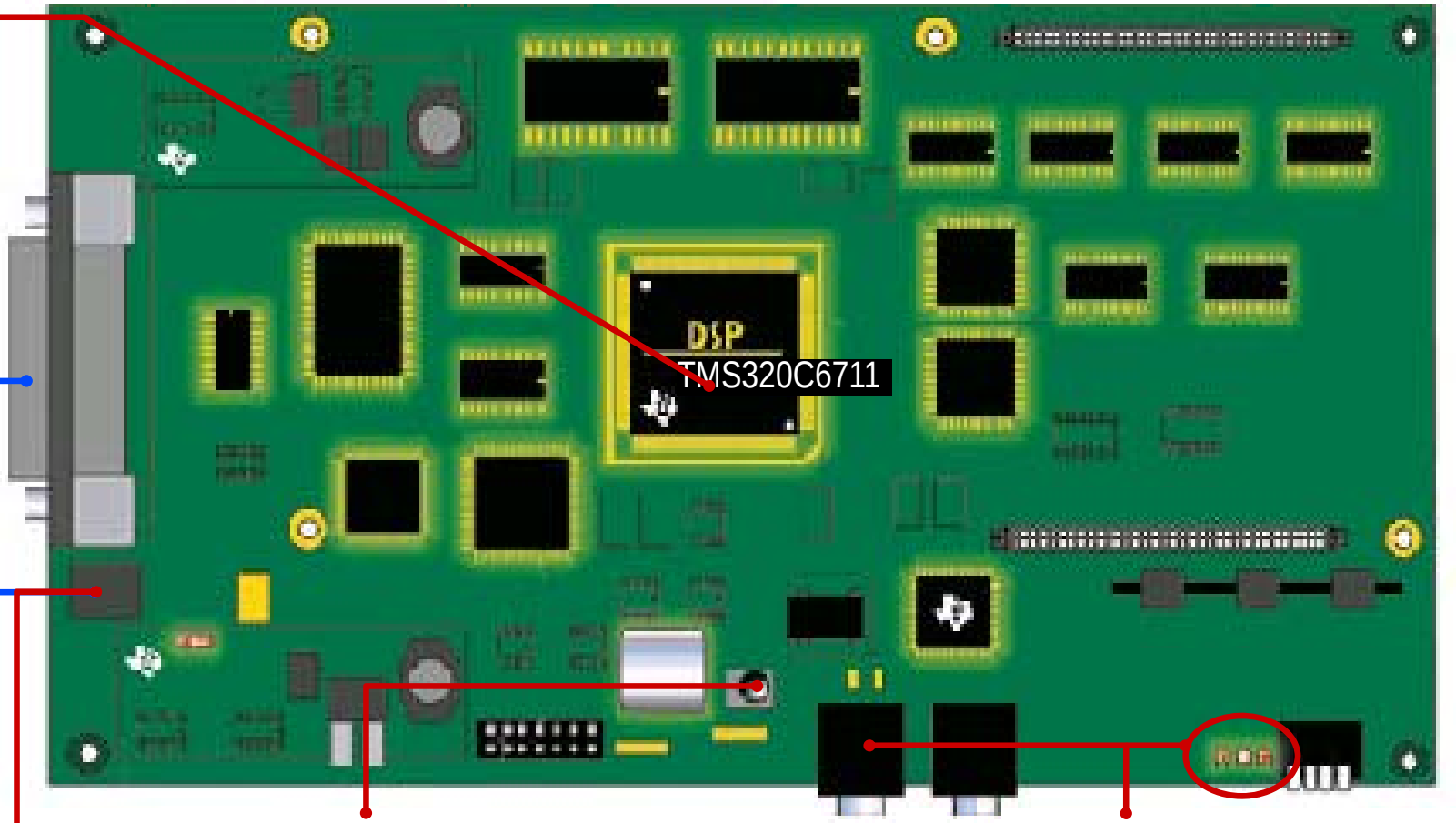
Hardware: (3) Power On Self Test (POST)

- There are three LEDs to provide the user with feedback from the test procedure.
- The test program (stored in the FLASH memory, code available on the DSK CD-ROM) runs every time DSK is powered on and reset.

Test	LED 3	LED 2	LED 1	Description
0	0	0	0	Start state
1	0	0	1	DSP internal SRAM test
2	0	1	0	External SDRAM test
3	0	1	1	DSP McBSP0 loop back test
4	1	0	0	External codec read/write test
5	1	0	1	External codec tone generation test
6	1	1	0	External LED and DSP timer test
7	1	1	1	Unused – available for future test use
	B L I N K	A L L		All tests completed successfully

Hardware: (4) Resets

CCS Reset
- Menu selection: Debug→DSP Reset
- Resets 'C6711 DSP
- Causes bootload from FLASH which overwrites internal memory



Absolute Reset

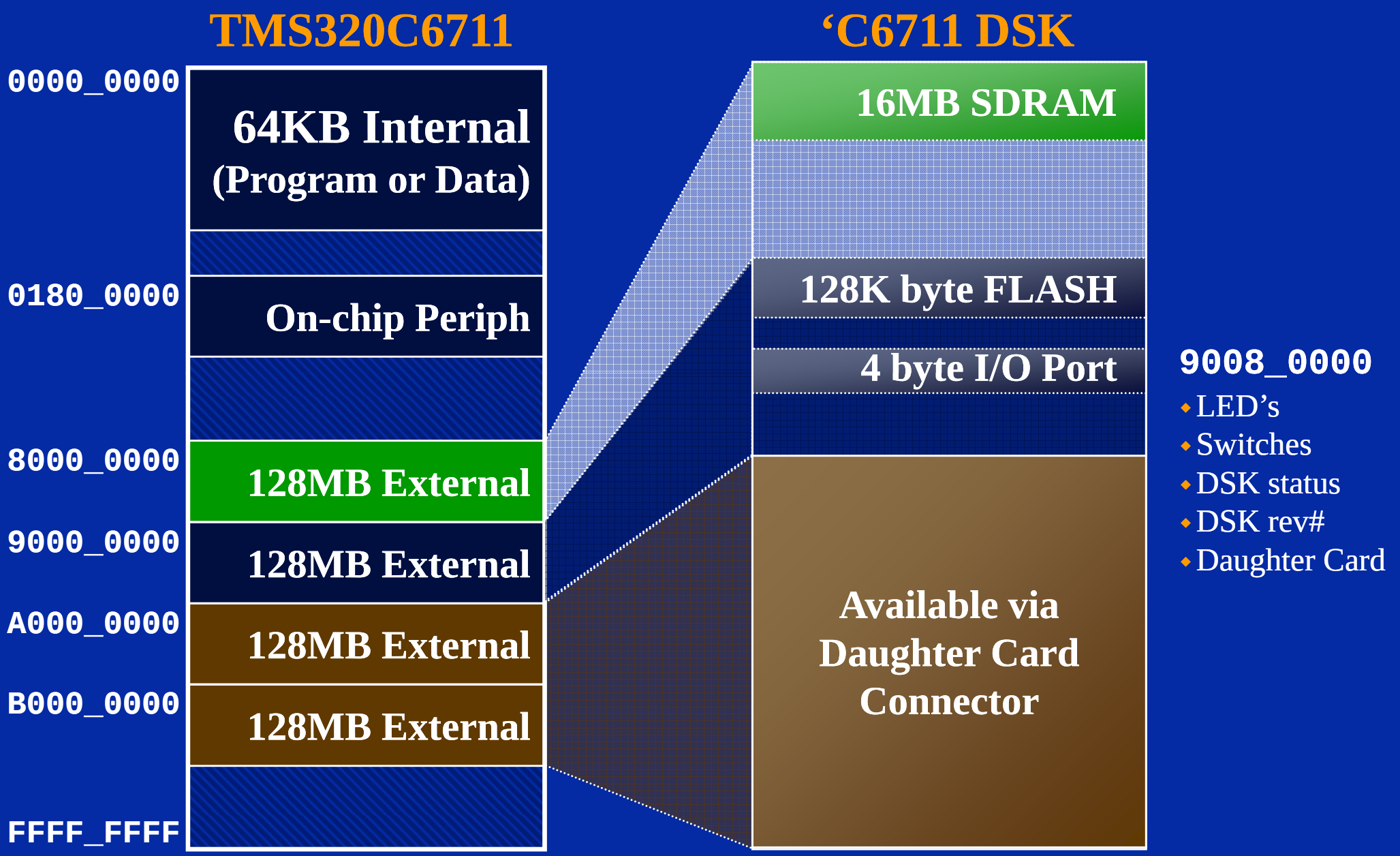
On rare occasions you might have to:
- Pull power jack
- Pull parallel port

