PURDUE
UNIVERSITY

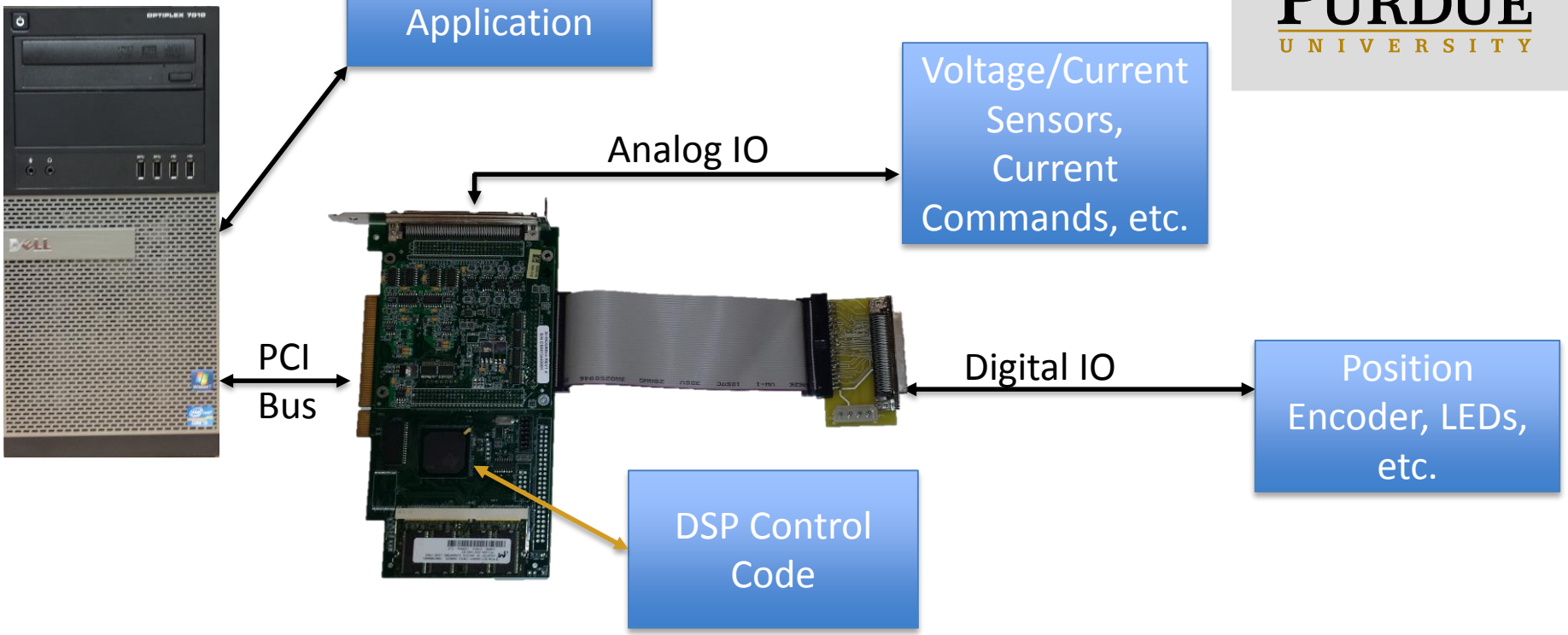
SHELDON DSP TUTORIAL

Dan Horvath

dhorvath@purdue.edu

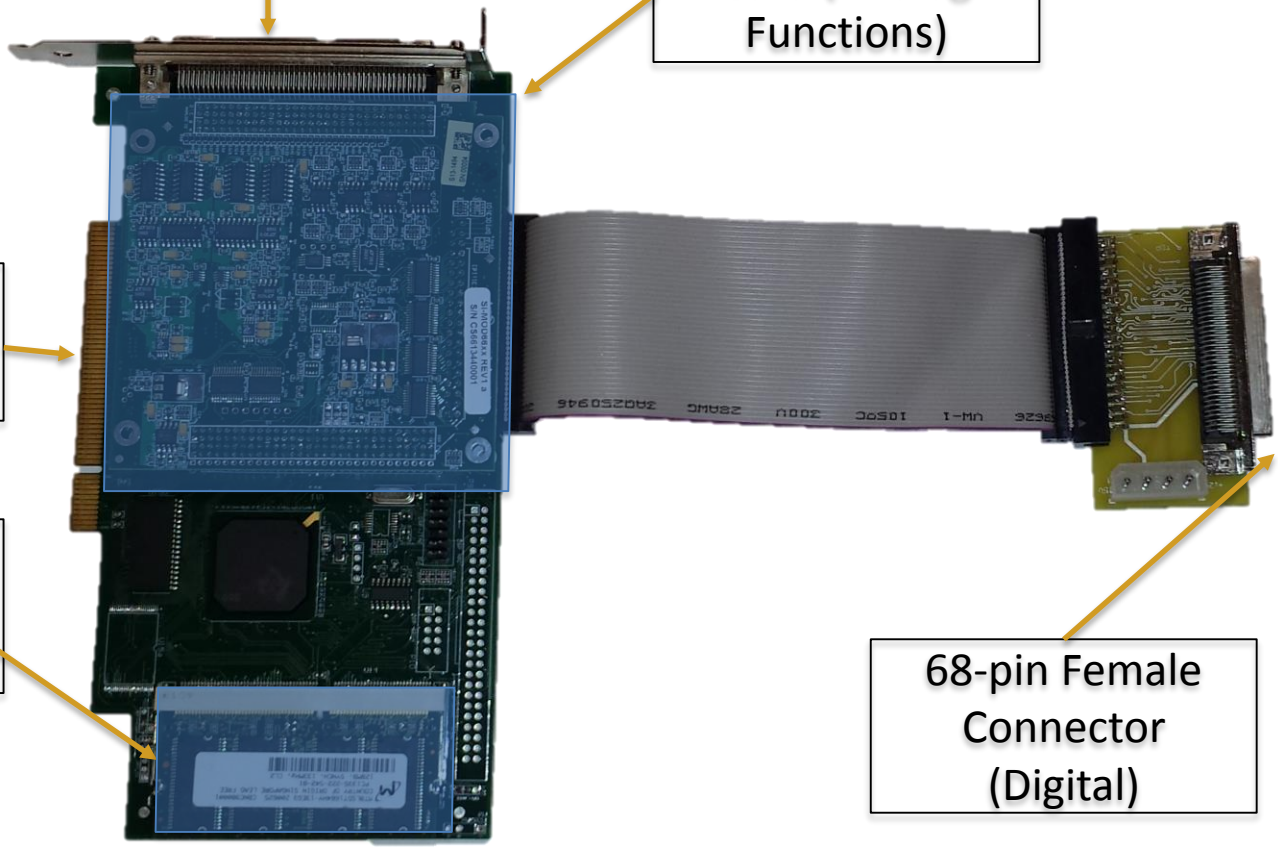
Ver. 1.0 September 30, 2014

PURDUE UNIVERSITY



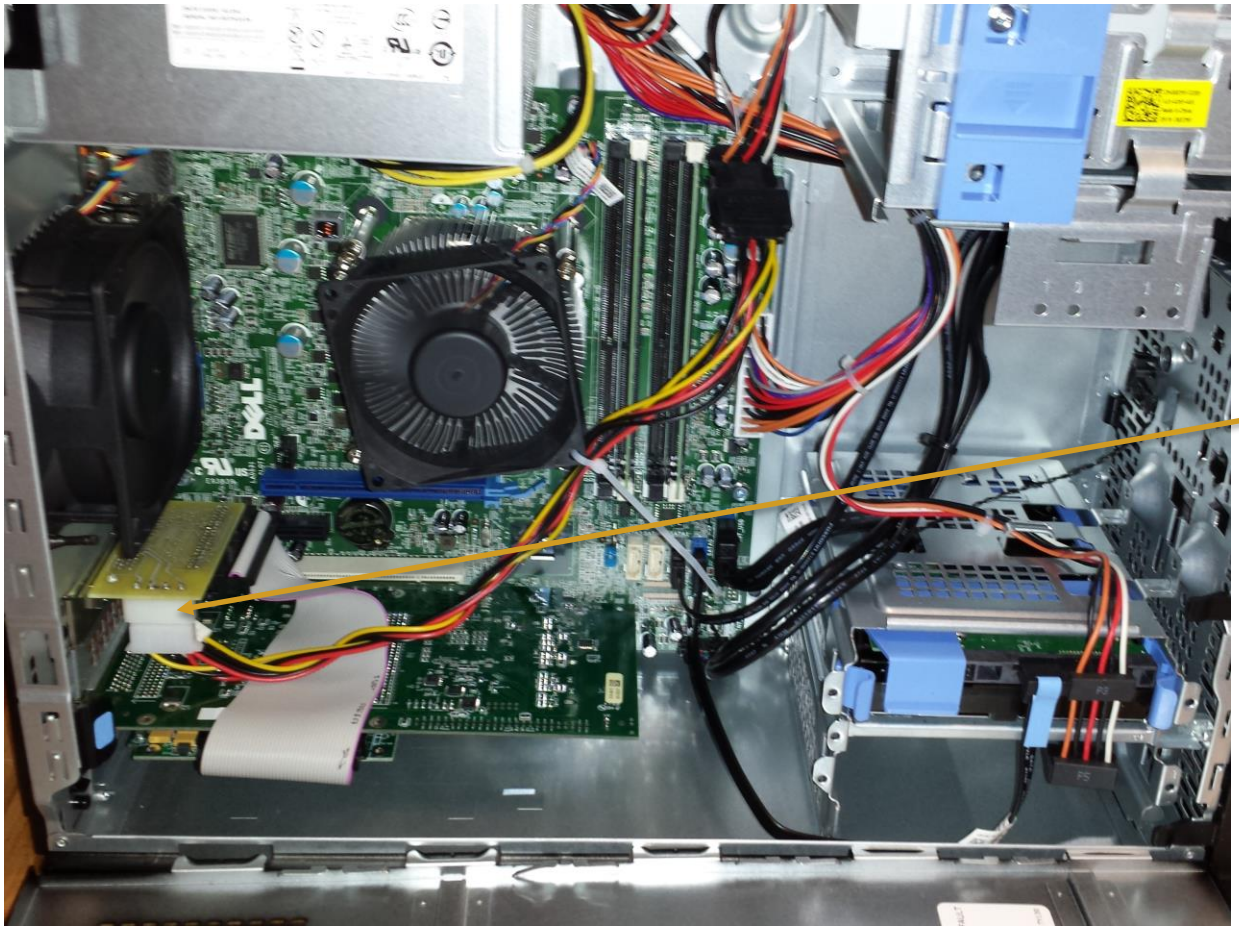
DSP HARDWARE OVERVIEW

DSP CARD COMPONENTS



PURDUE UNIVERSITY

INSTALLATION



Provides power to DSP Distribution System from PC Power Supply

PURDUE UNIVERSITY

CONNECTORS

68-pin Female Connector (Digital)

Pins in these connectors are not the typical cylindrical pins seen in other connectors

100-pin Female Connector (Analog)

PURDUE UNIVERSITY

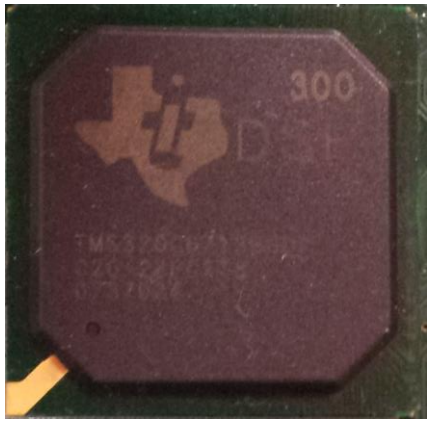
SPECIFICATIONS

TI TMS320C6713

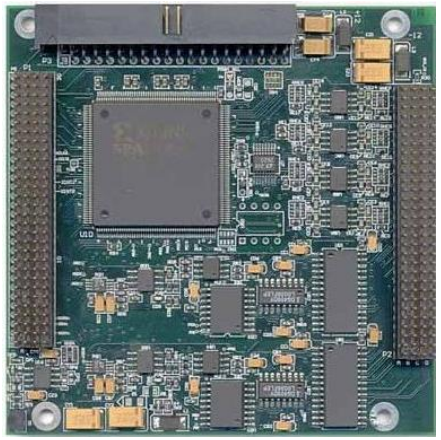
300 MHz, 32 bit

1800 MFLOP

36 Digital IO bits available

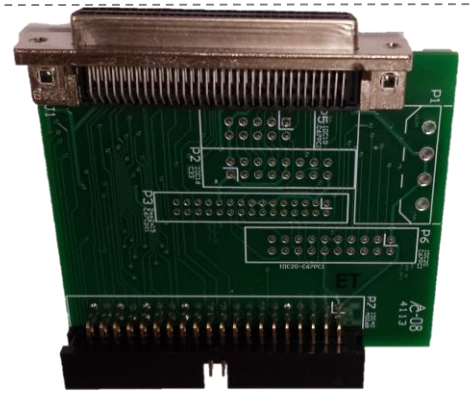


- SI-MOD6632-HG-250-16DAC Daughter Card
- 16 differential-ended analog input channels or 32 single-ended
- 16 Analog output channels
- All analog is 16 bits (input and output)
- Analog voltage Max +10V, Min -10V (input and output)
- Digital output voltage 0V – 3.3V

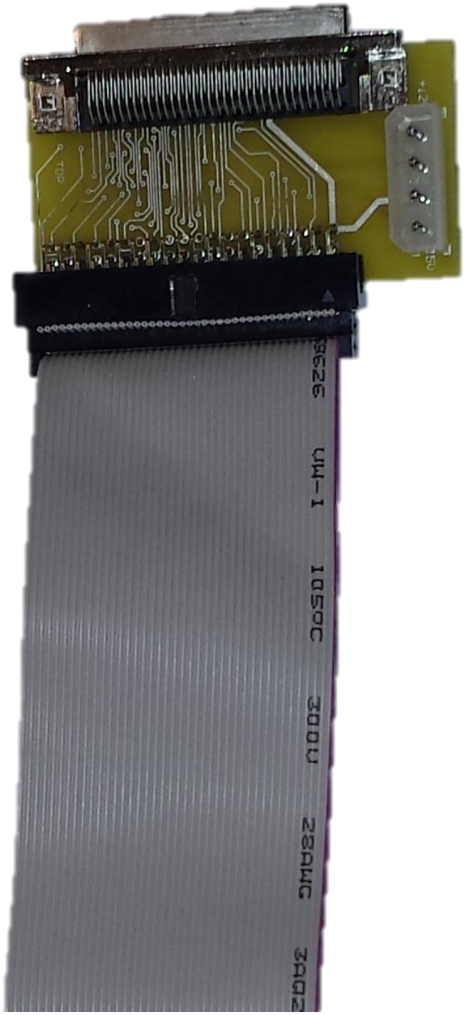


PURDUE UNIVERSITY

IMPORTANT!



