F28335 ControlCard Lab1

Toggle LED LD2 (GPIO31) and LD3 (GPIO34)

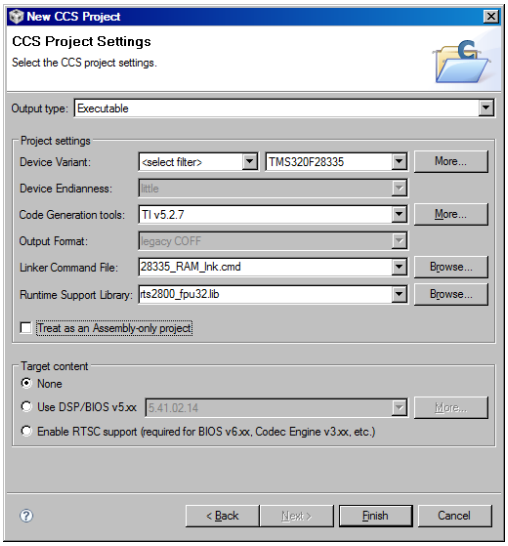
1. Project Dependencies

The project expects the following support files:

- Support files of controlSUITE installed in:

  C:\TI\controlSUITE\device_support\f2833x\v132

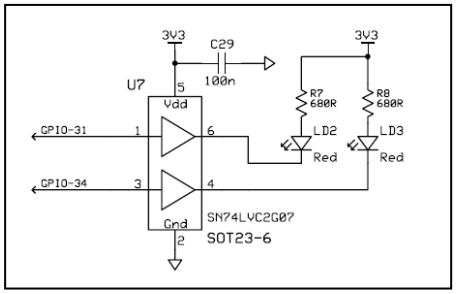
- Code Generation Tools Version 5.2.7:



- CodeComposerStudio Version 4.1.3:

2. Project Objective

- Blink LD2 (GPIO31) and LD3 (GPIO34) at the F28335 ControlCard.

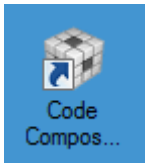


- Software delay loop in "main()" for approximately 100 milliseconds.
- No interrupt system in use.
- Watchdog Timer disabled.

3. Hardware Setup

No special setup required. Connect the F28335 Docking Station with a USB cable to your computer. Make sure the main switch SW1 is in position "USB". The LED LD1 (green) of the F28335 controlCARD should be on.

4. Start Code Composer Studio V4

Start Code Composer Studio V4: 

When CCS loads, a dialog box will prompt you for the location of a workspace folder. Use the default location for the workspace and click OK. This folder contains all CCS custom settings, which includes project settings and views when CCS is closed so that the same projects and settings will be available when CCS is opened again. The workspace is saved automatically when CCS is closed.

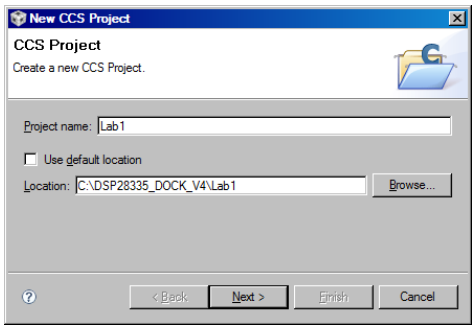
Note: The first time CCS opens a "Welcome to Code Composer Studio v4" page appears. Close the page by clicking on the CCS icon in the upper right or by clicking the X on the "Welcome" tab. You should now have an empty workbench.

The term workbench refers to the desktop development environment. The workbench will open in the "C/C++ Perspective" view. Notice the C/C++ icon in the upper right-hand corner. A perspective defines the initial layout views of the workbench windows, toolbars, and menus which are appropriate for a specific type of task (i.e. code development or debugging). The "C/C++ Perspective" is used to create or build C/C++ projects. A "Debug Perspective" view will automatically be enabled when the debug session is started.