Apply Power
- POST runs

Reset Pushbutton
- Don't push when CCS is running
- Does not force FULL system reset
- To fully reset board, pull power plug

Power On Self Test (POST)
- Counts 1 - 7
- 4: ○○● mic input → spkr out
- 5: ●○● sinewave → spkr out
- Don't start CCS until end (all 3 LEDs flash at the end)
- If switches are set to 0100, a fast version of POST is run

Hardware: (5) Memory Maps



Hardware: (5) Memory Maps

Description	Origin	Length
Internal RAM (L2) mem	0x00000000	0x00010000
EMIF control regs	0x01800000	0x00000024
Cache configuration reg	0x01840000	0x00000004
L2 base addr & count regs	0x01844000	0x00000020
L1 base addr & count regs	0x01844020	0x00000020
L2 flush & clean regs	0x01845000	0x00000008
CE0 mem attribute regs	0x01848200	0x00000010
CE1 mem attribute regs	0x01848240	0x00000010
CE2 mem attribute regs	0x01848280	0x00000010
CE3 mem attribute regs	0x018482c0	0x00000010
HPI control reg	0x01880000	0x00000004
McBSP0 regs	0x018c0000	0x00000028
McBSP1 regs	0x01900000	0x00000028
Timer0 regs	0x01940000	0x0000000c
Timer1 regs	0x01980000	0x0000000c
Interrupt selector regs	0x019c0000	0x0000000c
EDMA parameter RAM	0x01a00000	0x00000800
EDMA control regs	0x01a0ffe0	0x00000020
QDMA regs	0x02000000	0x00000014
QDMA pseudo-regs	0x02000020	0x00000014
McBSP0 data	0x30000000	0x04000000
McBSP1 data	0x34000000	0x04000000
CE0, SDRAM, 16 MBytes	0x80000000	0x01000000
CE1, 8-bit ROM, 128 Kbytes	0x90000000	0x00020000
CE1, 8-bit I/O port	0x90080000	0x00000004
CE2 - Daughtercard	0xA0000000	0x10000000
CE3 - Daughtercard	0xB0000000	0x10000000

Software: (1) PC Host Utilities

DSK Loader

`dsk6ldr.exe` *`filename.out`*

- Runs on PC host
- Downloads .out file to DSK memory map
- Stand alone DSK loader for when you want to bypass CCS

FLASH Programming

`hex6x.exe` *`f.out h.cmd`*

`flash.exe` *`f.hex`*

- First, convert *file.out* to *file.hex*
- The flash utility downloads the hex file into the on-DSK FLASH
- Both programs run on the PC host. Links\SPRA804.pdf

DSK Confidence Test

`dsk6xtst`

- Run from MSDOS prompt
- Command-line utility tests proper installation of the DSK board
- Additionally, it tests: Internal SRAM, SDRAM, FLASH, McBSP, Timers, EDMA, LEDs and Audio codec

Software: (2) CCS DSK Utilities

Confidence Test

1. Unload any gel files already loaded in CCS
2. Load `conftest.gel`
3. Run confidence tests from GEL menu

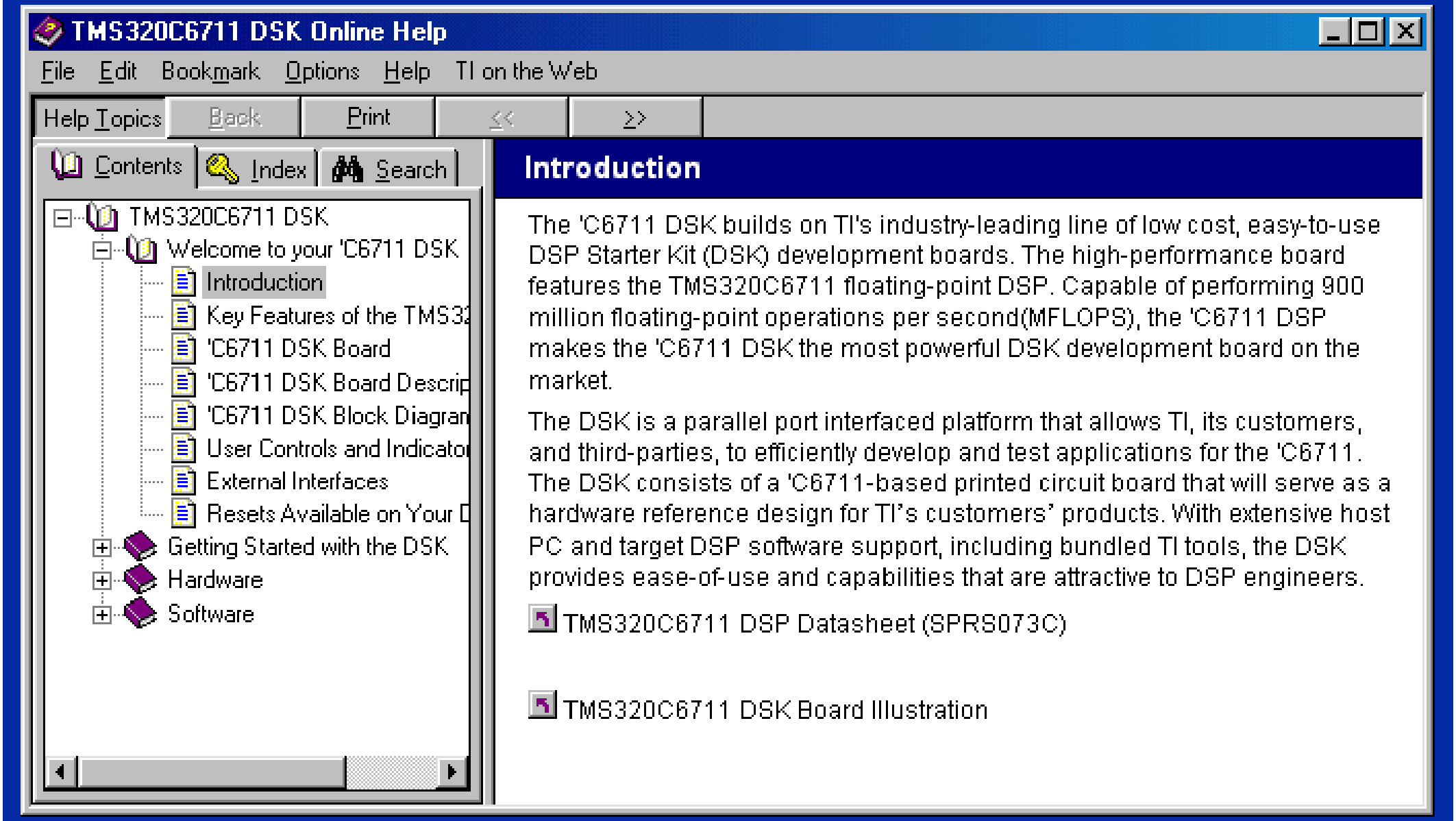
General Extension Language (GEL):

An interpretive language that enables you to write functions to configure the IDE and access the target processor.