≠



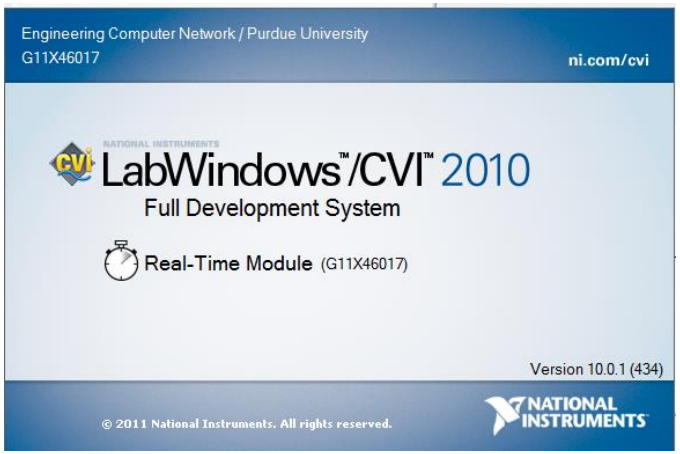
There is an important pinout difference here between the two PCBs

Plugging in DSP Dist. Box WILL SHORT VCC AND GND WHEN USING GREEN ADAPTOR

USE THE YELLOW ONE! (No Solder Mask)

PURDUE UNIVERSITY

PURDUE
UNIVERSITY



DSP SOFTWARE OVERVIEW

SOFTWARE DEVELOPMENT

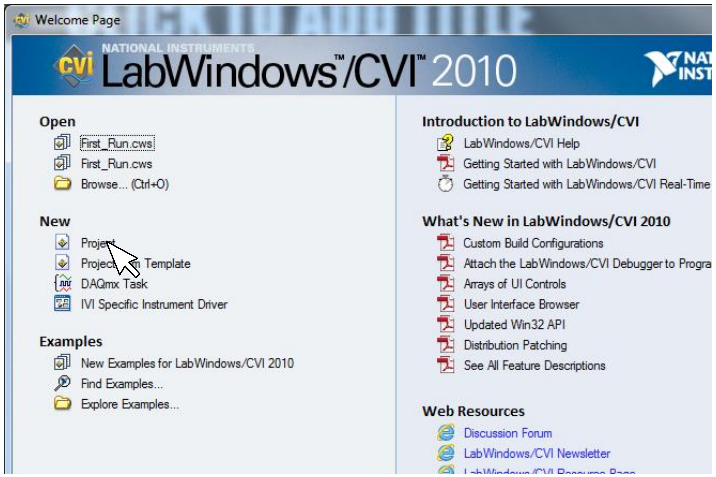
Three Components to Developing a Complete Control System:

- Desktop Application – National Instruments LabWindows/CVI (GUI)
- DSP Application – Texas Instruments Code Composer Studio (code running on DSP that controls equipment)
- Library that allows PC and DSP to talk – precompiled DLL (SIc67.dll)

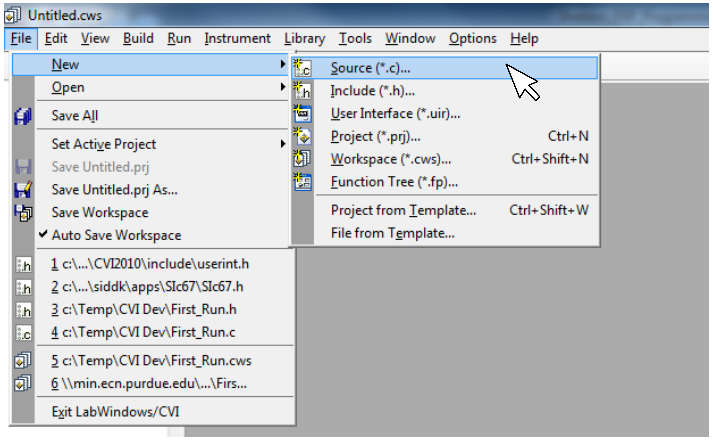
Only really need to deal with first 2 for most applications, will call the DLL functions but most likely will not need to modify their code

NI LABWINDOWS/CVI

CREATING A NEW PROJECT AND SOURCE FILE



Click “Project” under “New” on the left

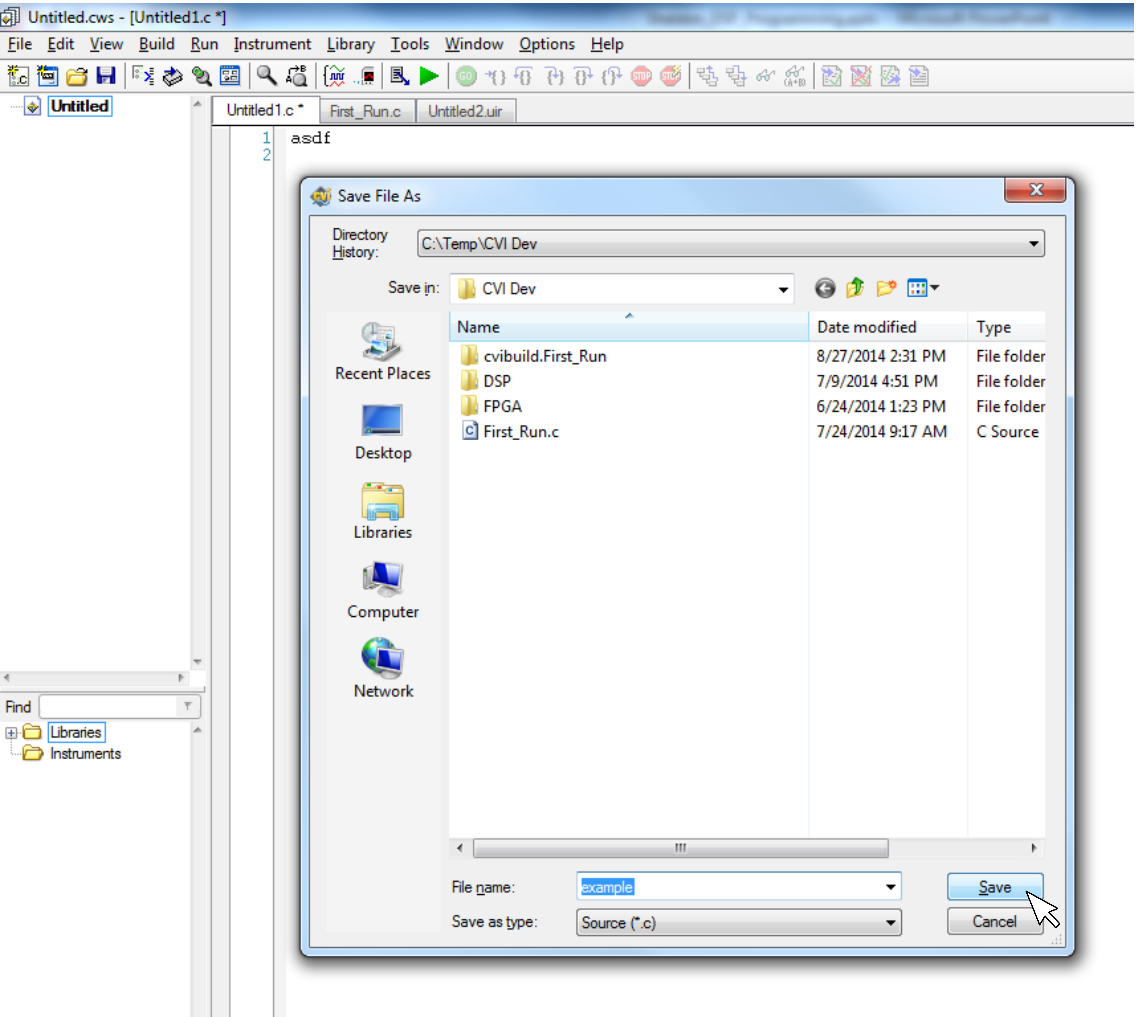


Click “File”, “New”, “Source(*.c)…”

PURDUE UNIVERSITY

NI LABWINDOWS/CVI

SAVING SOURCE FILE

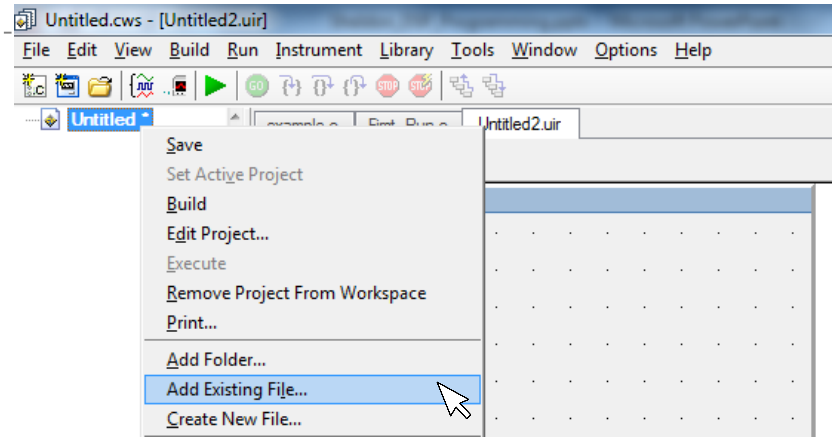


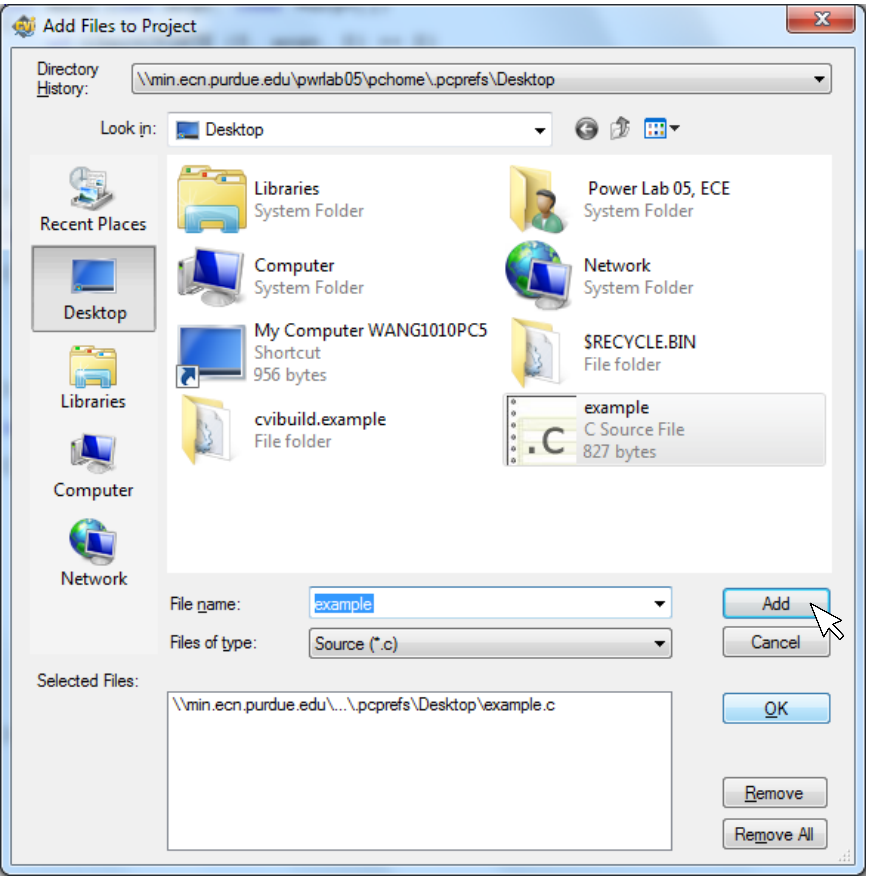
1. Type some text. “asdf” for example
2. Then save the file.

