F28069 ControlCard Lab1

Toggle LED LD2 (GPIO31) and LD3 (GPIO34)

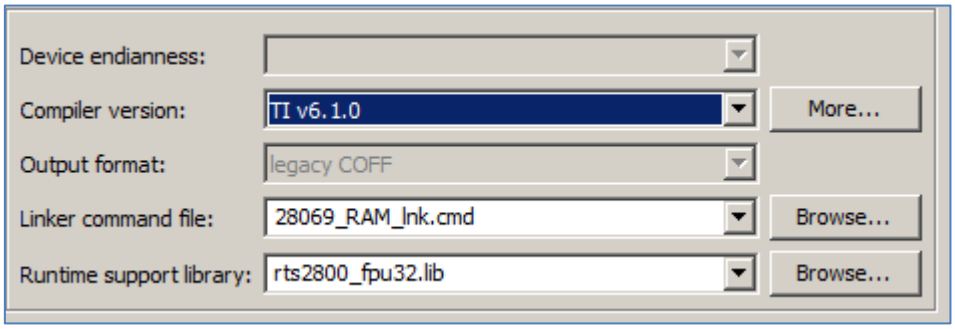
1. Project Dependencies

The project expects the following support files:

- Support files of controlSUITE installed in:

  C:\TI\controlSUITE\device_support\f28069\v135

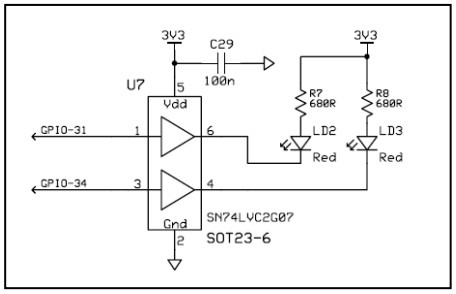
- Code Generation Tools Version 6.1.0:



- Code Composer Studio Version 5.3.0:

2. Project Objective

- Blink LD2 (GPIO31) and LD3 (GPIO34) at the F28069 ControlCard.

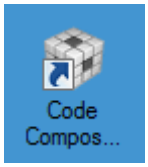


- Software delay loop in "main()" for approximately 100 milliseconds.
- No interrupt system in use.
- Watchdog Timer disabled.

3. Hardware Setup

No special setup required. Connect the F28069 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 F28069 controlCARD should be "on".

4. Start Code Composer Studio V5

Start Code Composer Studio V5: 

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.

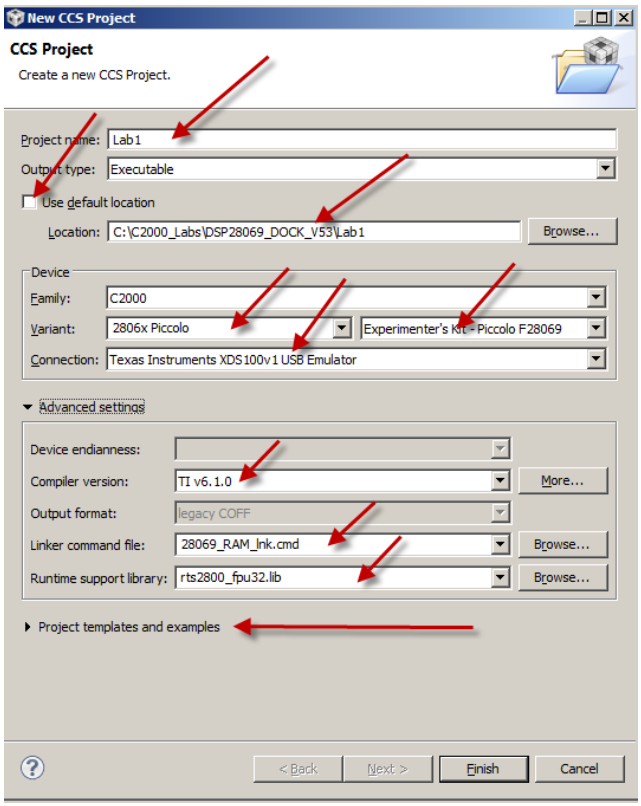
Note: The first time CCS opens a "Welcome to Code Composer Studio v5" page appears.