Quick Test

- Run from CCS GEL menu
- Defined in `dsk6xinit.gel`
- Non-intrusive test by reading and writing:
  - LEDs
  - Switches
  - DSK board revision
- Outputs switch values

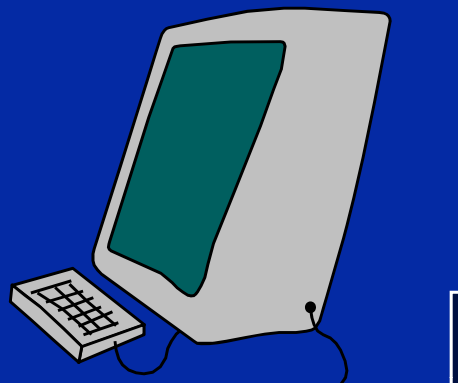
Software: (3) DSK Help

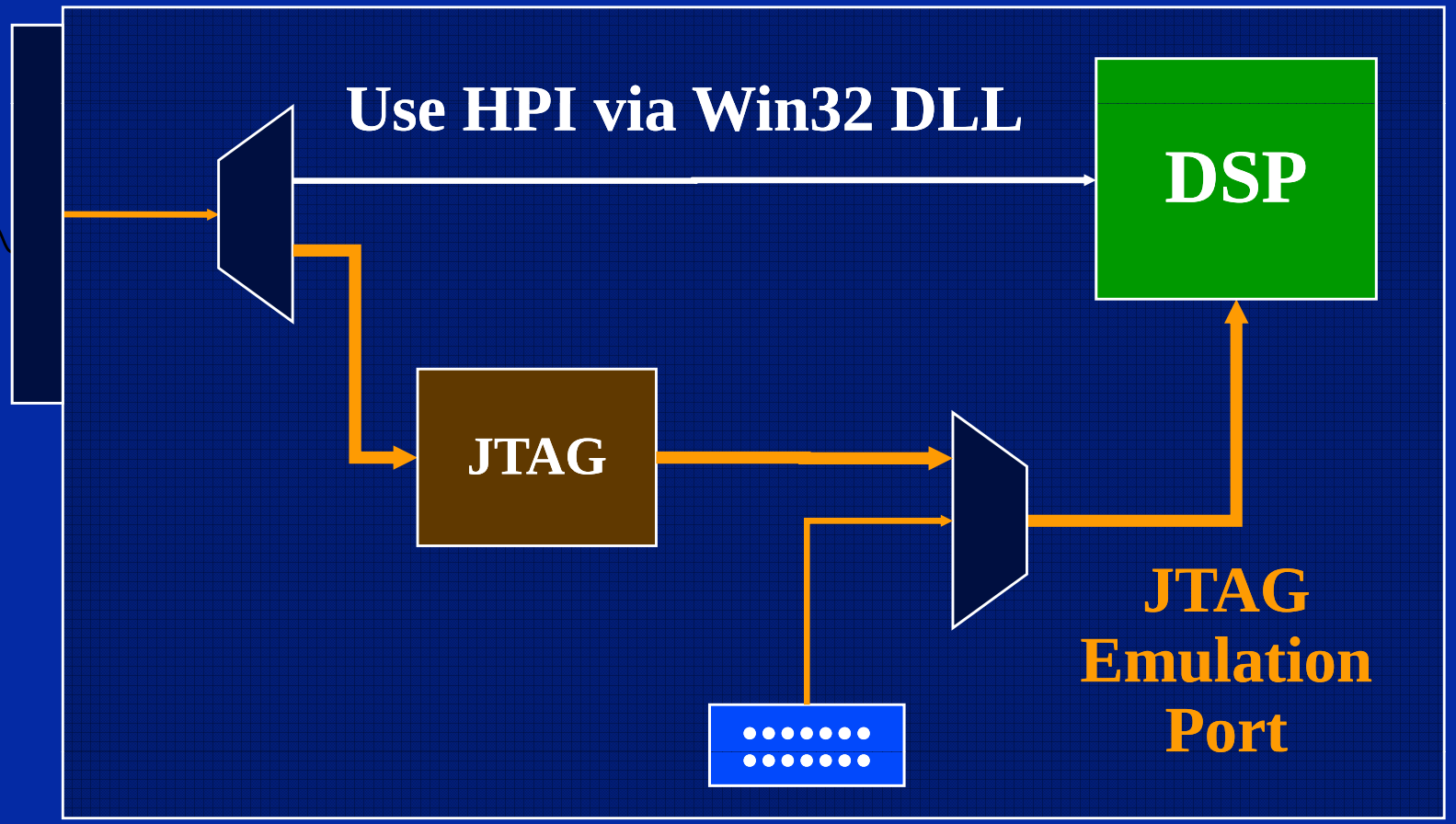


- **DSK6711 help is available via the Help menu in CCS.**

Software: (4) PC → DSK Communications

- CCS uses parallel port to control DSP via JTAG port
- You can use full TI eXtended Dev System (XDS) via 14 pin header connector
- Communicate from Windows program (C++, VB) via parallel port using Win32 DLL





Note: You should not use the parallel port for simultaneous emulation and HPI connection.

Software: (4) PC → DSK Communications

- **Win32 API functions for Host to DSK communications:**

dsk6x_open( )	Open a connection to the DSK
dsk6x_close( )	Close a connection to the DSK
dsk6x_reset_board( )	Reset the entire DSK board
dsk6x_reset_dsp( )	Reset only the DSP on the DSK
dsk6x_coff_load( )	Load a COFF image to DSP memory
dsk6x_hpi_open( )	Open the HPI for the DSP
dsk6x_hpi_close( )	Close the HPI for the DSP
dsk6x_hpi_read( )	Read DSP memory via the HPI
dsk6x_hpi_write( )	Write to DSP memory via the HPI
dsk6x_generate_int( )	Generate a DSP interrupt

Laboratory Exercise: DSK Hardware Setup

(1) Connect the following cables:

- Parallel port.
- Audio cables.

(2) Connect the power and observe the Power On Self-Test (POST) (Refer to Slide 15).

(3) Run the confidence test for the PC. There are three methods to run this test:

(a) Run the “dsk6xtst.exe” by double-clicking on the file in “\Code\Chapter 03 - CCS and DSK\”.

(b) Run the “dsk6xtst.exe” file from windows by double clicking on the file in “\ti\c6000\dsk\conftest\Host\Debug”.

(c) Open a DOS window and run the “dsk6xtst.exe” file located in the directory above.