NI LABWINDOWS/CVI

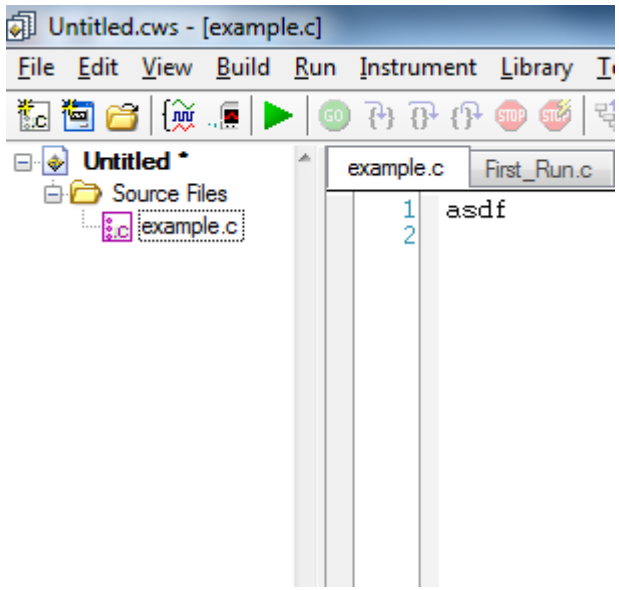
CREATING A NEW PROJECT

1. Add the file to the project by right clicking the project name and then clicking “Add Existing File” and add the source file you just saved.





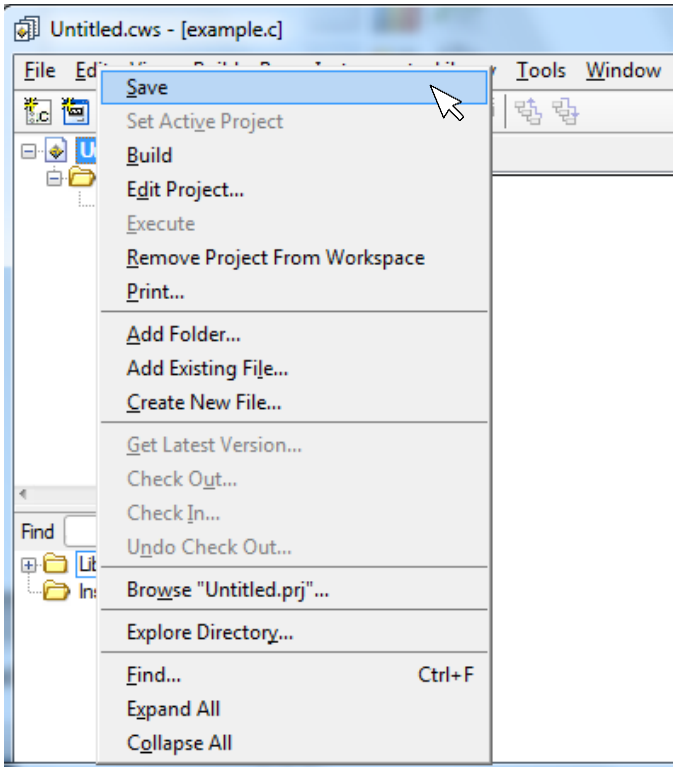
2. The project now shows the source file in the project tree

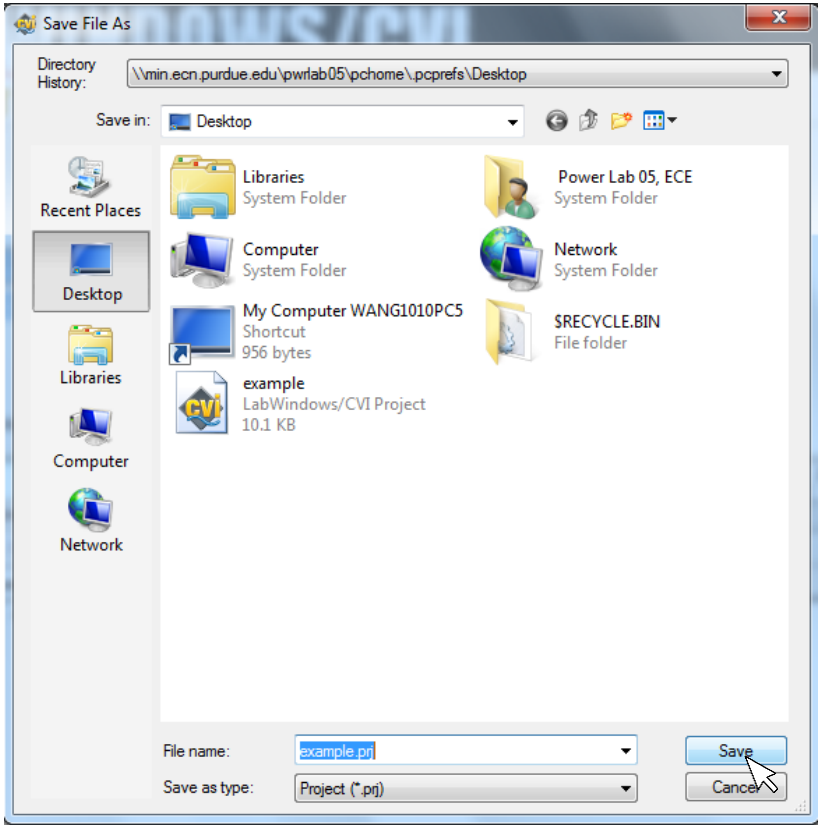


NI LABWINDOWS/CVI

SAVING THE PROJECT

Right click the project name in the tree and click “Save”

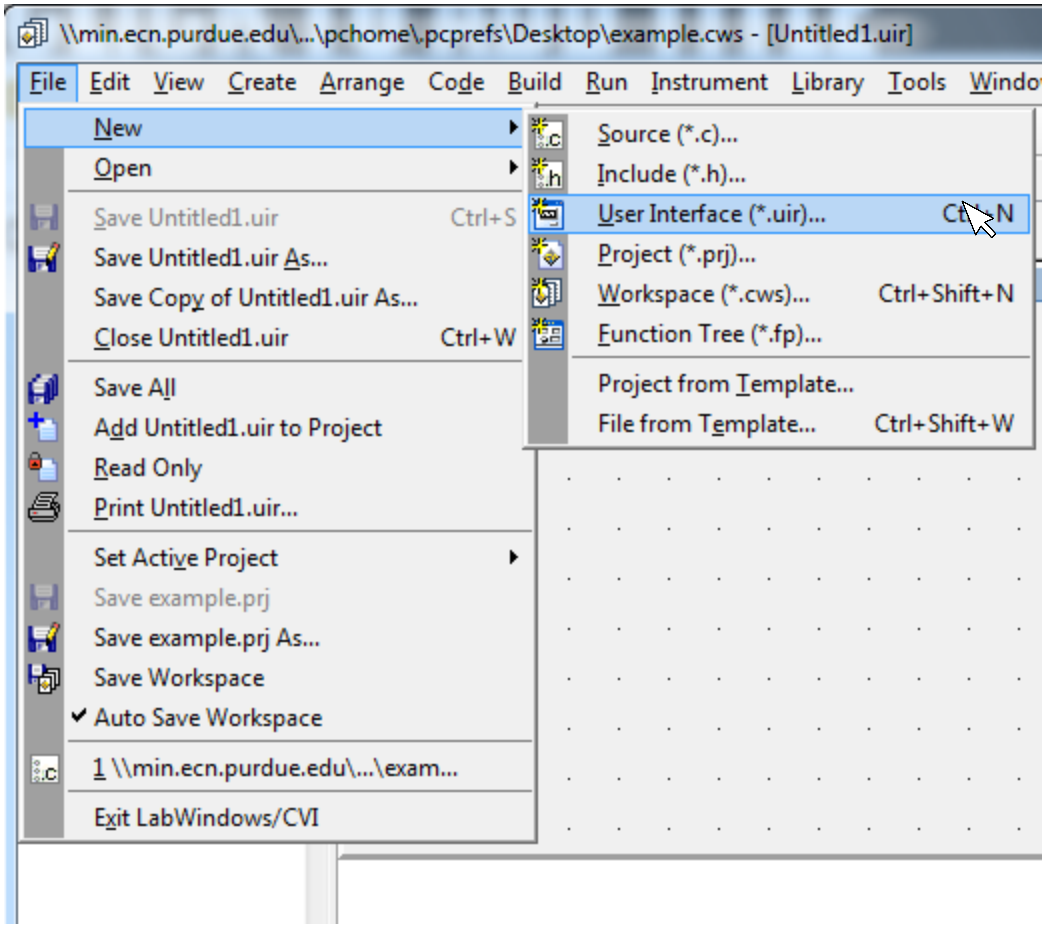




PURDUE UNIVERSITY

NI LABWINDOWS/CVI

CREATING A GUI



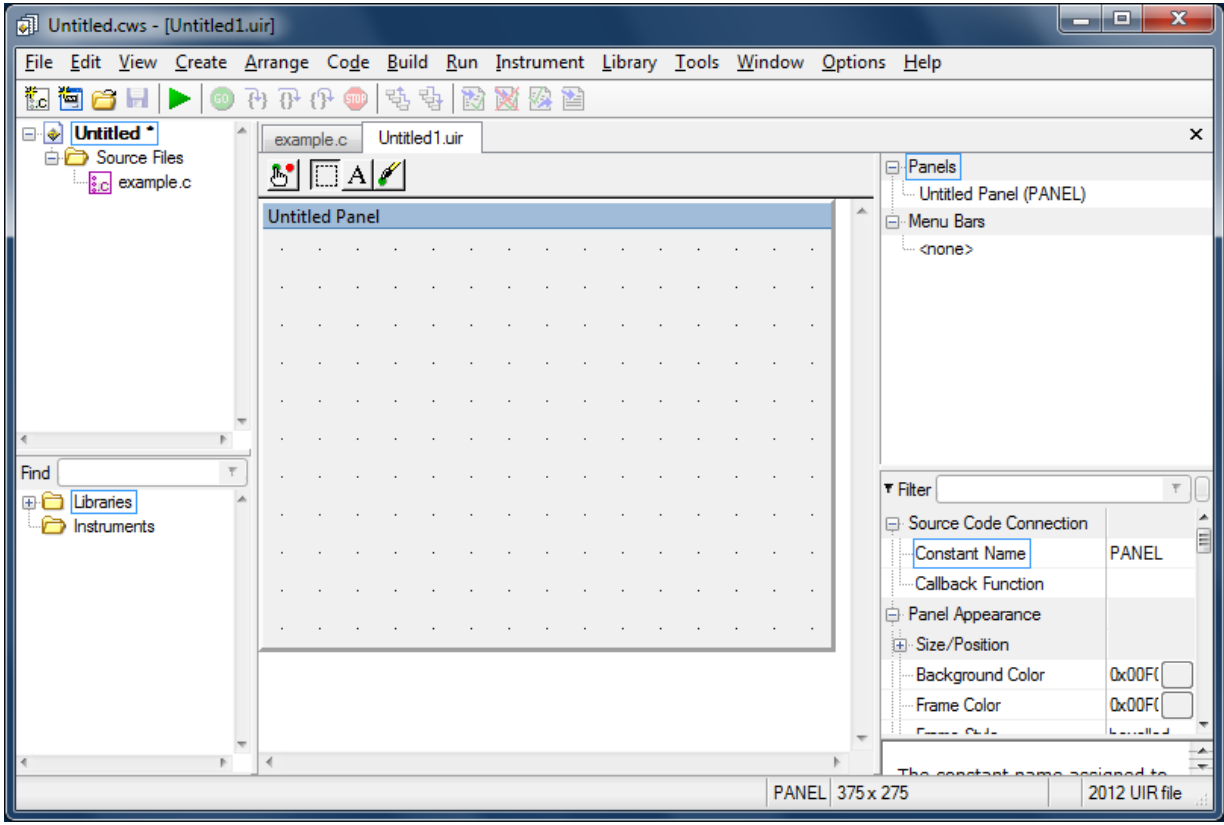
Create a new GUI by clicking “File,” “New,” “User Interface (*.uir)…”

PURDUE UNIVERSITY

NI LABWINDOWS/CVI

GUI PANEL

A blank GUI panel should be shown.