The term workbench refers to the desktop development environment. The workbench will open in the "CCS Edit" Perspective view. Notice the `CCS EDIT` *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 "CCS Edit" Perspective is used to create or build C/C++ or any other languages projects. A "Debug Perspective" view will automatically be enabled when the debug session is started.*

5. Project Design

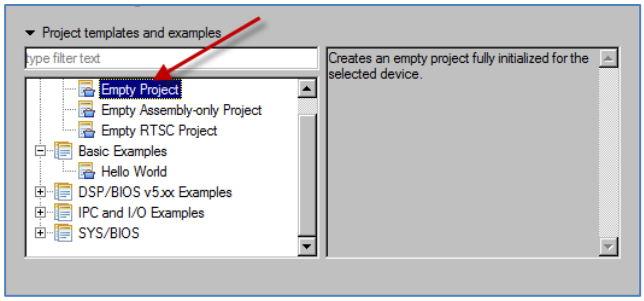
5.1. Create a new project

➔ **File ➔ New ➔ CCS Project:**



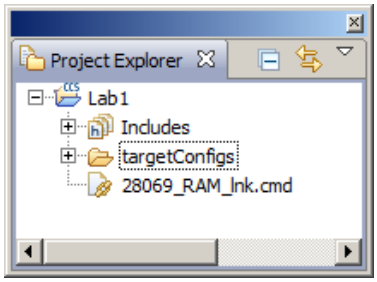
- Name the project “Lab1”
- Store it in a Location of your choice; make sure to store the project in a separate subfolder, e.g. “\Lab1”
- Select the Device Family “C2000”and the F28069
- Set the JTAG – connection to “Texas Instruments XDS100v1 USB Emulator”
- Select the Linker Command File “28069_RAM_lnk.cmd”
- Select the Runtime support library “rts2800_fpu32.lib”

Do NOT FINISH this window yet! Expand the “Project templates and examples” part and select “Empty Project”:



Now click “Finish”.

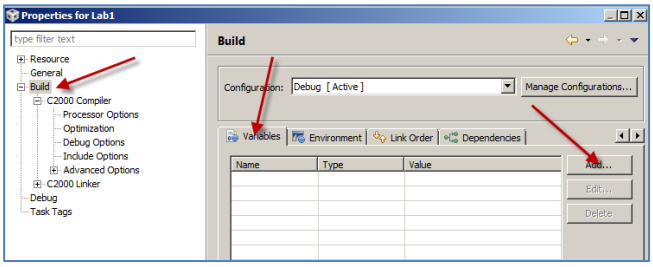
Next, change some of the project properties. In the project window, right click at “Lab1” and select “Properties”:



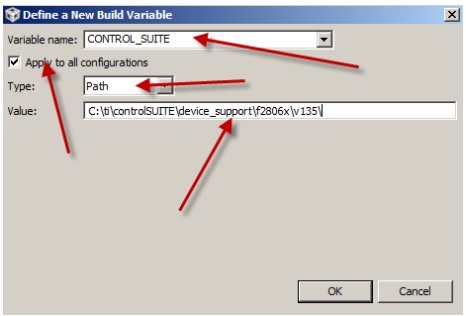
5.2. Edit Project Properties

5.2.1. Create a new Variable for include files

- Right Click at project “Lab1” and select “Properties”
- Add a new Build Variable:



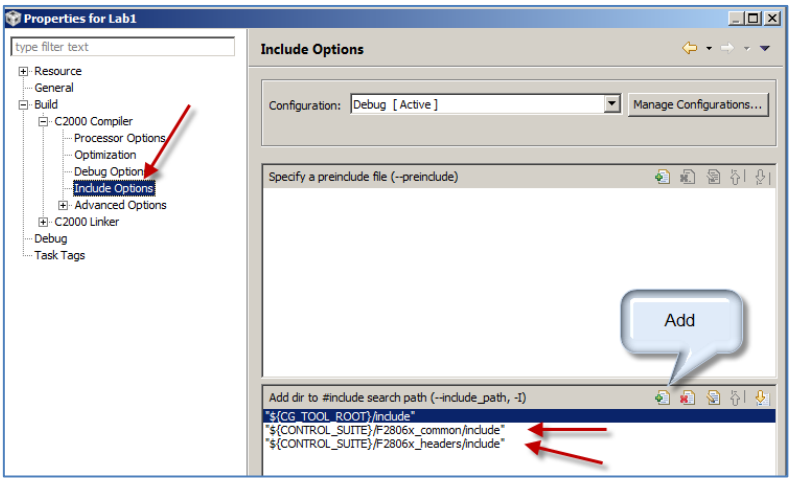
- Add:



5.2.2. Add 2 new include search path locations

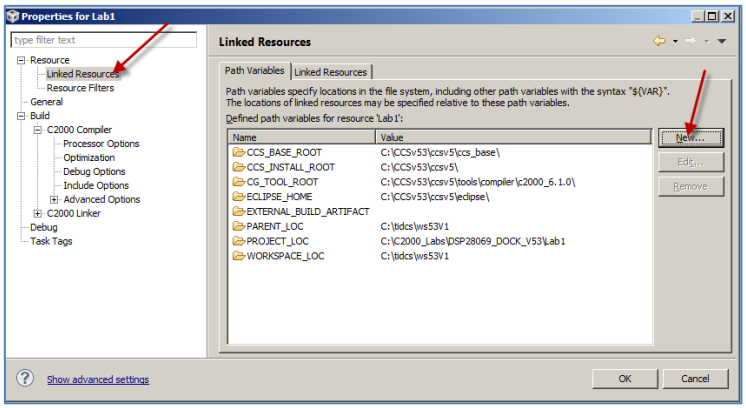
Extent the search path of the C-Compiler for additional include files. Select “Build”, “C2000 Compiler”, “Include Options”. In the box “Add dir to #include search path”, add the following lines:

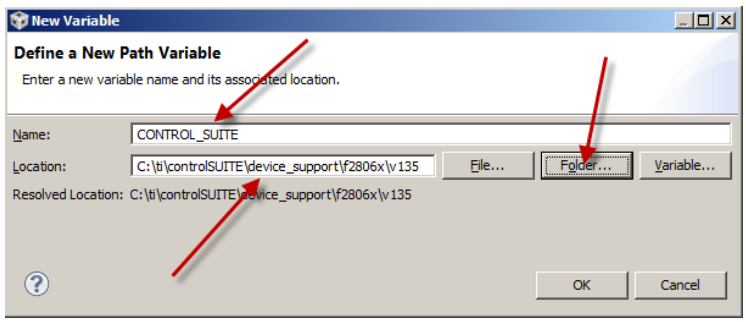
- **"${CONTROL_SUITE}/F2806x_common/include"**
- **"${CONTROL_SUITE}/F2806x_headers/include"**



5.2.3. Add a new link path location

- Create a new link path variable “CONTROL_SUITE”:





5.2.4. Change Diagnostic Options

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

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

Also, enter the following numbers:

- Suppress Diagnostic: 238
- Treat Diagnostic as Error: 225
- Treat Diagnostic as Error: 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**>.

5.3. Link Files to Project

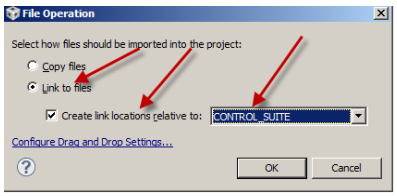
In the project window, right click at "Lab1" and select "Add Files":
First add a file, which defines the physical addresses for all data memory mapped registers:

```
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_headers\cmd\F2806x_Headers_nonBIOS.cmd
```

If you select the file, a new window will appear, asking you to select between "Add" and "Link" this file.