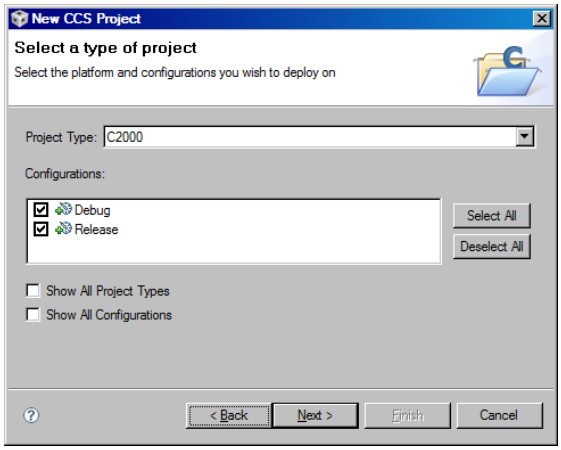
5. Project Design

5.1. Create a new project

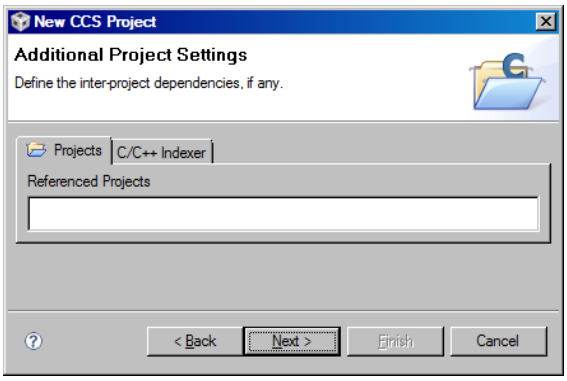
➔ **File ➔ New ➔ CCS Project:**



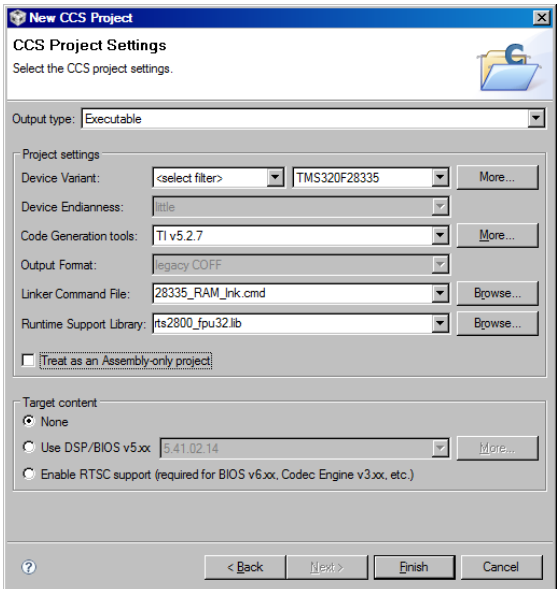
In the following window, select “C2000” for the project type.



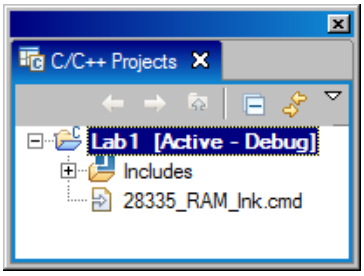
In the next window, do not add any additional settings:



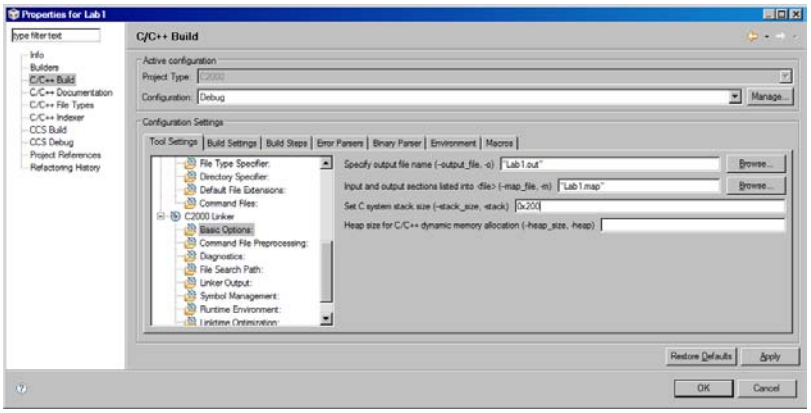
Fill in the project properties according to the following picture:



Click the button“Finish”.
Next, define the size of the C system stack. In the project window, right click at “Lab1” and select “Properties”:



In category “C/C++ Build”, “C2000 Linker”, “Basic Options” set the C system stack size to 0x200:

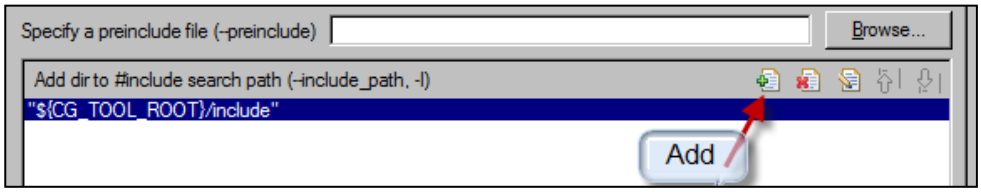


Note: The stack memory is used by the compiler to store local variables, parameters and the processors context in case of hardware interrupts. It is our task to specify a certain amount of memory for this purpose and 0x200 is sufficient for this lab.

We also have to extent the search path of the C-Compiler for include files. Right click at project "Lab1" and select "Properties". Select "C/C++ Build", "C2000 Compiler", "Include Options". In the box "Add dir to #include search path", add the following lines:

C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_common\include
C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_headers\include

Note: Use the "Add" Icon to add the new path:



In the "C2000 Compiler" tab, category "Diagnostic Options", enable

- ✓ Emit Diagnostic Identifier Numbers
- ✓ Issue Remarks
- ✓ Verbose Diagnostics

Also, enter the following numbers:

- Suppress Diagnostic: 238
- Treat Diagnostic as Error: 225;515

Background rationale:

Diagnostic 238 "Controlling Expression is constant" will be issued in case of a loop construction, such as "while(1)" – which is used in embedded systems for an endless loop in main.

Diagnostic 225 "function declared implicitly" will be issued in case of missing function prototypes in a C – code statement.

Diagnostic 515 "a value of type %t1 cannot be assigned to an entity of type %t2" will be generated in case of false union or structure accesses. This can happen for example in memory mapped registers.

Close the Property Window by Clicking <**OK**>.

5.2. Link Files to Project

In the project window, right click at “Lab1” and select “Link Files to Project”:
First link a file, which defines the physical addresses for all data memory mapped registers:

C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_headers\cmd\DSP2833x_Headers_nonBIOS.cmd

Now link the definition for all global variables to access the F28335 hardware registers:

C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_headers\source\DSP2833x_GlobalVariableDefs.c

Link the code file that defines the code-start entry-point branch instruction. This is required to connect the hardware reset entry-point sequence to a first address in code space (section “code_start”).

C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_common\source\DSP2833x_CodeStartBranch.asm

Link 3 more files with functions to perform the basic core initialization of the F28335:

C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_common\source\DSP2833x_SysCtrl.c
C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_common\source\DSP2833x_usDelay.asm
C:\TI\controlSUITE\device_support\f2833x\v132\DSP2833x_common\source\DSP2833x_ADC_cal.asm

5.3. Create the main C - file

➔File ➔ New ➔ Source File ➔ “Lab1_1.c”

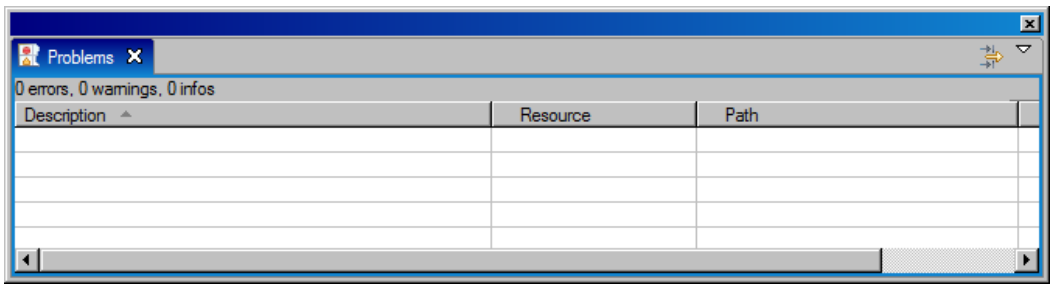
A new edit window will open. Enter the following source code:

```
#include "DSP2833x_Device.h"
void main(void)
{
}
```