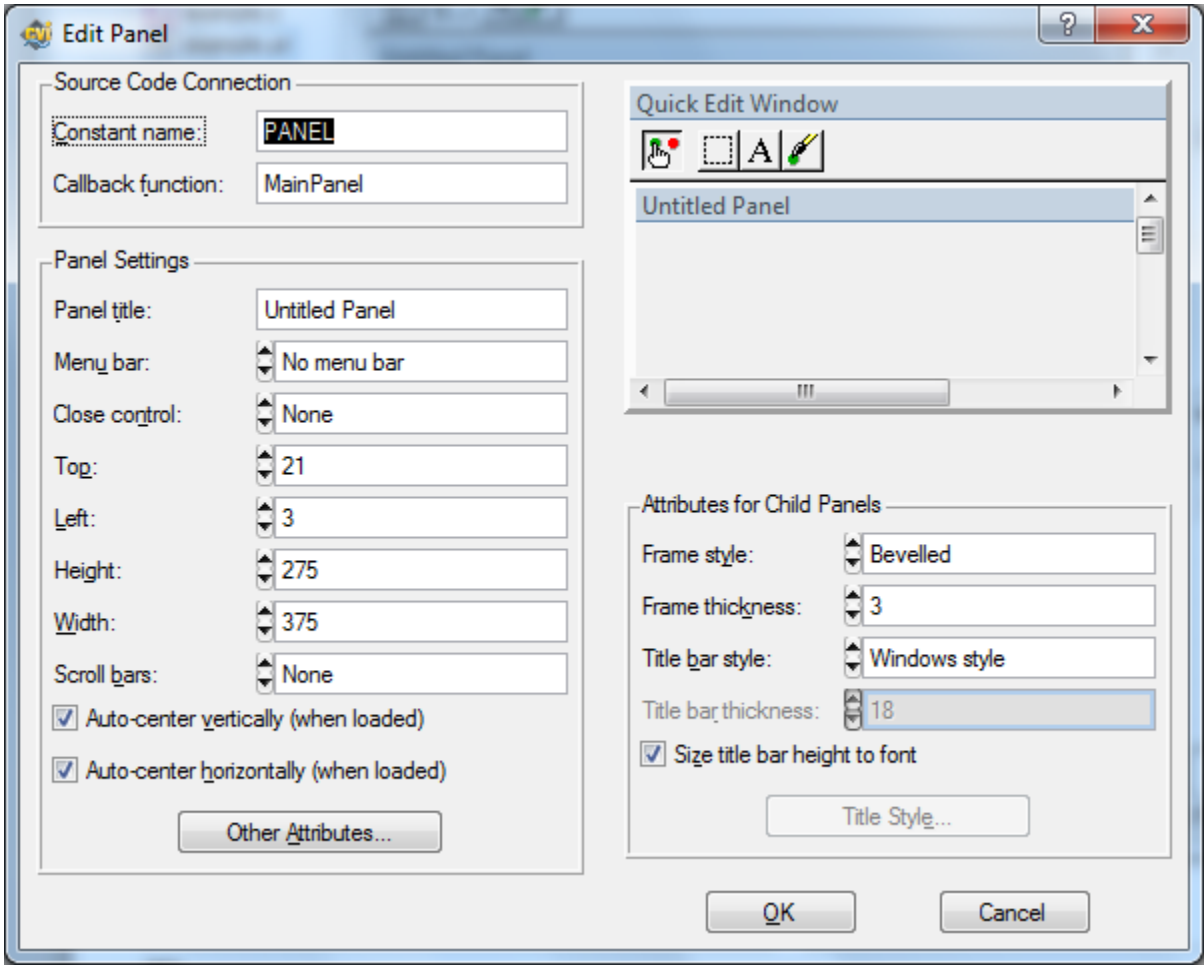


PURDUE UNIVERSITY

NI LABWINDOWS/CVI

PANEL CALLBACK FUNCTION NAMING

Double click the grey area in the newly-created panel; the panel's settings window will be brought up, shown right. In the "Callback function" field, type "MainPanel." This is the name of the function that gets called every time something happens in this GUI panel (Getting focus of the window, exiting the window, etc.).

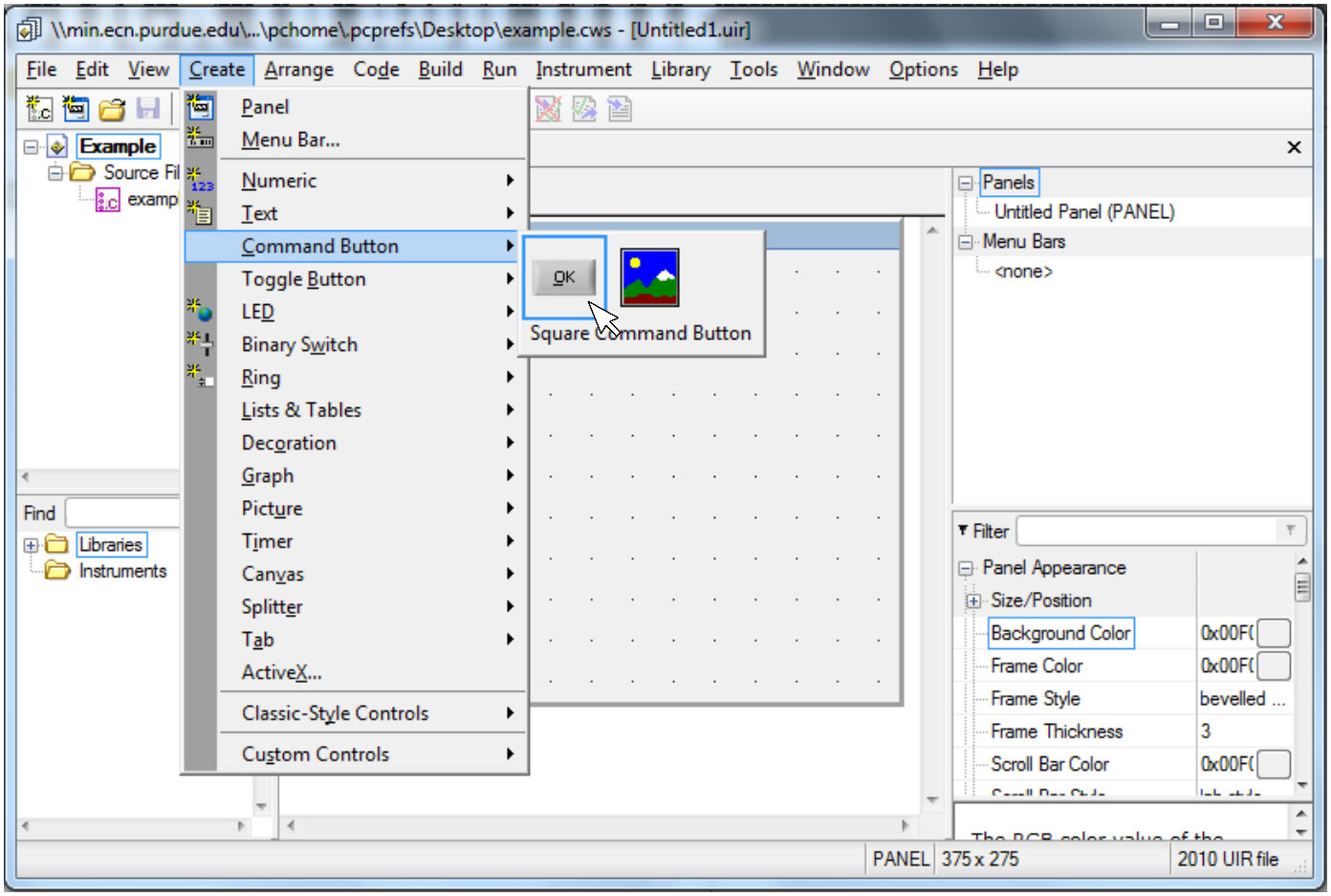


PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ADD A BUTTON TO THE GUI

Click “Create,” “Command Button,” “Square Command Button”

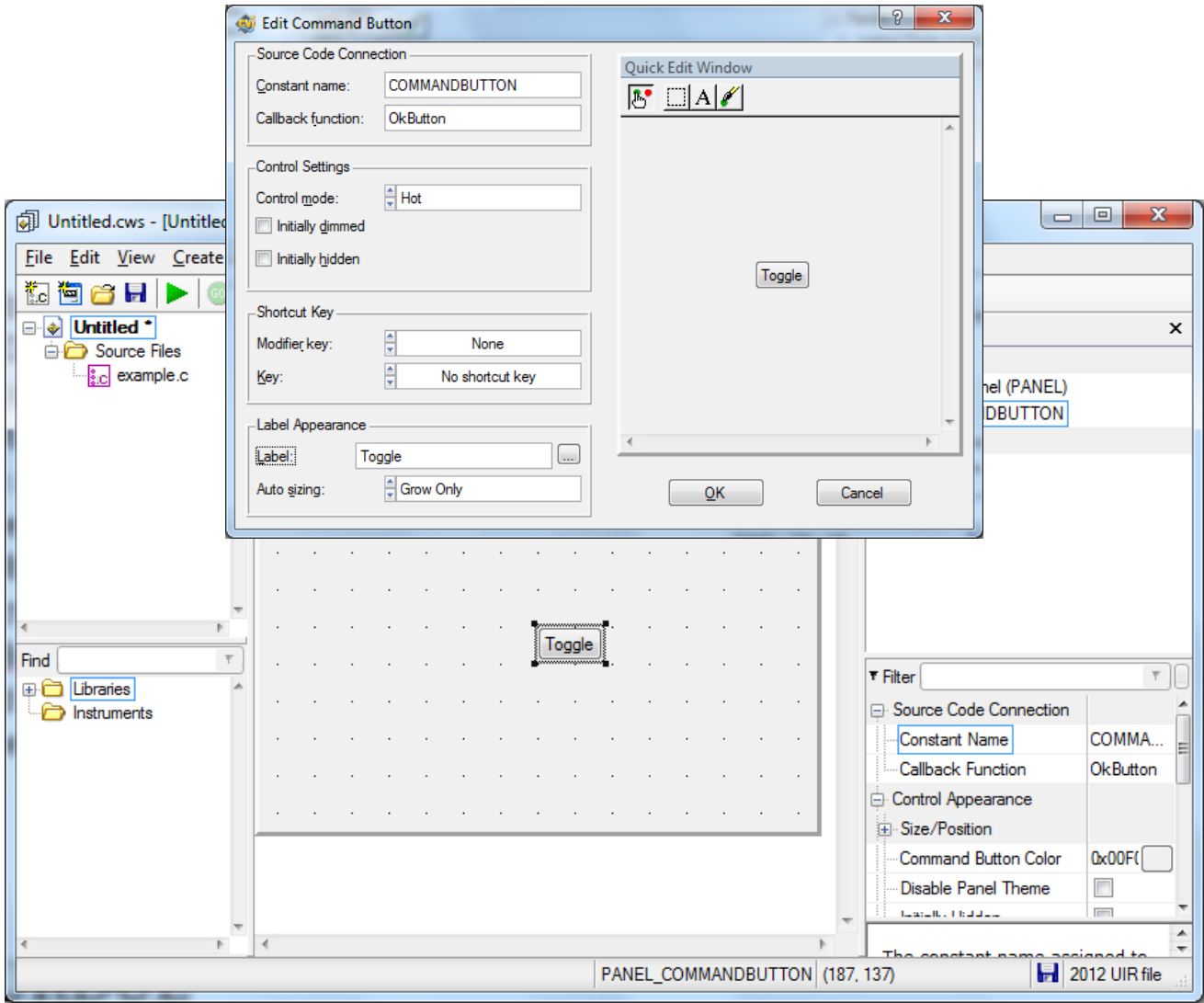


PURDUE UNIVERSITY

NI LABWINDOWS/CVI

NAMING THE BUTTON CALLBACK FUNCTION

Double click the OK button you just created and type a function name into the “Callback Function” box. “OkButton” is a good function name. Type “Toggle” in the Label field. Click “OK.”

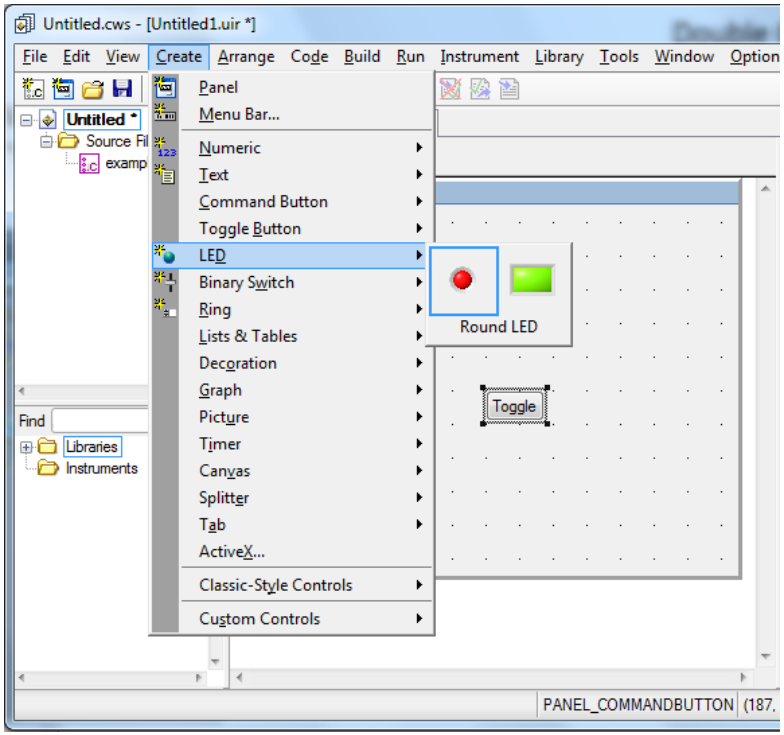


PURDUE UNIVERSITY

NI LABWINDOWS/CVI

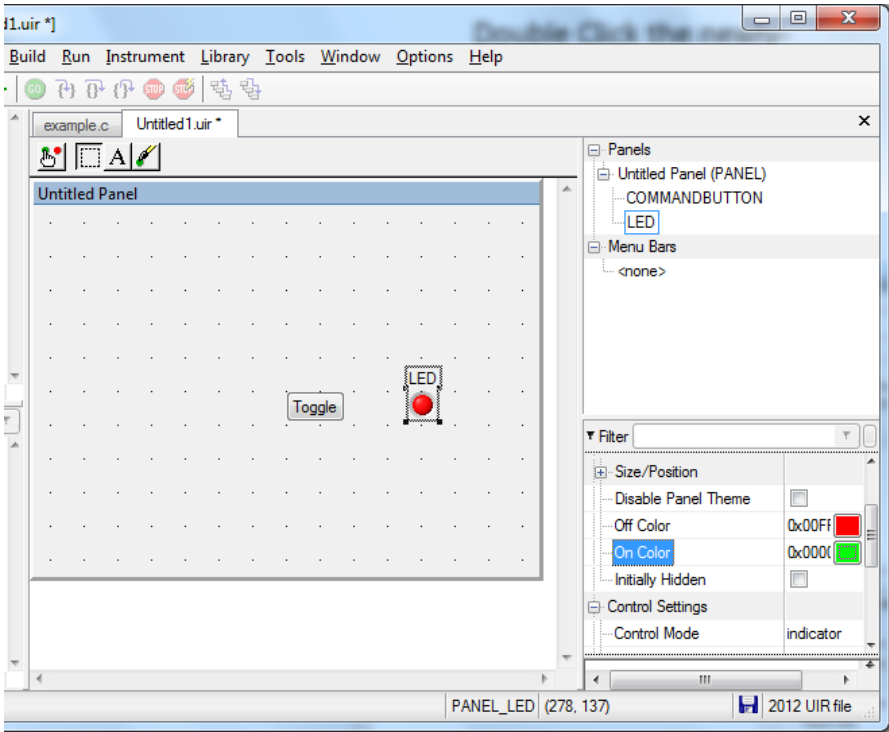
ADD AN LED TO THE GUI

With .uir editor in focus, Click “Create,” “LED,” “Round LED.”