- *"Add File": makes a copy of the original file and work with that copy in the local project directory.*
- *"Link File": sets a reference to the original file to use this original file.*

If we do not need to change the file contents, "Link" is the preferred selection. It is also recommended to use a "relative" link, relative to the new link variable "CONTROL_SUITE":



In the same way, repeat the linking of additional files for:

- the definition for all global variables to access the F28069 hardware registers:

```
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_headers\source\F2806x_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").

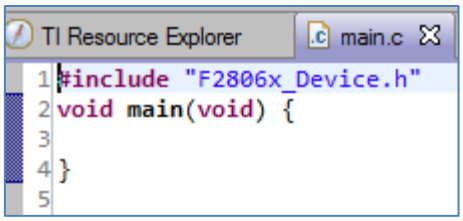
```
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_common\source\F2806x_CodeStartBranch.asm
```

- Link 2 more files with functions to perform the basic core initialization of the F28069:

```
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_common\source\F2806x_SysCtrl.c
C:\TI\controlSUITE\device_support\f2806x\v135\F2806x_common\source\F2806x_usDelay.asm
```

5.4. Create the main C – file

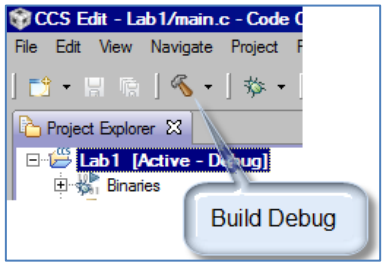
Create a new file "Lab1_1.c" and enter the following "#include" line:



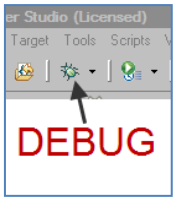
5.5. Build and Download the Project

➔Project ➔ Build Project

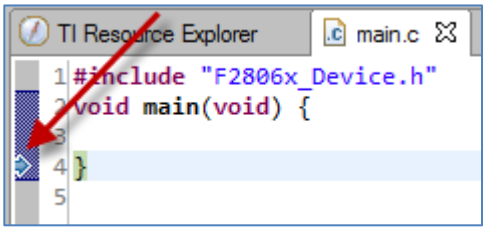
- Click on "Build Debug":



- Click 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 properly downloaded into the F28069.



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 F28069controlCARD.

- switch back to the “CCS Edit” 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 “F2806x_SysCtrl.c” and will initialize the device to 80 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”:

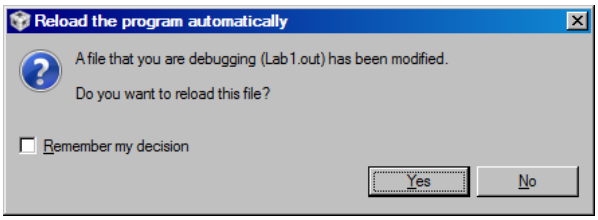
```
while(1)
{
GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;     // toggle LD2
```

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

6.2. Rebuild Project

➔ Project ➔ Build Project

A window will pop up to indicate that the machine code for the target has changed:



Answer: “Yes”.

If the compilation and the download were successful, the instruction pointer (blue arrow) is 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

```
//
//      Lab1: TMS320F28069
//      (c) Frank Bormann
//
//###########################################################################
//
// FILE:    Lab1_1.c
//
// TITLE:   DSP28069_Control_Card
//          blink LD2(GPIO31) and LD3(GPIO34);
//          solution file for Lab1
//###########################################################################
//  Ver | dd mmm yyyy | Who  | Description of changes
// =====|=============|======|===============================================
//  1.0 | 25 Jan 2011 | F.B. | Lab1 for F28069;
//###########################################################################
#include "F2806x_Device.h"
extern void InitSysCtrl(void);

void main(void)
{
    unsigned long i;
    InitSysCtrl();
    EALLOW;
    GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;         // output for LED LD2
    GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;         // output for LED LD3
    EDIS;
    GpioDataRegs.GPACLEAR.bit.GPIO31 = 1;
    GpioDataRegs.GPBSET.bit.GPIO34 = 1;
    while(1)
    {
        GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;  // toggle LD2
        GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;  // toggle LD3
        for (i=0; i<1000000; i++);          // time delay
    }
}
```