5.4. Build the Project

➔Project ➔ Rebuild Active Project (Alt + Shift + P)

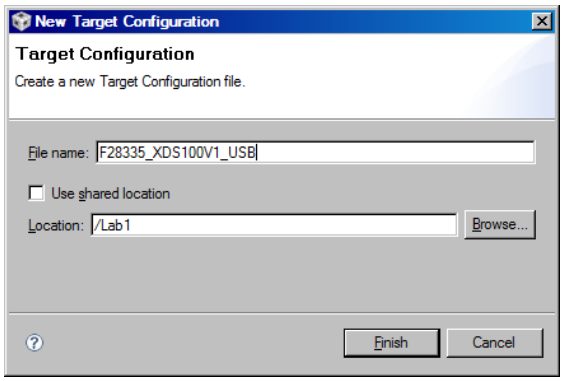
Monitor the build procedure of the tools in the output window. The final message should give you “0 Errors, 0 Warnings, 0 Remarks”.



5.5. Create a Target Configuration

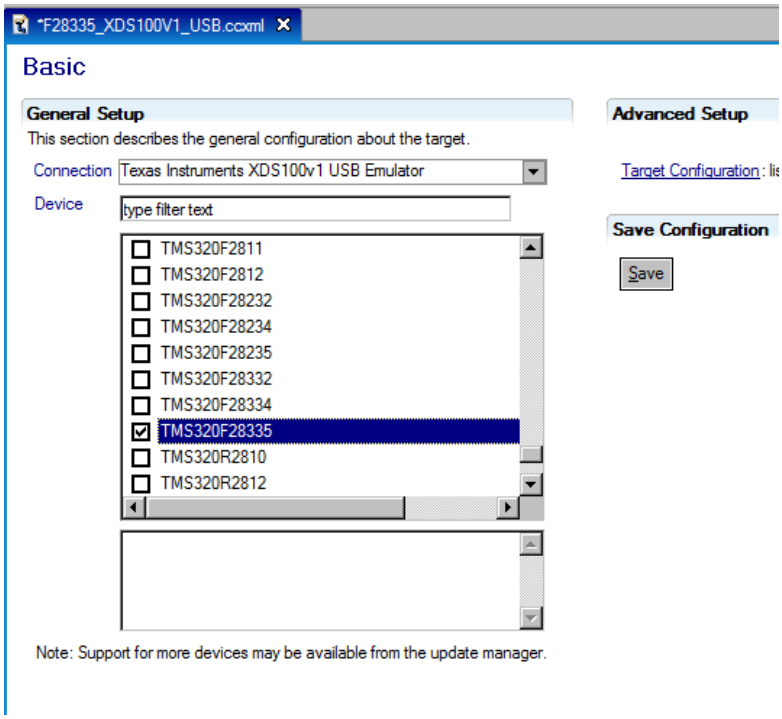
Before we can download the machine code into the F28335, we have to define the "target configuration".

➔ Target ➔ New Target Configuration



Type a name for the target configuration file in box "File name". You can use any name here but it makes sense to indicate the JTAG-emulation interface, which we will use for the download later. In case of the F28335USB – stick we use the XDS100V1, so let us call the file "F28335_XDS100V1_USB. The suffix ".ccxml" will be added automatically. Do not use the "shared location"; Instead, use the project folder "/Lab1".

In the window that appears next, select the emulator "Texas Instruments XDS100v1 USB Emulator" via the "Connection" pull-down list and select the "TMS320F28335" device checkbox:



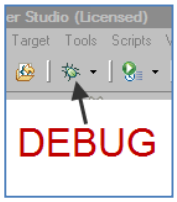
Save the Target Configuration and close the configuration window.