Double Click the newly-created LED.
Change “Label Appearance” field to “LED.”

With the LED selected, make Off Color 0x00FF0000,
On Color 0x0000FF00 in the menu - bottom right

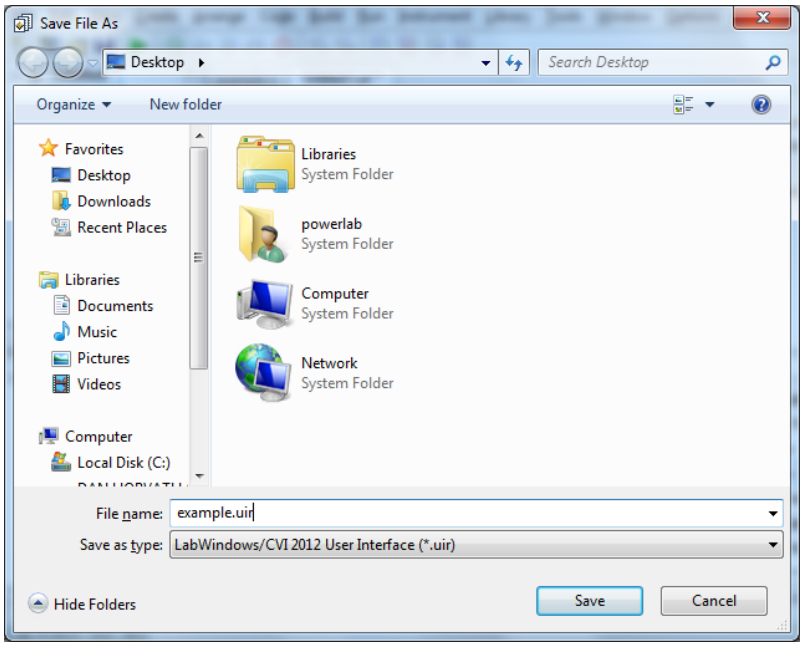


PURDUE UNIVERSITY

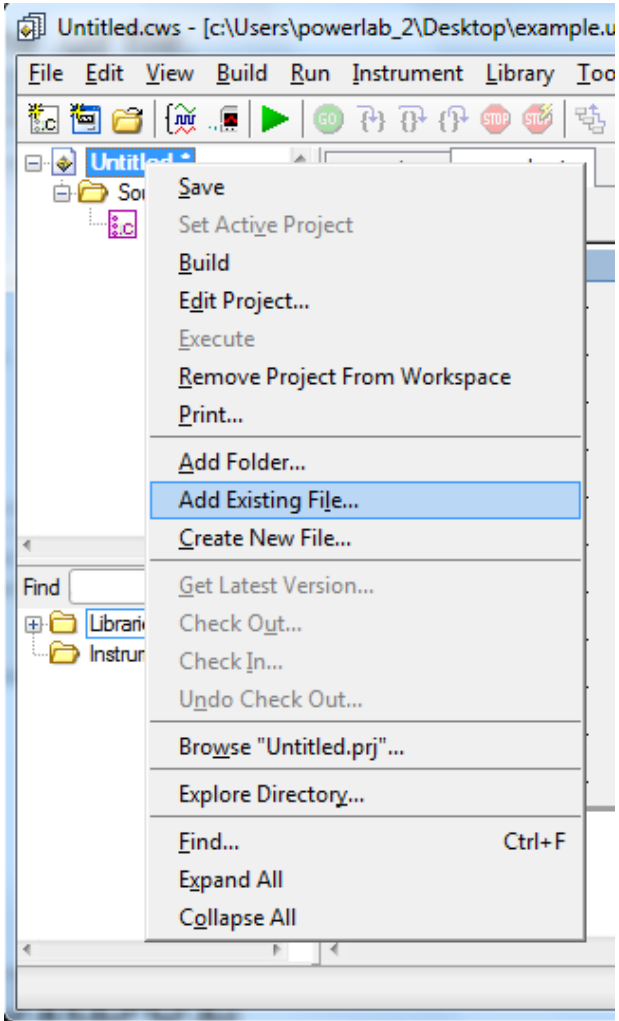
NI LABWINDOWS/CVI

ADD THE GUI FILE TO THE PROJECT

Save the .uir file.



Right click project, click add existing file, and add example.uir to the project. Then save the project.



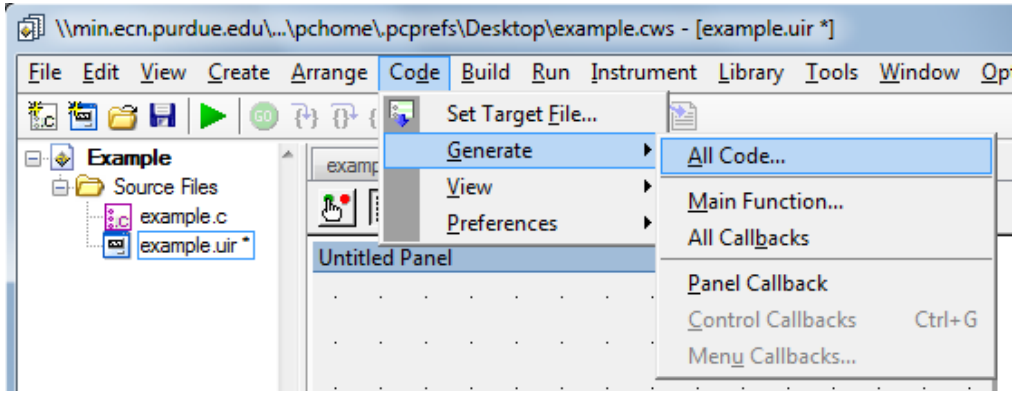
PURDUE UNIVERSITY

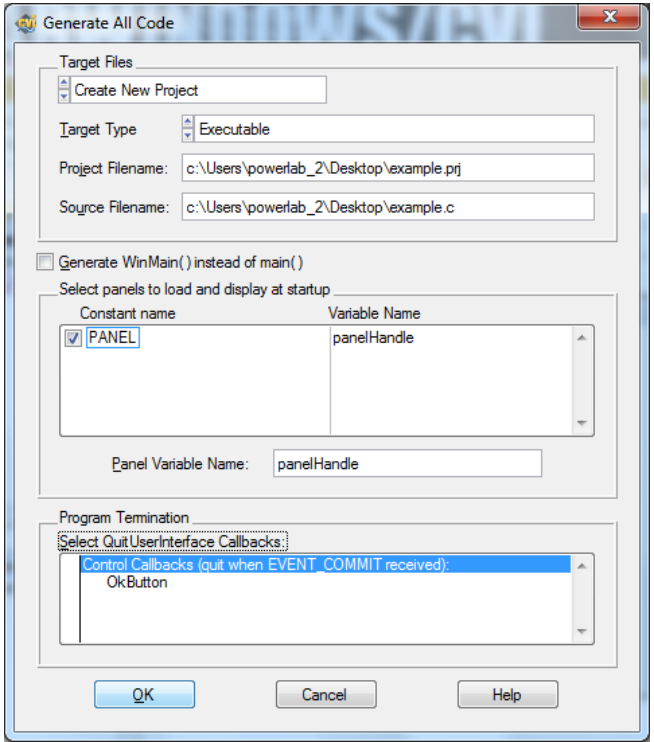
NI LABWINDOWS/CVI

GENERATING THE MAIN FUNCTION AND CALLBACK FUNCTIONS

With .uir editor in focus, Click “Code,” “Generate,” “All Code…”

This generates the main function and the callback functions using the function names that we selected.





PURDUE UNIVERSITY

NI LABWINDOWS/CVI

INITIAL CODE GENERATION

The main function is generated, as well as callback functions for the main panel and the OK button. These functions don't do anything at the moment.

```
#include <cvirte.h>
#include <userint.h>
#include "example.h"

static int panelHandle;

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "example.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    DiscardPanel (panelHandle);
    return 0;
}

int CVICALLBACK OkButton (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
    }
    return 0;
}

int CVICALLBACK MainPanel (int panel, int event, void *callbackData,
        int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_GOT_FOCUS:

            break;
        case EVENT_LOST_FOCUS:

            break;
        case EVENT_CLOSE:

            break;
    }
    return 0;
}
```

PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ADDING CODE TO THE OKBUTTON CALLBACK FUNCTION

Add the code shown right to the OkButton callback function. Declare the variable "int temp;" before the switch statement. GetCtrlVal() grabs the current value(color) of the LED. SetCtrlVal() changes the value of the LED. The rest of the code changes the LED to the opposite state of the current state. The point of this button is to toggle the LED between red and green each time the button is pressed.

```
int CVICALLBACK OkButton (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    int temp;
    switch (event)
    {
        case EVENT_COMMIT:
            GetCtrlVal(panel, PANEL_LED, &temp);
            if(temp == 1)
                SetCtrlVal(panel, PANEL_LED, 0);
            else
                SetCtrlVal(panel, PANEL_LED, 1);
            break;
    }
    return 0;
}
```

PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ADD CODE TO MAIN PANEL CALLBACK

Add the function call "QuitUserInterface(0);" to the "EVENT_CLOSE" case. This will allow the application to exit when you press the exit button. If you forget this, it's semi-annoying because you have to go into Task Manager and kill the process.

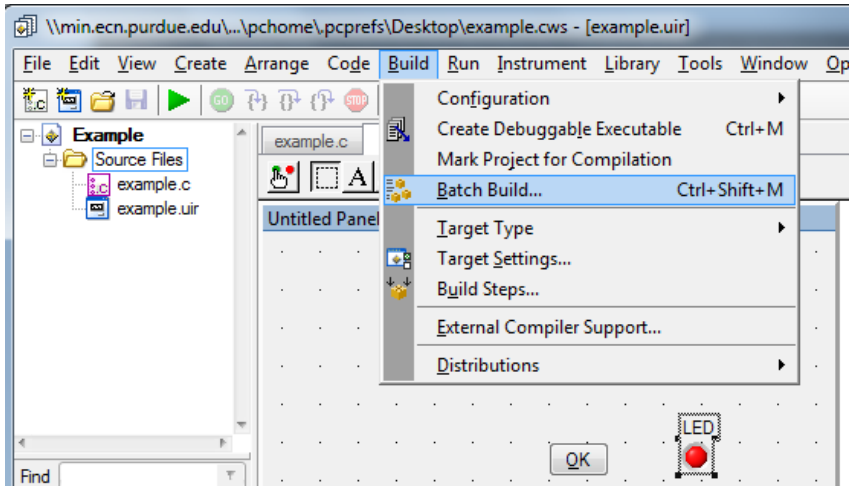
```
int CVICALLBACK MainPanel (int panel, int event, void *callbackData,
        int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_GOT_FOCUS:

            break;
        case EVENT_LOST_FOCUS:

            break;
        case EVENT_CLOSE:
            QuitUserInterface(0);
    }
    return 0;
}
```

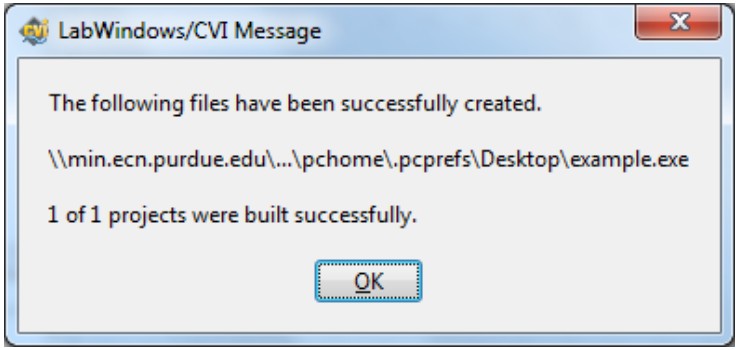
PURDUE UNIVERSITY

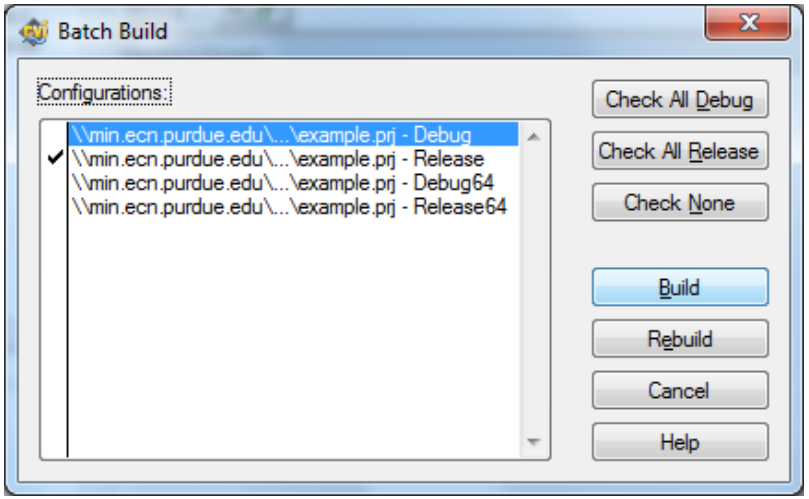
NI LABWINDOWS/CVI

SAVE AND BUILD

Bring the .uir editor into focus and save it (ctrl + s). Bring the code editor into focus and save example.c.
Once all is saved, click “Build,” “Batch Build” from the Menu Bar
Click “Release” option (checkmarked option bottom right), and then “Build.”
If compile is successful, the message below left will pop up.