Laboratory exercise: DSK hardware setup

Notes:

- **The SDRAM may take a while due to the large amount of SDRAM on the ‘C6711 DSK.**
- **The CODEC test performs two operations: (1) a 1kHz tone output, and (2) an audio input to output loopback. You must have a speaker connected to the the output jack to hear the test.**
- **If the confidence test fails:**
  - **(1) Remove the power and parallel cable from the DSK.**
  - **(2) Reset your PC.**
  - **(3) Reconnect the power and the parallel cable.**
  - **(4) Invoke CCS.**

Laboratory Exercise: Using CCS

Implement: $$y = \sum_{i=0}^{N-1} a_i x_i$$

with: $a_i = \{40, 39, \ldots, 1\}$

$x_i = \{1, 2, \ldots, 40\}$

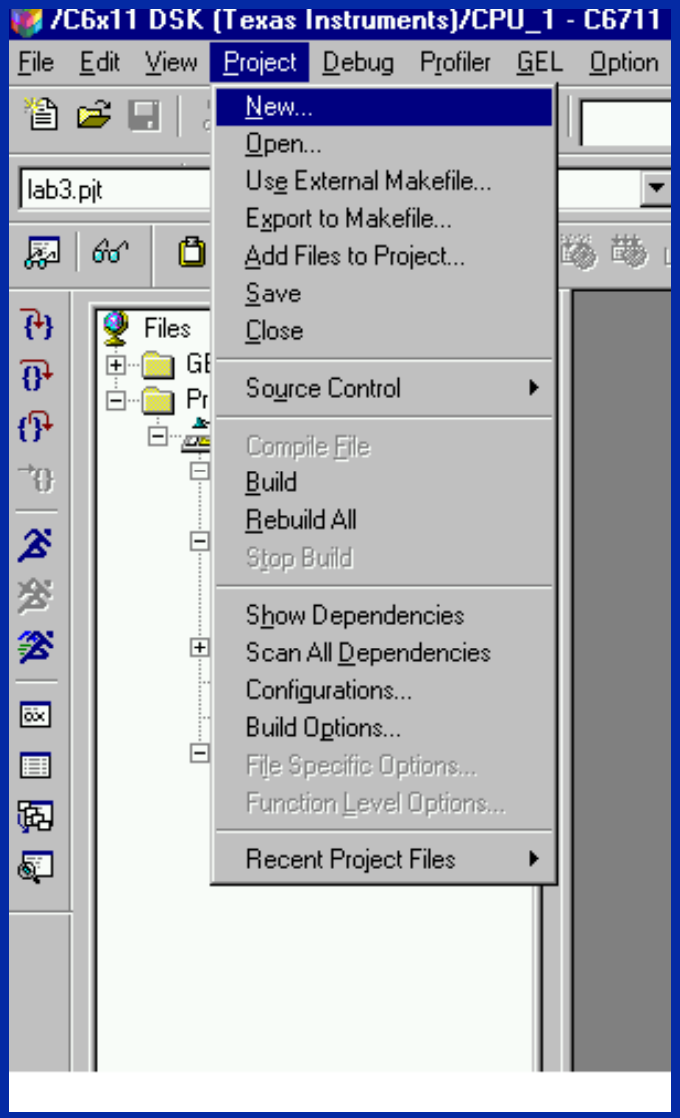
(1) Create a working directory and copy the following files from \Code\Chapter 03 - CCS and DSK\:

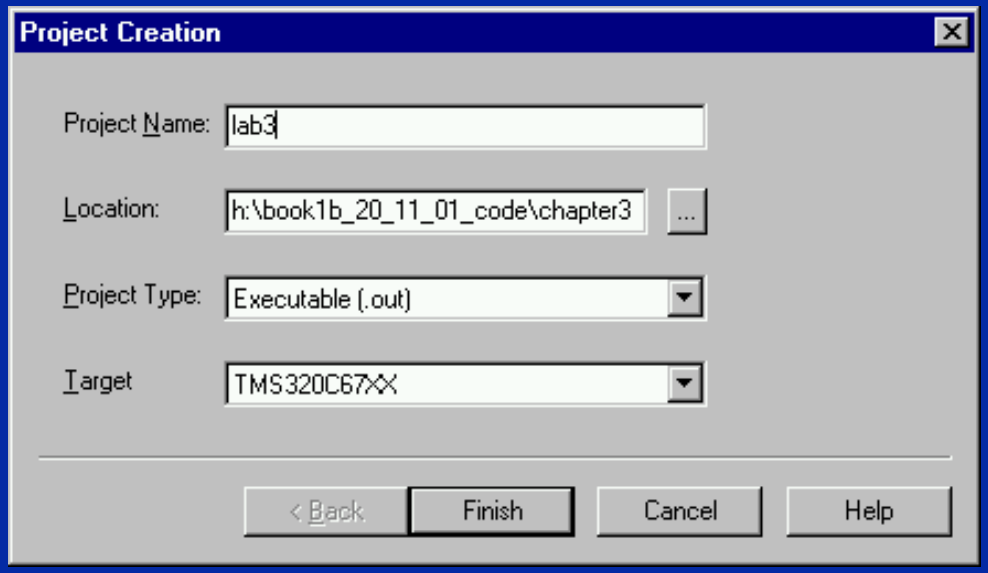
- (a) lab3.cdb
- (b) lab3.c
- (c) lab3cfg.cmd

(2) Create a new project:

- (a) Start CCS.
- (b) Create a new project as shown on the following slide.