5.6. Load the code into the target

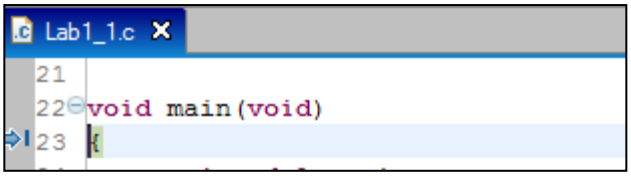
Load the machine code into the device. Click:

➔ Target ➔ Debug Active Project

Or use the “Debug” icon:



A blue arrow should now point to the end of function “main()”. This is an indication that the machine code has been downloaded properly into the F28335.



6. Code Implementation

6.1. Modify file “Lab1_1.c”

Now that we have a working project framework, we can start to develop our own code for Lab1. Recall that the objective is to toggle the LEDs LD2 (GPIO31) and LD3 (GPIO34) at the F28335 controlCARD.

- switch back to the “C/C++” perspective and edit the file “Lab1_1.c”.

At the beginning of the file, add an external prototype for function “InitSysCtrl()”. This function is defined in file “DSP2833x_SysCtrl.c” and will initialize the device to 150 MHz, Watchdog disabled and all peripheral clocks enabled.

```
extern void InitSysCtrl(void);
```

In function “main()”, add a call of function “InitSysCtrl()”:

```
InitSysCtrl();
```

Next, we have to set the direction for GPIO31 and GPIO34 to output. By default all pins are multiplexed to the GPIO – function and the direction is set to “input”. The register block “GpioCtrlRegs” is secured by the “EALLOW” secure switch. Therefore we have to open this switch, before we can modify any register:

```
EALLOW;
GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;
GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;
EDIS;
```

As initial condition (LD2 = ON, LD3 = OFF), add:

```
GpioDataRegs.GPACLEAR.bit.GPIO31 = 1;
GpioDataRegs.GPBSET.bit.GPIO34 = 1;
```

Finally we have to add an endless “while(1)” control loop to main. Inside, we will toggle the LED and install a simple software delay for-loop. At the beginning of main, add the definition for local variable ‘i’ with data type “unsigned long”:

```
while(1)
{
GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;    // toggle LD2
GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;    // toggle LD3
for (i=0; i<1000000; i++);                // time delay
}
```

6.2. Rebuild Project

➔Project ➔ Rebuild Active Project (Alt + Shift + P)

➔ Target ➔ Debug Active Project

If the compilation and the download were successful, the instruction pointer (blue arrow) is again set to the beginning of main.

Now perform a Run (F8)!

The LEDs should toggle with a period of approximately 100 milliseconds. If not, debug.

7. Solution for Lab1_1.c

Lab1_1.c

```
1  //
2  //      Lab1: TMS320F28335
3  //      (c) Frank Bormann
4  //
5  //###########################################################################
6  //
7  // FILE:    Lab1_1.c
8  //
9  // TITLE:   DSP28335_Control_Card
10 //          blink LD2(GPIO31) and LD3(GPIO34);
11 //          solution file for Lab1
12 //###########################################################################
13 //  Ver | dd mmm yyyy | Who  | Description of changes
14 // =====|=============|======|===============================================
15 //  1.0 | 01 Jan 2010 | F.B. | Lab1 for F28335;
16 //  1.1 | 26 Aug 2010 | F.B. | ControlSUITE
17 //###########################################################################
18 extern void InitSysCtrl(void);
19
20 #include "DSP2833x_Device.h"
21 void main(void)
22 {
23     unsigned long i;
24     InitSysCtrl();
25     EALLOW;
26     GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;         // output for LED LD2
27     GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;         // output for LED LD3
28     EDIS;
29     GpioDataRegs.GPACLEAR.bit.GPIO31 = 1;
30     GpioDataRegs.GPBSET.bit.GPIO34 = 1;
31     while(1)
32     {
33         GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;  // toggle LD2
34         GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;  // toggle LD3
35         for (i=0; i<1000000; i++);          // time delay
36     }
37 }
38
```