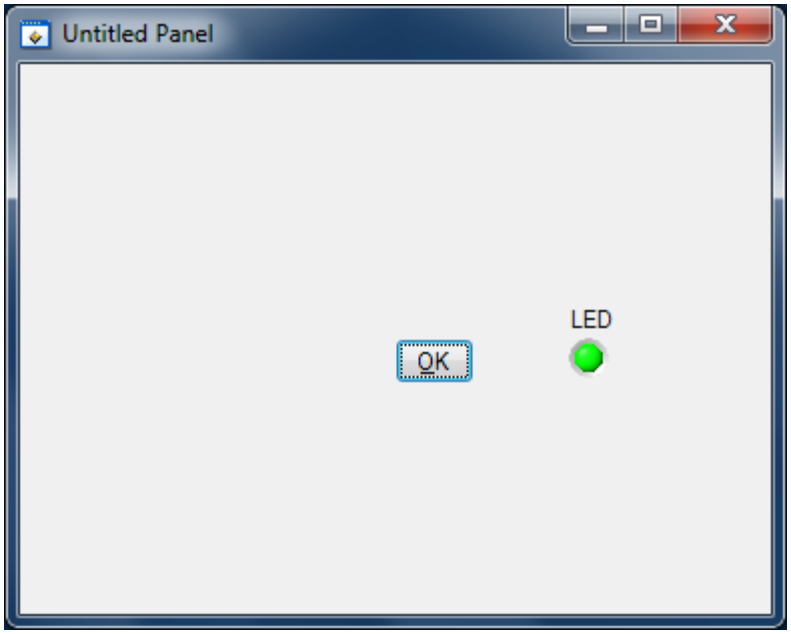


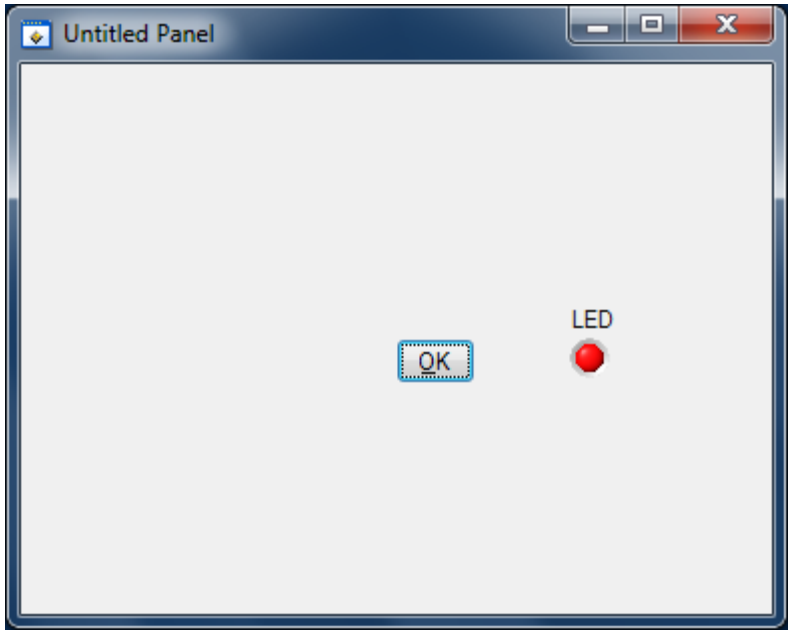
PURDUE UNIVERSITY

NI LABWINDOWS/CVI

OPERATION OF THE EXECUTABLE

Run the .exe file that is produced by the build. If everything went well, you will now have an application that you can run, and when you click the OK button, the LED color toggles.



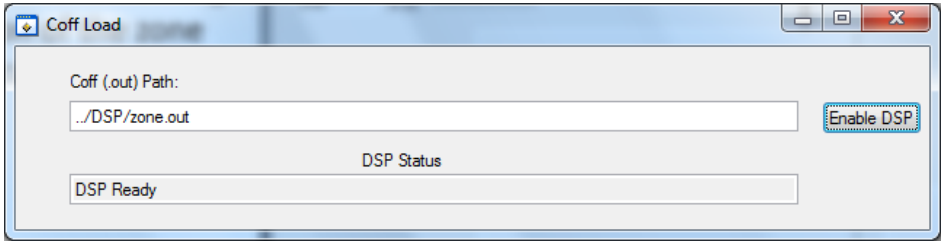


PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ZONES - MOST RECENT WORKING VERSION

Shown right is the most recent working version of the zone application. It begins with a splash screen to pick which coff file you want to load (Common Object File Format – the executable that runs on the DSP) onto the DSP and is equipped with an Enable DSP button and a DSP Status bar.



PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ZONES - ENABLE DSP CALLBACK (1/3)

The call to DLL_RESET() (the source code for which resides in the Sheldon DLL project developed in Visual Studio) releases the DSP from reset – the parameter passed to it is e_Enable_DSP as opposed to e_Disable_DSP which does the opposite and is used later. SIc67_LoadCofffile() attempts to load the coff file specified in the input line. The rest of the code on this page just does error checking and updates the DSP status window.

```
int CVICALLBACK EnableDSP (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    int error;
    UINT32 sync = 0;
    UINT32 buffer[2];
    UINT32 count = 1;
    UINT32 region = 1;

    if (event == EVENT_COMMIT)
    {
        error = DLL_RESET(boardID, e_Enable_DSP);
        if(error)
        {
            sprintf(errmsg, "DLL_RESET failed with return value %d\n", error);
            SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
        }
        sprintf(errmsg, "DSP Reset Lifted");
        SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
        GetCtrlVal(panel, CONFIG_COFF_FILE_PATH, coffpath);
        error = SIc67_LoadCofffile(boardID, coffpath ,0);
        if(error)
        {
            sprintf(errmsg, "SIc67_LoadCofffile() failed with return value %d\n", error);
            SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
            return 0;
        }
        sprintf(errmsg, "COFF Load Successful");
        SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
```

NI LABWINDOWS/CVI

ZONES - ENABLE DSP CALLBACK (2/3)

Syncing with the DSP involves checking a Communication Register inside the DSP's memory that the DSP will fill with a specific value when it is done with it's initializations. It also grabs the addresses of the TxBuffer in the DSP's memory so that communication can take place. Finally, enable the timer (more on this later).

```
//Sync up with DSP before allowing program to continue
error = SIc67_ReadTarget(boardID, region, count, CommReg16, &sync);
if(error)
{
    sprintf(errmsg, "SIc67_ReadTarget() failed with return value %d\nWhile trying to obtain sync value the first time.\n", error);
    SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
    return 0;
}
while(sync != (0x600DC0DE))
{
    Delay(.1);
    //printf("Delay\n");
    error = SIc67_ReadTarget(boardID, region, count, CommReg16, &sync);
    if(error)
    {
        sprintf(errmsg, "SIc67_ReadTarget() failed with return value %d\nWhile trying to obtain sync value.\n", error);
        SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
        return 0;
    }
}
Delay(.1);
//Read CommRegs 14 and 15 to get TxBuffer and TxBufferF Addresses
count = 2;
error = SIc67_ReadTarget(boardID, region, count, CommReg14, buffer);
if(error)
{
    sprintf(errmsg, "SIc67_ReadTarget() failed with return value %d\nWhile trying to obtain TxBuffer Addresses.\n", error);
    SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
    return 0;
}
DSP_TxBuffer_Addr = buffer[0];
DSP_TxBufferF_Addr = buffer[1];

sprintf(errmsg, "DSP Ready");
SetCtrlVal (panel, CONFIG_DSP_STATUS, errmsg);
DisplayPanel (MainPanelHandle);
HidePanel(ConfigPanelHandle);
SetCtrlAttribute (MainPanelHandle, PANEL_TIMER, ATTR_ENABLED, 1);
```

NI LABWINDOWS/CVI

ZONES - ENABLE DSP CALLBACK (3/3)

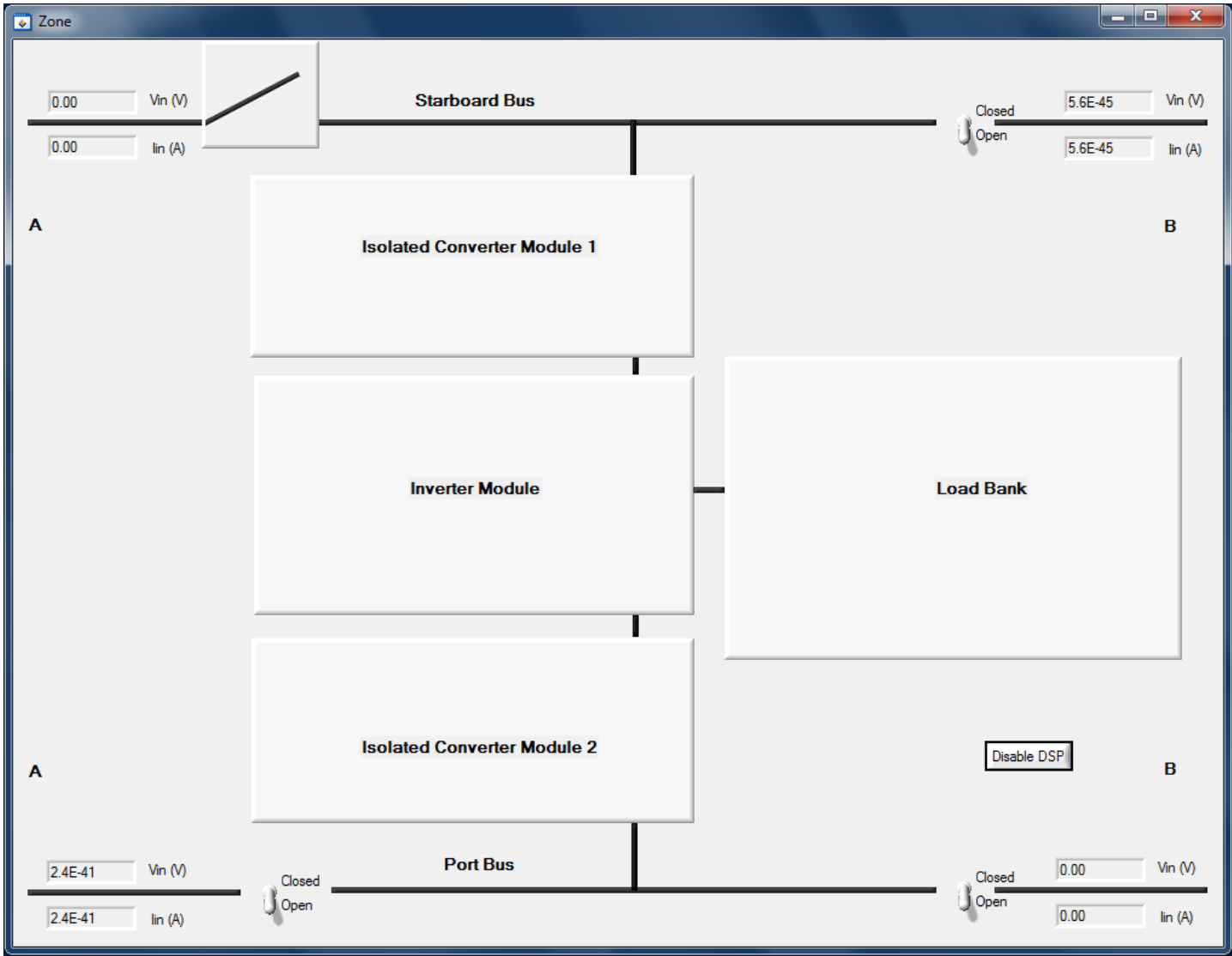
The remainder of the Enable DSP callback is executed if nothing goes wrong in the code in the previous two slides. All that remains is turning on an LED on the front of the DSP Dist. Box with SIc67_SetDigOut(). SIc67_ConfigDaughterCard() sets the parameters of the analog daughter cards to their default settings, essentially “turning the daughter card on.” This function call to configure the analog card unfortunately overwrites the DIO direction setting in the DSP code and so the direction must be set back to what we want it to be for this application – a fix that is being explored currently is to set the Daughter Card parameters individually rather than using a bulk “default” setting that also tampers with the digital I/O direction registers.

```
        //Turn on DSP Status LED on DSP Dist. Unit
        DIO |= DSP_STATUS;
        error = SIc67_SetDigOut(boardID, &DIO);
        if(error)
        {
            printf("SIc67_SetDigOut Failed with error: %d\n", error);
        }
        error = SIc67_ConfigDaughterCard(boardID);
        if(error)
        {
            printf("SIc67_SetDigOut Failed with error: %d\n", error);
        }
        SIc67_SetDIODirection(boardID, 0x2);
    }
    return 0;
}
```

NI LABWINDOWS/CVI

ZONES - MAIN PANEL

After clicking the Enable DSP button, this GUI shown right pops up and will allow the user to control the equipment in the zone by communicating with the DSP about contactor states, and voltage/current references sensor readings, etc.