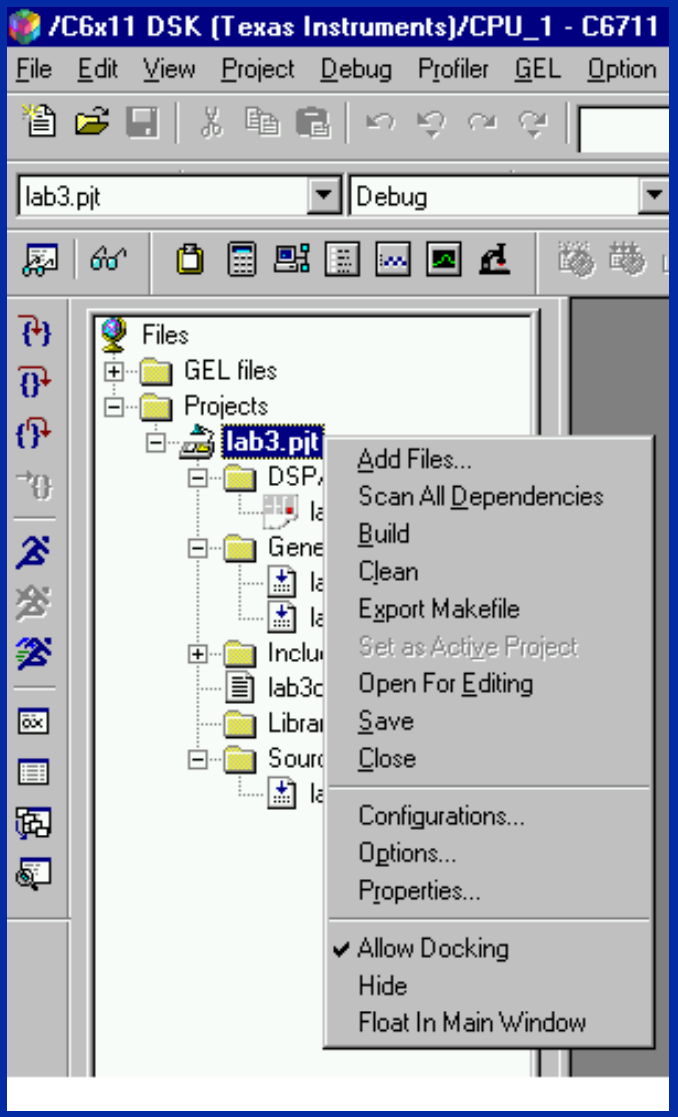
Laboratory Exercise: Using CCS





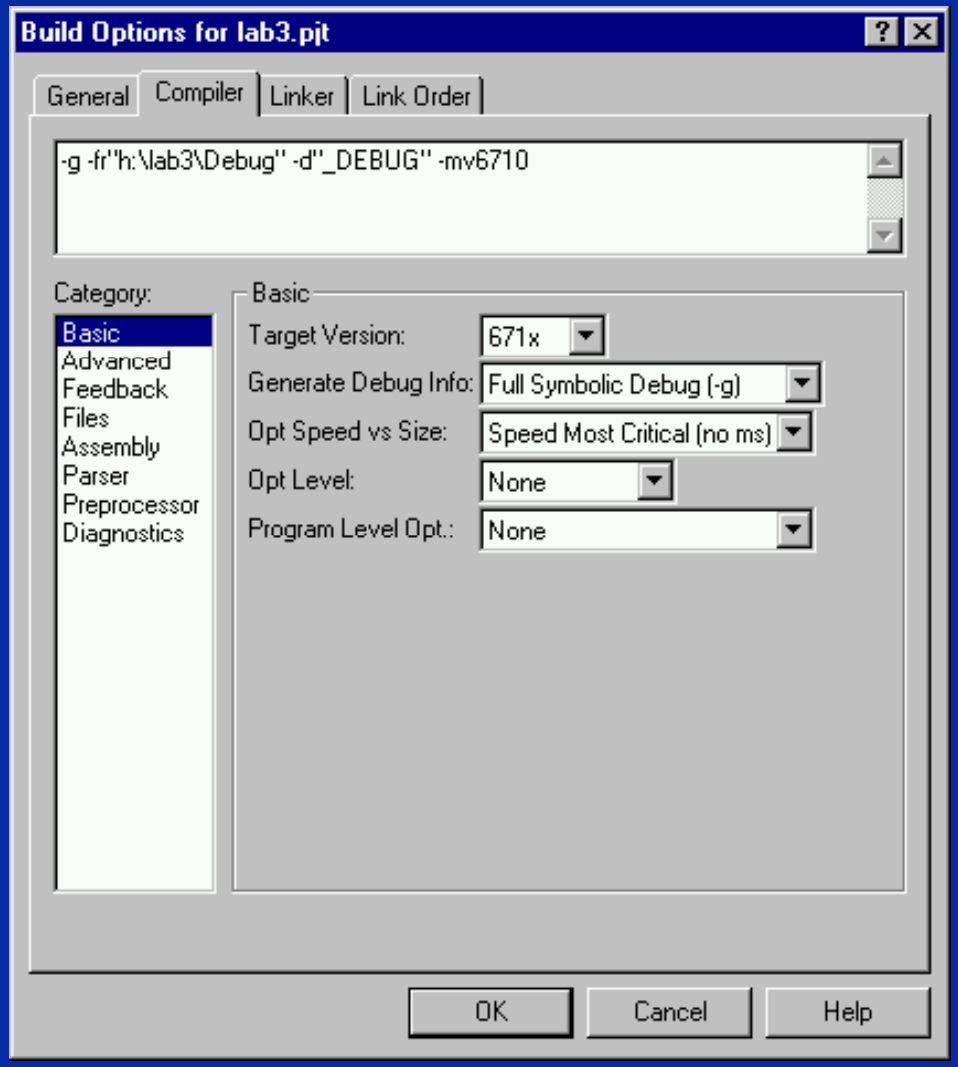
Laboratory Exercise: Using CCS

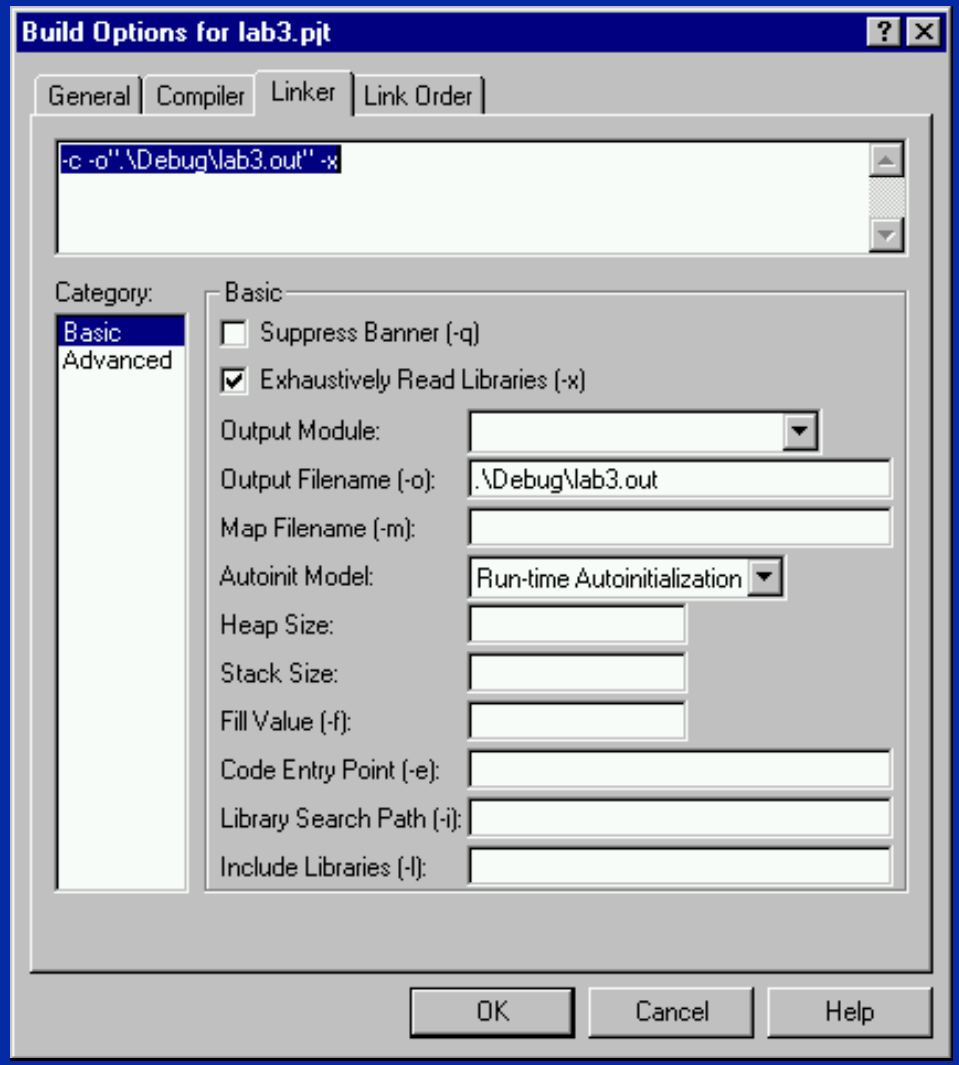
(3) Add files to the project (lab3.c, lab3.cdb, lab3cfg.cmd).