PURDUE UNIVERSITY

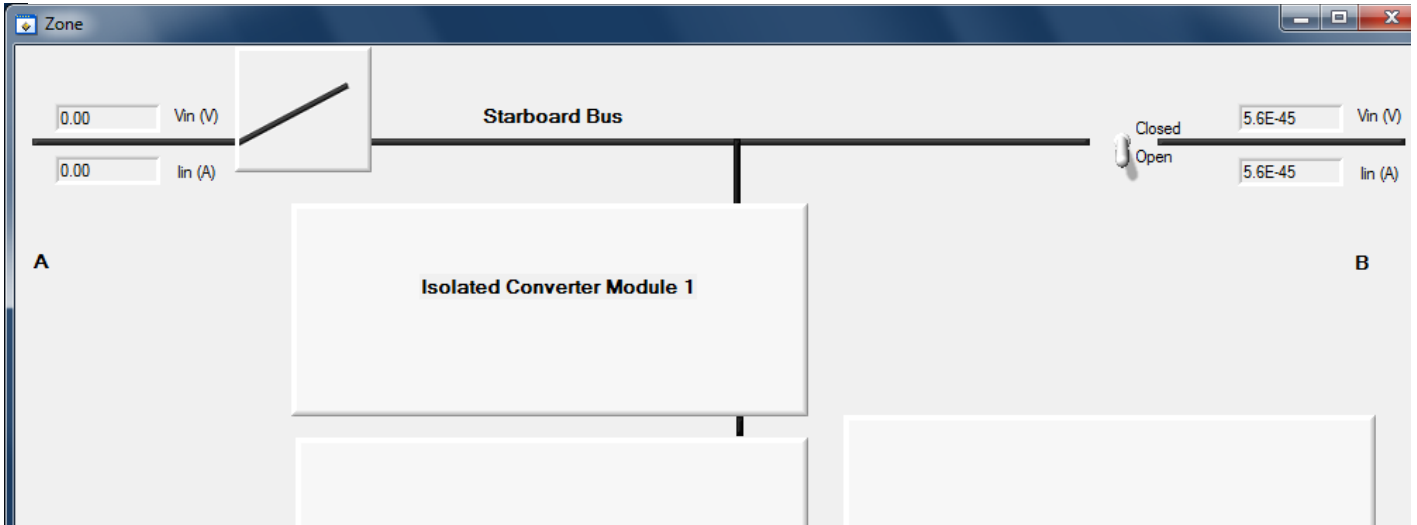
NI LABWINDOWS/CVI

ZONES - A NOTE ABOUT BUTTONS

The button to the left is called a picture ring button. Like the switch on the right, it shows two different pictures depending on the state of the button, but these pictures are user-selectable.

To select which pictures display, double click the picture ring button to bring up its settings, and click on “Image/value pairs”

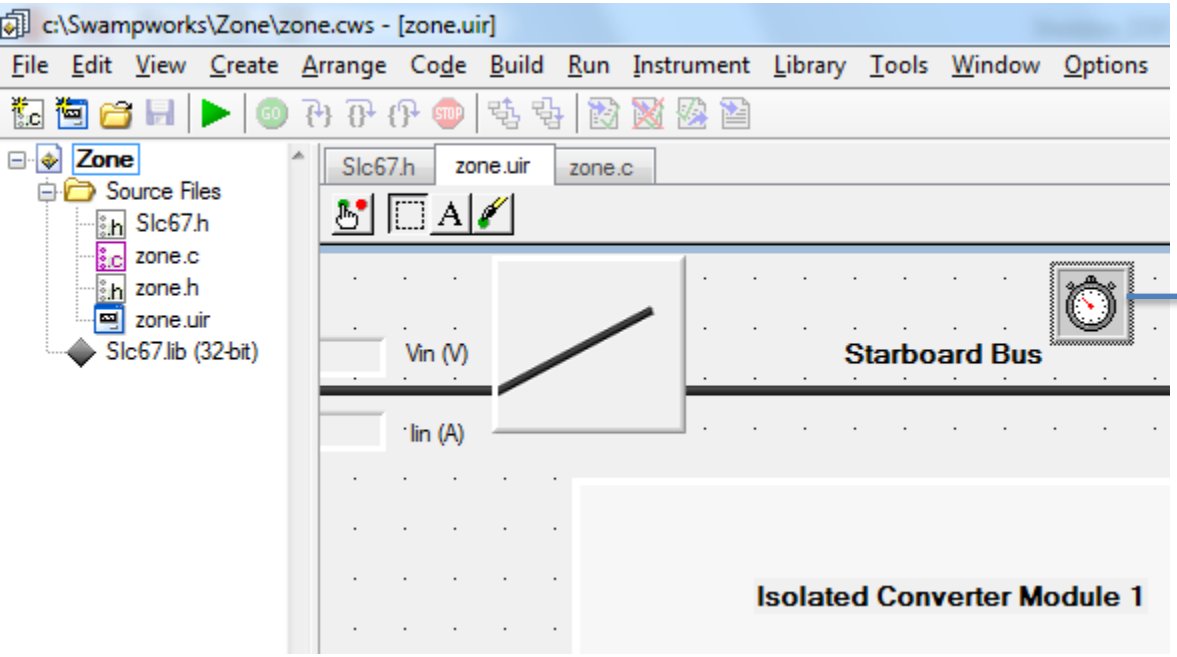
```
int CVICALLBACK ChangePic (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_LEFT_CLICK:
            GetCtrlAttribute (panel, PANEL_SECT_STARB_A, ATTR_CTRL_INDEX, &picIndex);
            if(picIndex){
                SetCtrlAttribute (panel, PANEL_SECT_STARB_A, ATTR_CTRL_INDEX, 0);
                break;
            }
            SetCtrlAttribute (panel, PANEL_SECT_STARB_A, ATTR_CTRL_INDEX, 1);
            break;
    }
    return 0;
}
```



PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ZONES - TIMER



Timer that allows you to execute code on every “Timer Tick.” The interval is set to .2 seconds in this application.

PURDUE UNIVERSITY

NI LABWINDOWS/CVI

ZONES - TIMER CALLBACK (1/2)

All of the number boxes in the GUI need updating. On every timer tick, we grab the list of variables from inside the DSP with a call to SIc67_ReadDSPComm. Then we update all the number boxes.

```
int CVICALLBACK TimerCB (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    char str[256];
    int error;
    UINT32 mode = 2;
    UINT32 count = 50;
    switch (event)
    {
        case EVENT_TIMER_TICK:
            SIc67_ReadDSPComm(boardID, count, DSP_TxBufferF_Addr, (UINT32 *)TxBufferF, mode);
            SIc67_ReadDSPComm(boardID, count, DSP_TxBuffer_Addr, (UINT32 *)TxBuffer, mode);
            SetCtrlVal(panel, PANEL_STARB_A_VIN, DSP_Vin_Starb_A);
            SetCtrlVal(panel, PANEL_STARB_B_VIN, DSP_Vin_Starb_B);
            SetCtrlVal(panel, PANEL_PORT_A_VIN, DSP_Vin_Port_A);
            SetCtrlVal(panel, PANEL_PORT_B_VIN, DSP_Vin_Port_B);

            SetCtrlVal(panel, PANEL_STARB_A_IIN, DSP_Iin_Starb_A);
            SetCtrlVal(panel, PANEL_STARB_B_IIN, DSP_Iin_Starb_B);
            SetCtrlVal(panel, PANEL_PORT_A_IIN, DSP_Iin_Port_A);
            SetCtrlVal(panel, PANEL_PORT_B_IIN, DSP_Iin_Port_B);

            GetCtrlVal(panel, PANEL_SECT_PORT_A, &DSP_sect_port_A);
            GetCtrlVal(panel, PANEL_SECT_PORT_B, &DSP_sect_port_B);
            GetCtrlVal(panel, PANEL_SECT_STARB_A, &DSP_sect_starb_A);
            GetCtrlVal(panel, PANEL_SECT_STARB_B, &DSP_sect_starb_B);

            SetCtrlVal(panel, PANEL_INT_IND_1, DSP_asdf_1);
            SetCtrlVal(panel, PANEL_INT_IND_2, DSP_asdf_2);
            SetCtrlVal(panel, PANEL_INT_IND_3, DSP_asdf_3);
            SetCtrlVal(panel, PANEL_INT_IND_4, DSP_asdf_4);
            SetCtrlVal(panel, PANEL_INT_IND_5, DSP_asdf_5);
```

NI LABWINDOWS/CVI

ZONES - TIMER CALLBACK (2/2)

```
        if(DSP_sect_port_A)
        {
            DIO |= SECT_PORT_A;
        }
        else
        {
            DIO &= (~SECT_PORT_A);
        }

        if(DSP_sect_port_B)
        {
            DIO |= SECT_PORT_B;
        }
        else
        {
            DIO &= (~SECT_PORT_B);
        }

        if(DSP_sect_starb_A)
        {
            DIO |= SECT_STARB_A;
        }
        else
        {
            DIO
                &= (~SECT_STARB_A);
        }

        if(DSP_sect_starb_B)
        {
            DIO |= SECT_STARB_B;
        }
        else
        {
            DIO &= (~SECT_STARB_B);
        }
        error = SIc67_SetDigOut(boardID, &DIO);
        if(error)
        {
             printf("SIc67_SetDigOut Failed with error: %d\n", error);
        }
        break;
    }
    return 0;
}
```

Additionally, we grab the current state of the on/off switches in the GUI to control the contactor coils and tell the DSP to change it's digital outputs based on the buttons.

PURDUE UNIVERSITY

NI LABWINDOWS/CVI

FURTHER INFORMATION

This was certainly not a comprehensive tutorial of LabWindows. For more advanced topics, view Getting Started with LabWindows/CVI from National Instruments

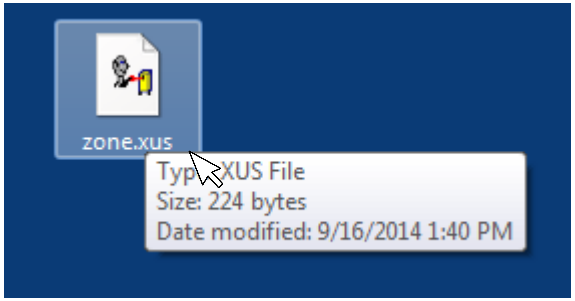
http://www.ni.com/pdf/manuals/373552g.pdf

Additionally, the control panels developed in LabWindows/CVI for the M44 systems are further developed than the zones are currently and serve as a good reference for well-working code.

PURDUE
UNIVERSITY

NI LABWINDOWS/CVI

A NOTE ABOUT THE SIC67 DLL



When you include the SIc67 DLL in your project to communicate with the DSP, the .exe that Labwindows creates when you build the project now requires administrator privileges.

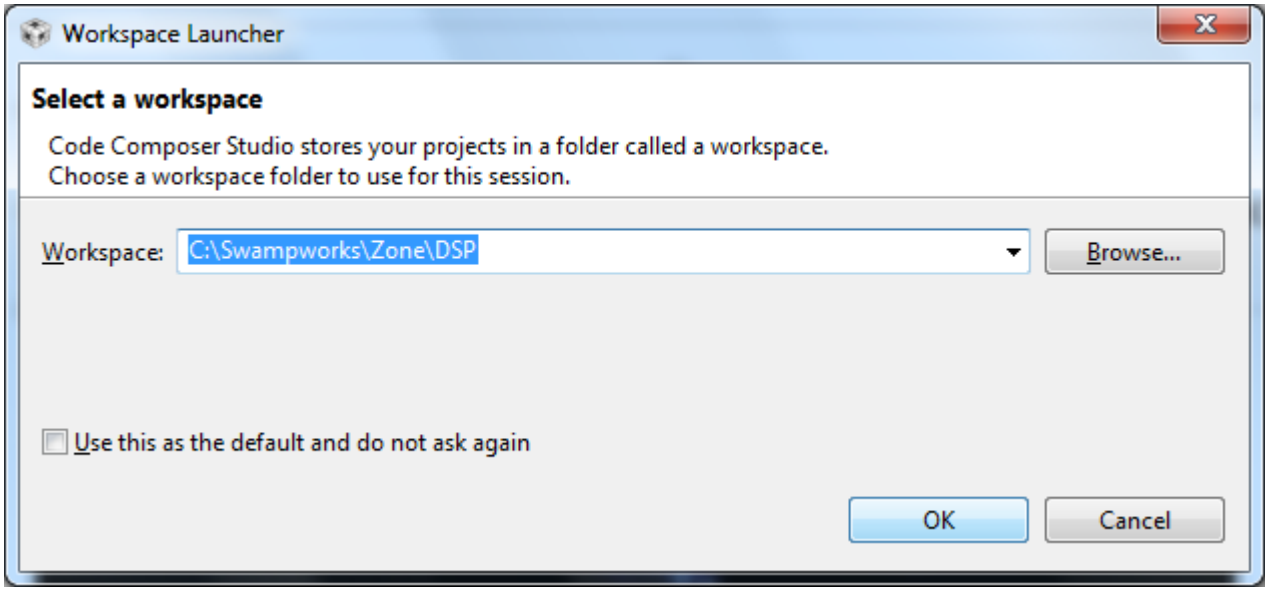
To allow those on powerlab accounts to run the executables, someone with admin password needs to make a .xus file as shown left, which will allow any executable with the filename you specify to run without admin privileges.

PURDUE
UNIVERSITY

TI CODE COMPOSER STUDIO

STARTING CCS

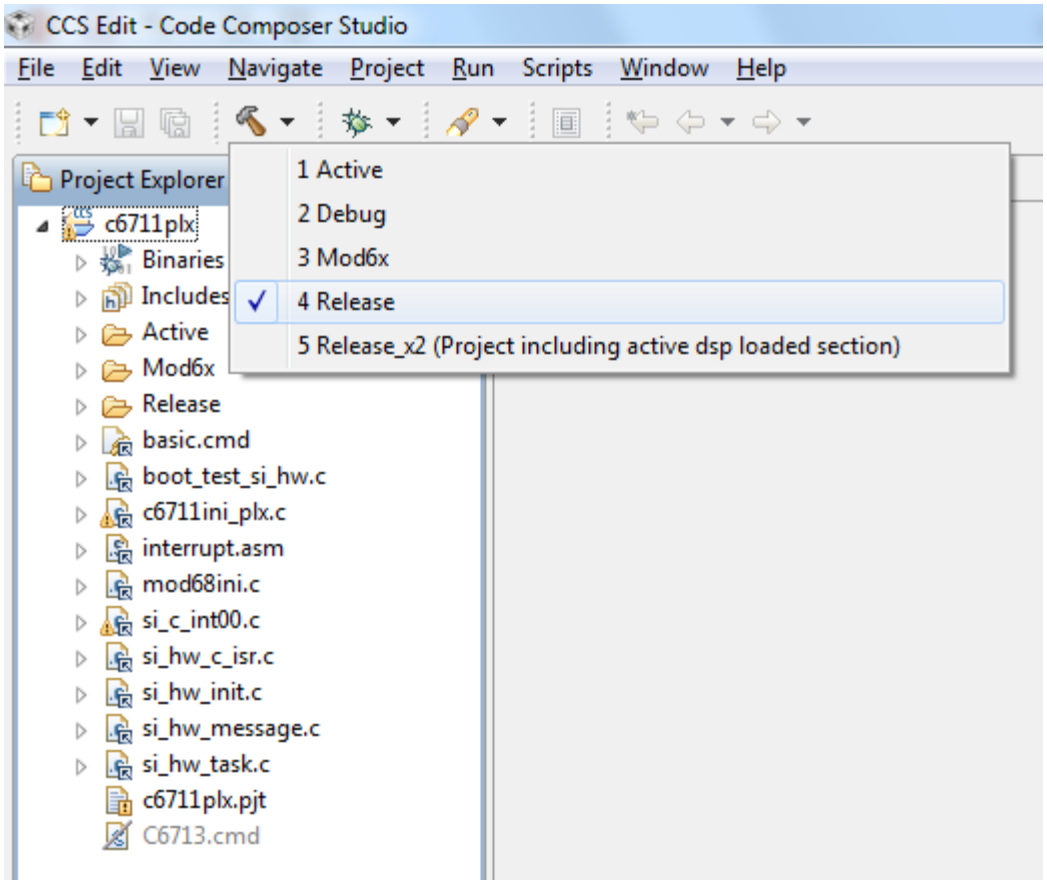
1. Open Code Composer Studio
2. It will prompt you to choose a directory for your workspace



PURDUE UNIVERSITY

TI CODE COMPOSER STUDIO

OVERVIEW OF WORKSPACE



The default project from Sheldon Instruments is shown in the project tree (left). There are 8 C files. The only one that needs to be modified is "c6711ini_plx.c," all the others are for functionality we don't need to alter *.

Make sure the project is set to build in "Release" mode in the build (hammer) drop down menu.

When you are satisfied with your code, click the hammer to build it. The "coff" file (filename.out) will be put into the Release directory.

PURDUE UNIVERSITY

*For more information on the other 7 C files, see document sidsp_api_supplement.rtf

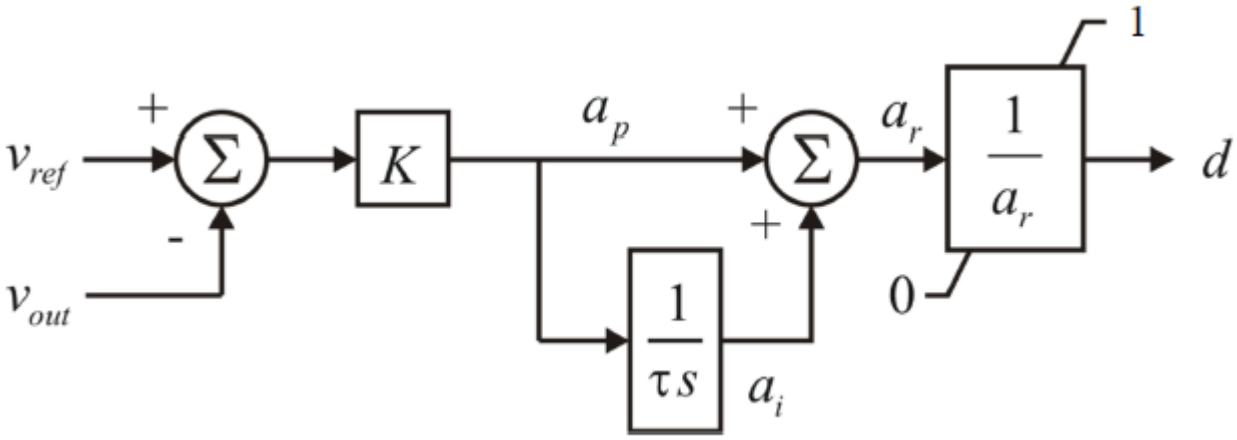
TI CODE COMPOSER STUDIO

EXAMPLE SYSTEM TO CONTROL

Example

Let's say we wanted to control a DC-DC converter with the following control block diagram.

We need to measure the output voltage of the converter, and run a control loop.



PURDUE UNIVERSITY

TI CODE COMPOSER STUDIO

THE MAIN FUNCTION - INITIALIZATIONS

Lines 160 – 182: Here we configure the system with the function SI_HW_InitDSP6711(), and configure the timer interrupt to run at 10 kHz, add the ISR and enable it. Additionally, set direction of digital I/O.

```
158 void main (void)
159 {
160     UINT32 hBeat = 0;
161     UINT32 cnt = 0;
162     register UINT32 hBeat_stall = 0;    //register used for debug jtag builds
163     UINT32 msg, msgCnt = 0;
164     UINT32 reg = 0x0;
165
166     //finishes system initialization
167     //this code and access to it must be kept in internal
168     //memory or else it will not be reachable
169     SI_HW_InitDSP6711();
170
171     //Configure Timer0 and Interrupt for it
172     mWriteRegister(TIMER0_PRD, 0x1D4C);
173     mWriteRegister(TIMER0_COUNT, 0);
174     mWriteRegister(TIMER0_CTRL, 0x2c0);
175     SI_HW_Intr_AddISR(Timer_Interrupt, TINT0, 10);  // TIMER0
176     mInterrupt_Enable(10);
177
178     //Set DIO direction: least sig 16 input, most sig 16 output
179     reg = mReadRegister(kDSPAddr_dDIO_CSR);
180     reg &= 0xfffffffC;  //Make last two bits 0
181     reg |= 0x00000002;
182     mWriteRegister(kDSPAddr_dDIO_CSR, reg);
183
184     mWriteRegister(kFPGACommReg14, (UINT32)&TxBuffer[0]);   //Letting host know where TxBuffer and TxBufferF are in DSP memory
185     mWriteRegister(kFPGACommReg15, (UINT32)&TxBufferF[0]);
186
187     mWriteRegister(kFPGACommReg16, 0x600DC0DE); //Indicate to host that DSP is ready
188
```

Lines 184 – 187:
Put the address of the TxBuffers in Communication Registers 14 & 15 so the host can grab them and know which addresses the variables of interest may be found at. The last step is to put a certain hex value into Communication Register 16 – 0x600DC0DE (“Good Code”) – to sync up with host.

TI CODE COMPOSER STUDIO

THE MAIN FUNCTION – INFINITE WHILE LOOP

Now is one of the few times in your life that you actually do want a while(1) loop. Don't worry about the inability to get out of the while loop; when you click the Disable DSP button in the GUI, the DSP will be held in reset externally until you click the Enable DSP button to download a new coff file, and allow the code to be executed.

The important component of the infinite while loop is the filling of the TxBuffer with variables of interest. The host will grab them with a function call to SIc67_ReadDSPComm() in the LabWindows code. The rest has to do with the default heartbeat message variables that were set up by Sheldon Instruments in their factory CCS project – we could remove them.

```
189     while (1)
190     {
191         cnt++;
192         if(cnt == 50000)
193         {
194             cnt = 0;
195         }
196         if ( ++hBeat_stall == 0xFFFF ) {
197             hBeat++;
198             mHBeat = hBeat;
199
200             hBeat_stall = 0;
201         }
202
203         TxBufferF[0] = sect_ibus_starb_B;
204         TxBufferF[1] = sect_ibus_starb_A;
205         TxBufferF[2] = sect_ibus_port_B;
206         TxBufferF[3] = sect_ibus_port_A;
207         TxBufferF[4] = sect_vbus_starb_B;
208         TxBufferF[5] = sect_vbus_starb_A;
209         TxBufferF[6] = sect_vbus_port_B;
210         TxBufferF[7] = sect_vbus_port_A;
211
212         if(gMsg)
213         {
214             msgCnt++;
215             msg = gMsg;
216
217             //clear message
218             gMsg = 0;
219
220             while(SI_HW_MessageSend(-1, msgCnt))
221             {
222             }
223
224             if(msgCnt == msg)
225                 msgCnt = 0;
226         }
227     }
228 }
```

PURDUE UNIVERSITY

TI CODE COMPOSER STUDIO

EXAMPLE INTERRUPT SERVICE ROUTINE CODE

The interrupt service routine (ISR) shown right shows how to do basic analog inputs, digital inputs as well as analog outputs and digital outputs. Always read the inputs IN THE VERY BEGINNING of the ISR. This ensures regular sample periods. This ISR is a way to implement the control diagram shown earlier.

Reminder: Don't forget to reset the interrupt flag!

PURDUE UNIVERSITY

```
////////////////////////////////////////////////////////////////////////////////
// interrupt void Timer_Interrupt(void)
//
// Description:
// This is just a test code to demonstrate how to install another ISR.

#pragma CODE_SECTION(Timer_Interrupt, "SECTISR");
interrupt void Timer_Interrupt(void)
{
    //Read in analog and digital variables first thing
    AIN_0 = readADC(0);
    dig_in_word = readDIG;

    //Reset ISR flag
    ICR = (1 << 4);

    //Convert integer measurement to float
    v_meas = ((float)(AIN_0))/3276.7 + offset;

    //low-pass filter measured voltage
    pv_lpf = (v_lpf - v_meas)/tau_filt;
    v_lpf += pv_lpf*deltat;

    //PI Controller
    v_error = v_ref - v_lpf;
    ap = K*v_error;
    ai_int = ai_int + ap*deltat;
    ai = ai_int/tau_control;
    ar = ap + ai;

    //duty cycle
    if(ar >= 1.0) d = 1/ar;
    else if(ar >= 0.0) d = 1;
    else d = 0;

    //Output DAC and digital
    outpDAC(0, (int)(d*3276.7));
    outpDIO(dig_out_word);
}
```

TI CODE COMPOSER STUDIO

ANALOG INPUT AND OUTPUT SCALING

The analog hardware doesn't accept floating point format directly; there is a conversion that must take place beforehand. This is achieved by casting in C, either casting to int or casting to float. Values input into outpDAC() macro must be of int data type.
The readADC() macro returns an int.

Voltage at pins	Converted to int
10.0 V	0xDDDD7FFF
5.0 V	0xDDDD3FFF
0.0 V	0xDDDD0000
-5.0 V	0xDDDDC000
-10 V	0xDDDD8000

Mask off most sig upper 16 bits

```
//Macros
#define readADC(chan) (0x0000ffff & (mReadRegister(kDSPAddr_dADC + chan*0x4)))
#define outpDAC(chan, val) (mWriteRegister(kDSPAddr_dDAC + chan*0x4, val))


//Read in analog and digital variables first thing
AIN_0 = readADC(0);
//Convert integer measurement to float
v_meas = ((float)(AIN_0))/3276.7 + offset;
outpDAC(0, (int)(d*3276.7));
```

$2^{16} = 65536$, divided into +/- halves, bit 16 is sign bit
0x7FFF → 32767
0x8000 → -32768 (Think 0xFFFF – 0x7FFF = 0x8000)

Note: The 16 most significant bits don't matter (represented by D's) when you declare AIN_0 as a short int. The DSP will only acknowledge the 16 least significant bits. Both ADC and DAC are 16-bit analog hardware. We are used to the M44s which pack two 16-bit analog values into a single 32-bit word.

PURDUE UNIVERSITY

GOOD PRACTICE

- We want to preserve these DSPs as long as we can
- Test your circuit to be sure that it is not going to have any voltage higher than the +/- 10V that it is rated for on the analog side, and the 3.3V on the digital side
- You can use bidirectional TVS diodes to clamp voltage waveforms past +/- 10V
- Spice the circuit!
- Don’t drive LEDs directly from the output of the DSP – they can’t provide that much current without damaging the outputs – use a buffer.
- Use isolators where possible on logic signals

PURDUE
UNIVERSITY

FOR MORE INFORMATION

See these docments in the C:\SIC67DSP-Sidev folder on all the new lab PCs with a Sheldon Instruments DSP

Hardware Capabilities:

- Sheldon Instruments' pdf manual of the DSP system - SIC671xPCI_r1b.pdf
- Sheldon Instruments' pdf manual for the mod66 daughter card - mod66xx_R02.pdf

Software:

- Using the sisample utility provided by Sheldon Instruments - gettingstarted.pdf
- Documentation for code used on DSP side in CCS - sidsp_api_supplement.rtf
- Documentation for SIc67 DLL code - siddk_api_supplement.rtf
- Anything in the C:\SIC67DSP-SIdev directory related to this DSP card (there is information for other DSP cards from Sheldon Instruments in this folder!)

PURDUE
UNIVERSITY