Laboratory Exercise: Using CCS

(4) Change the build options (compile and link):





Laboratory Exercise: Using CCS

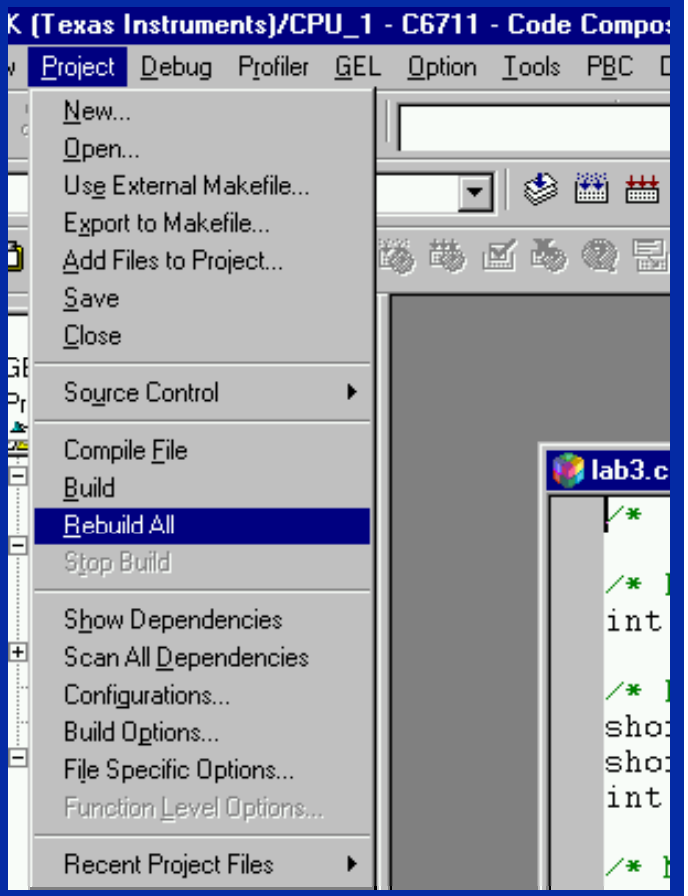
(5) Build the output program (lab3.out):

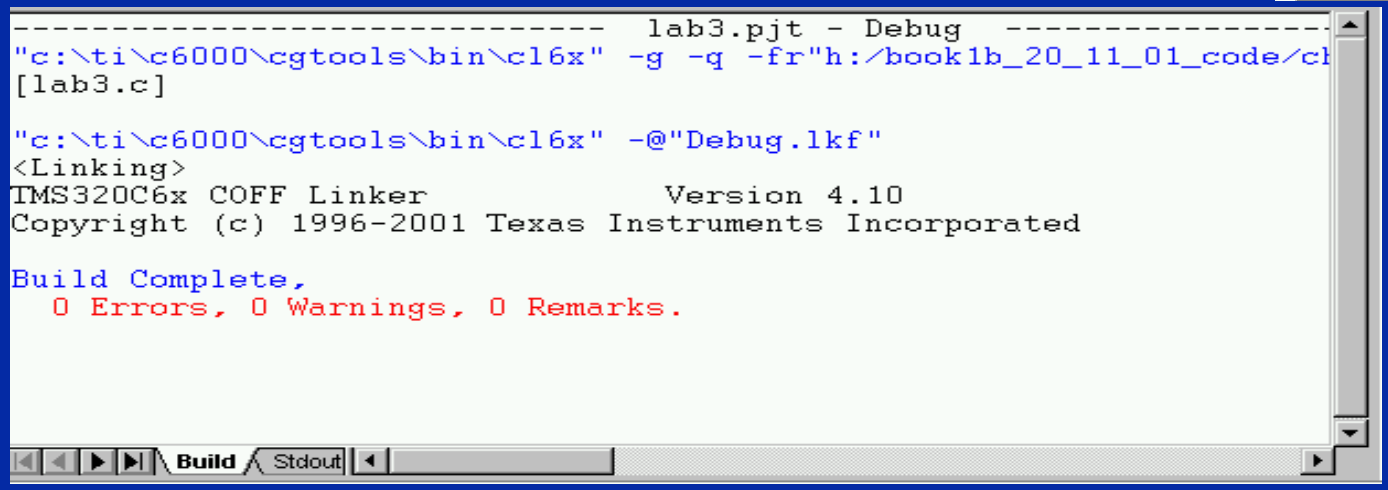
(a) Build the project by:

(i) Clicking the Rebuild All toolbar icon.

(ii) Selecting Rebuild All in the project menu.

(b) Verify that the build output window is complete with "0 errors, 0 warnings":

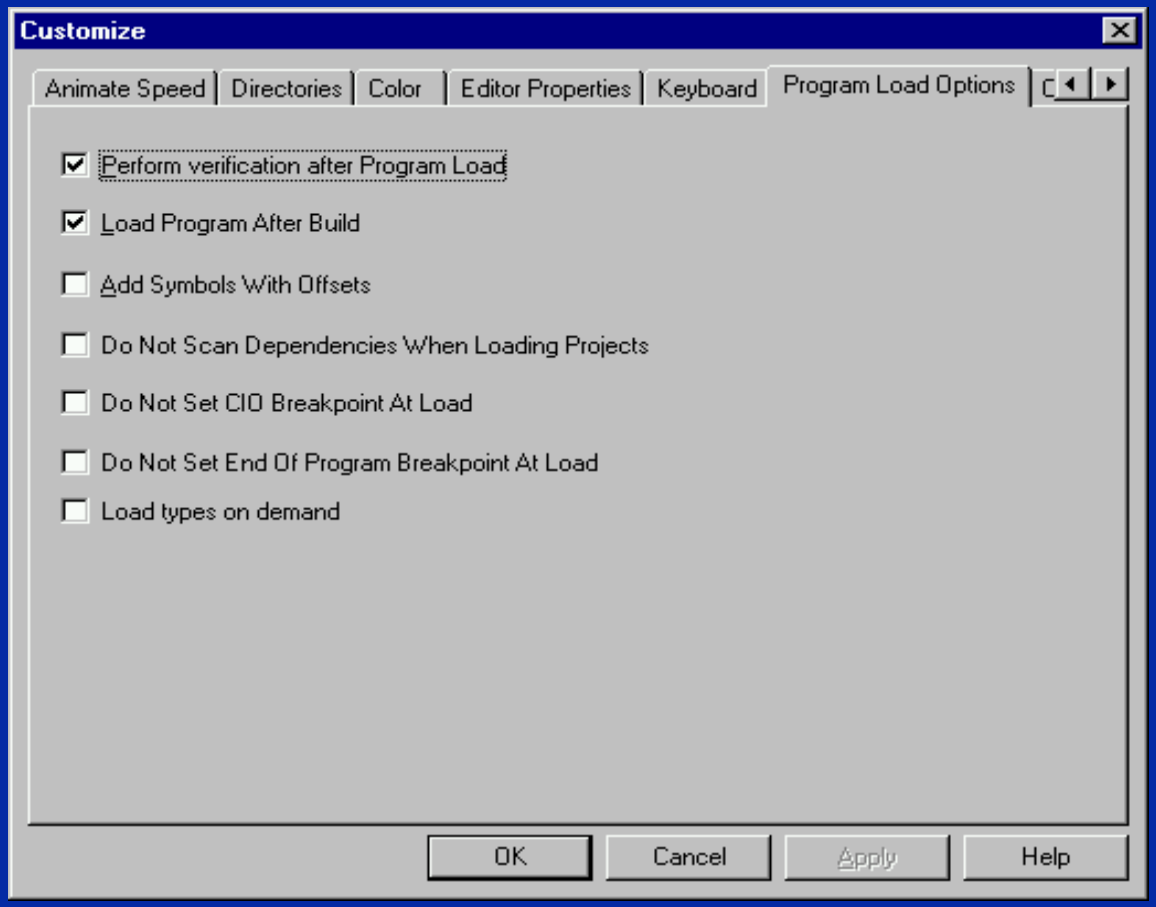




Laboratory Exercise: Using CCS

(6) Load the output file lab3.out into DSP memory:

(a) The program will be automatically loaded after each project build if the “Program Load after Build” option is selected as shown below:



Laboratory Exercise: Using CCS

(6) Load the output file lab3.out into DSP memory:

(b) Load the lab3.out by selecting File:Load Program as shown below:

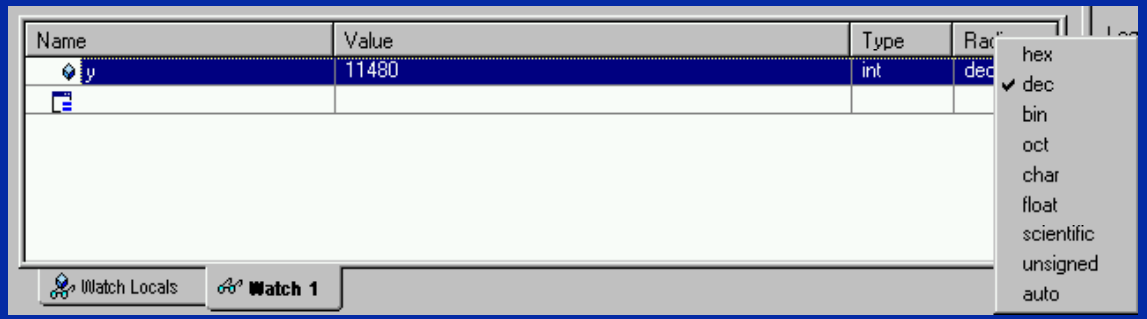
Laboratory Exercise: Using CCS

(7) Debug and run code:

(a) Go to the beginning of the program, that is main() by selecting Debug:Go Main.

(b) Watch variables:

(i) Select the variable (to be watched) from the lab3.c file, right click and select “Add To Watch Window”. If the variable is y for instance, the following window will be shown.

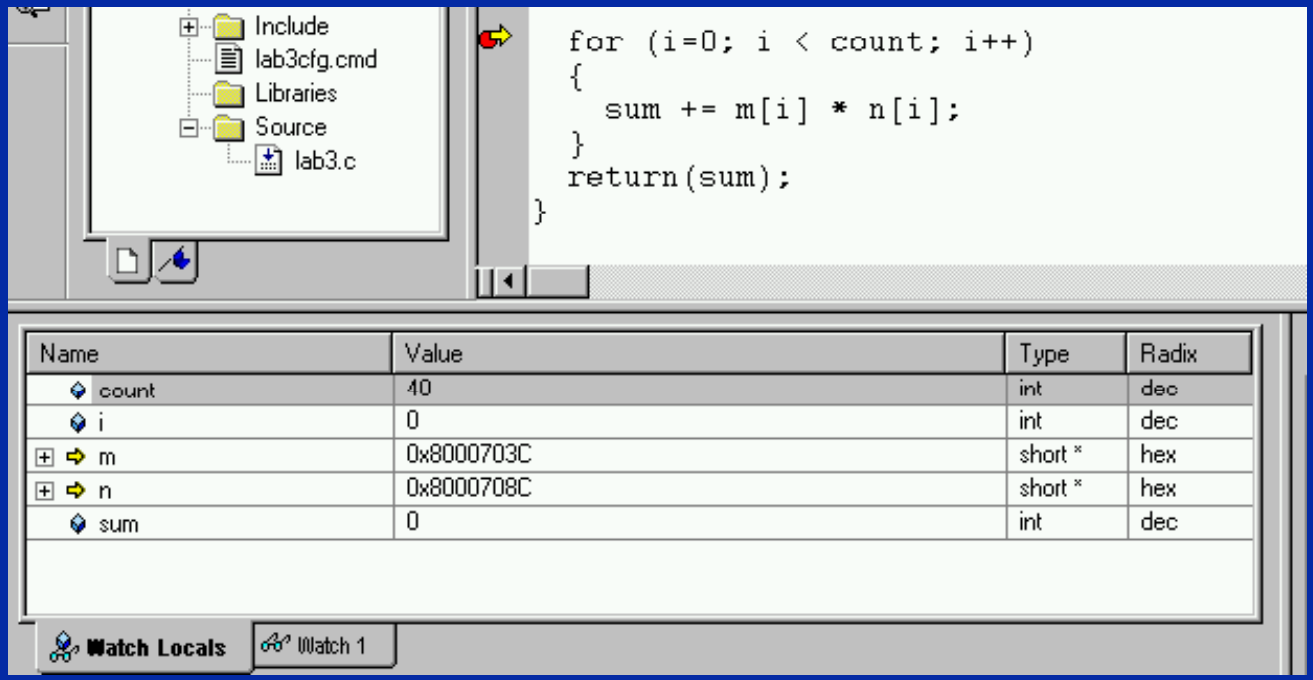


(ii) To add another variable to the watch select it and then drag and drop it on to the window.

Laboratory Exercise: Using CCS

(7) Debug and run code:

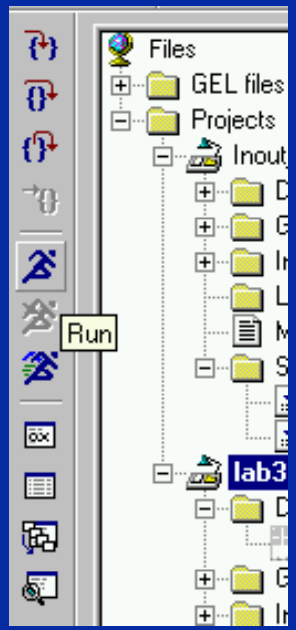
(c) CCS will automatically add the local variables:

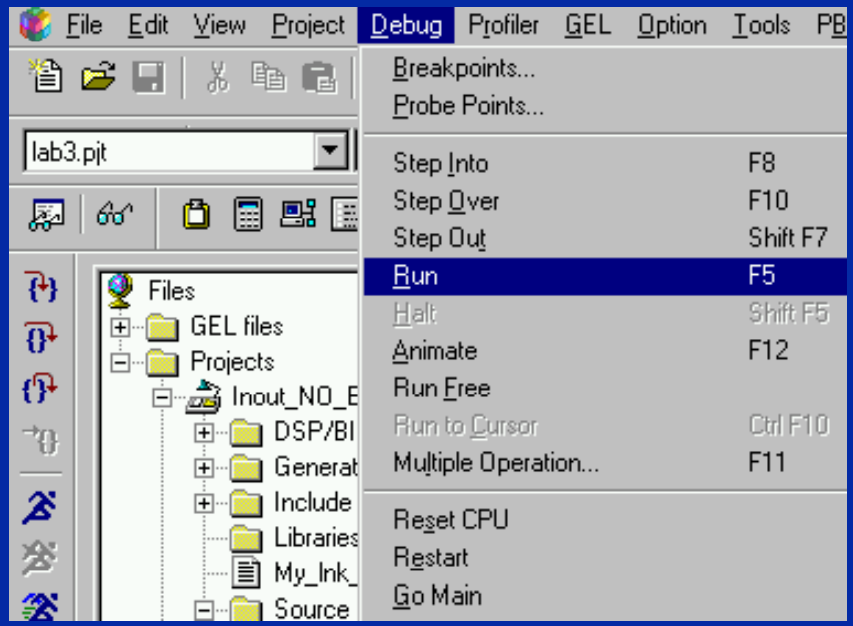


Laboratory Exercise: Using CCS

(7) Debug and run code:

(d) You can run or step through the code by using the various icons on the toolbar or use the Debug menu:





Laboratory Exercise: Using CCS

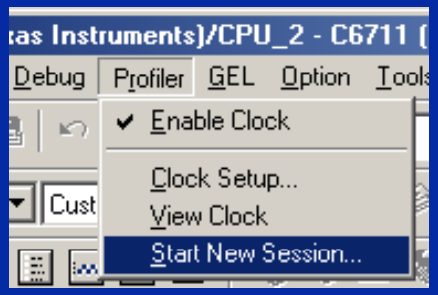
(e) Stop the processor from running and watch the variable y:

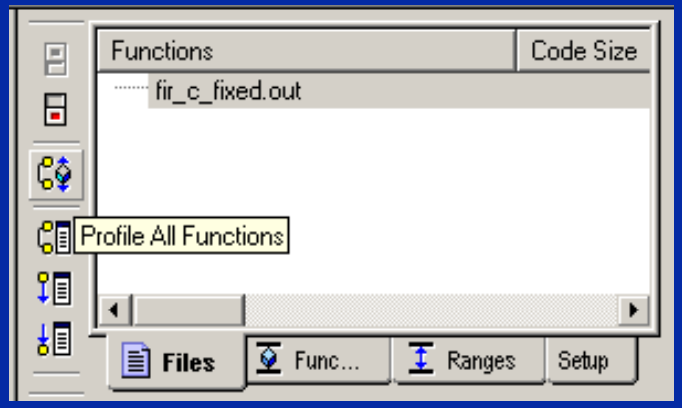
y = 0x2cdb or 11480

(8) Benchmarking and profiling code:

(a) Stop the processor, reload the code or select Debug:Restart then select Debug:Go Main.

(b) Open a new profiling session and name it "Session 1" and select "Profile All Functions" by clicking the following toolbar button:





Laboratory Exercise: Using CCS

(8) Benchmarking and profiling code:

(c) Expand the lab3.c as shown below:

```
int dotp(short *m, short *n, int count);

/* Declarations                                          */
short a[40] = {40,39,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,
short x[40] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
int y = 0;

/* Main Code */
main()
{
  y = dotp(a, x, 40);

  for(;;);
}

int dotp(short *m, short *n, int count)
{ int i;
  int sum = 0;

  for (i=0; i < count; i++)
  {
    sum += m[i] * n[i];
  }
  return(sum);
```

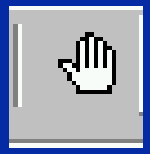
Laboratory Exercise: Using CCS

(8) Benchmarking and profiling code:

(d) Add a breakpoint at “for(;;);”. This can be done by:

(i) Click the cursor on the highlighted line below.

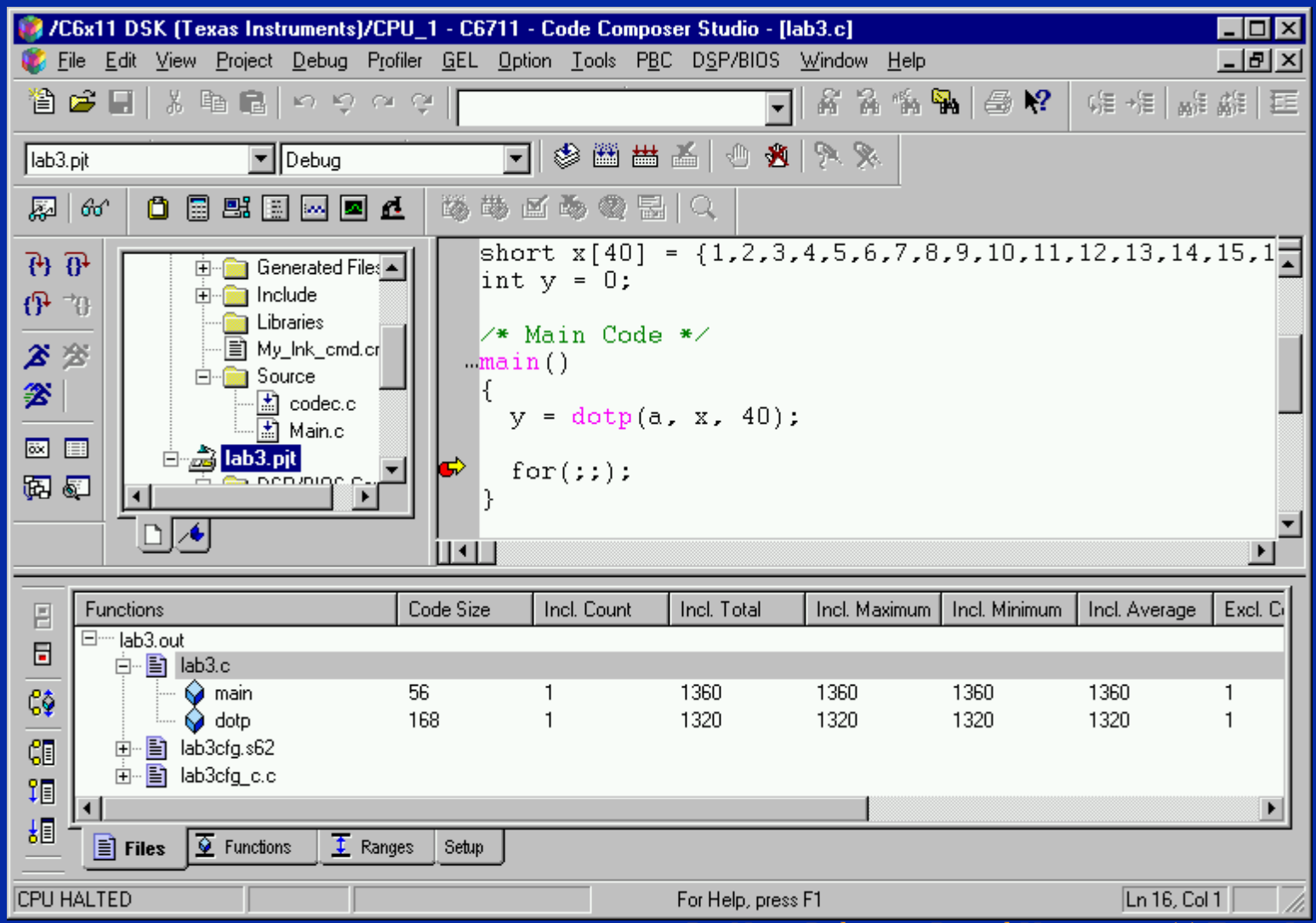
(ii) Click the “Add Breakpoint” toolbar button:



Laboratory Exercise: Using CCS

(8) Benchmarking and profiling code:

(e) Run the program and examine the profile window:



CCS and DSK

- **CCS Overview:**
  - **\Links\spru301c.pdf**
- **Resets:**
  - **\Links\ccs_dsk.pdf**

Chapter 3
Code Composer Studio and the DSK
- End -