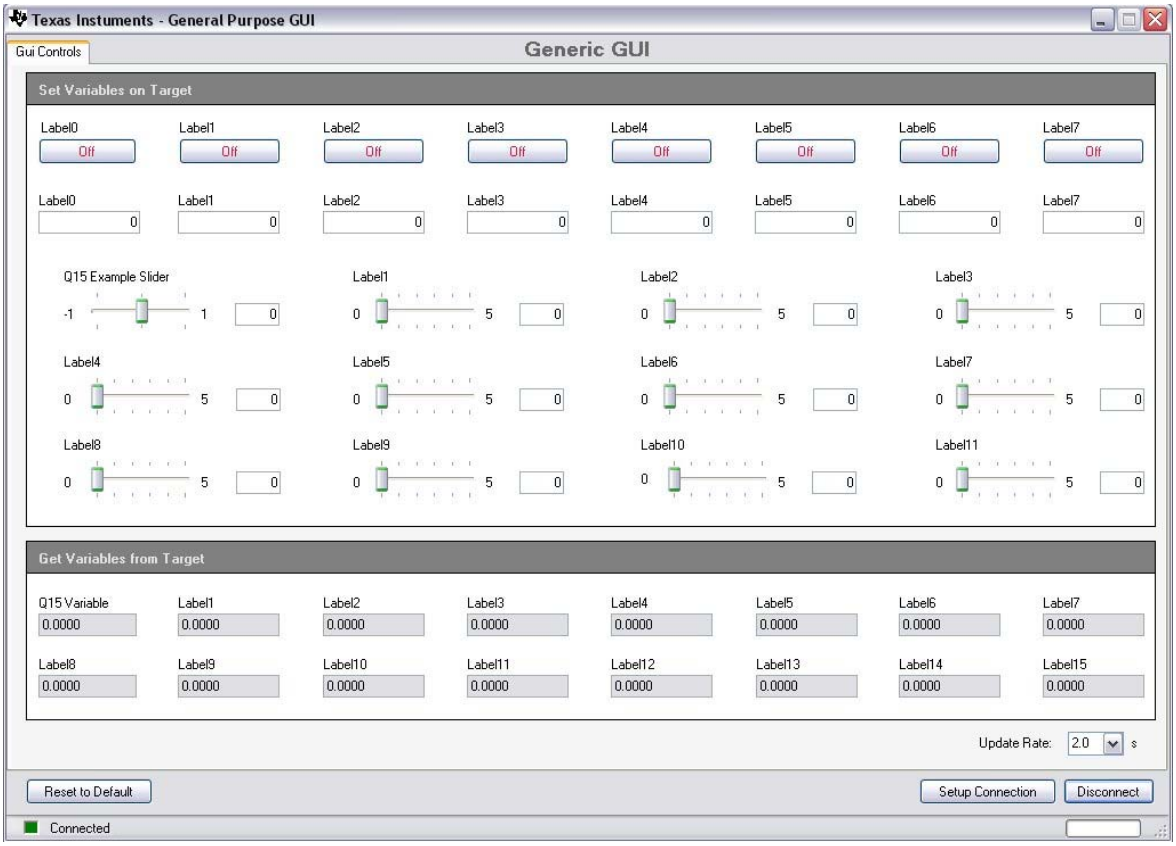
General Purpose GUI Overview

May 2008 – Revised July 2008

The Generic GUI is an easy to use tool designed to simplify demos by allowing the user to graphically edit program variables on the fly. With usability in mind, the GUI was developed to only need a small amount of time and additional coding to interface with a Code Composer Studio (CCS) project. From a functionality perspective the GUI itself works much like a CCS watch window in that numbers are shown with a QValue, initial value, and minimum and maximum value limits. The Generic GUI is freeware built with Microsoft Visual Studio.NET and written in C#. Overall the GUI provides the following features:

- Minimal effort and coding needed to use the GUI with any project
- Ability to boot the F28xxx from the SCI (RS-232) with a .a00 file, connect to an emulator controlled DSC, or a flashed DSC
- Create a bootable .a00 file eliminating the need to start up Code Composer Studio.
- Customizable GUI controls
- Flexibility to give any controls a specific QValue, an initial value, or a set of minimum and maximum values

Note that the General Purpose GUI requires Microsoft .NET framework 2.0 or higher to run. Please ensure that this software is installed prior to running this program.



Getting Started

Hardware Considerations

In order to connect the hardware to an external GUI a four-pin SCI to Serialport connection is needed. This cable is available out-of-the-box in many of the major EVM packages. See the SCI-Serialport_Assembly file for information on how to build your own. This is located at: *C:\TI_F28xxx_SysHW*

Assuming a board is able to run with Code Composer Studio, the GeneralPurposeGUI should be able to connect to the project from a hardware perspective without any other additional hardware consideration. However, to ensure the program can boot to the target board the following must be true:

- For 280xx targets boot pins GPIO18 and GPIO29 must be pulled high and GPIO34 must be pulled low.
- For 283xx targets boot pins GPIO85, GPIO86, and GPIO87 must be pulled high and GPIO84 must be pulled low.

NOTE: for customers using a controlCARD all of the mentioned GPIOs are weakly pulled high so that the DSC will boot from flash. Therefore to ensure SCI (RS-232) can boot the target DSC only GPIO34 and/or GPIO84 will need to be pulled low by approximately a 4.7kΩ resistor. All TI F28xxx EVM boards allow a jumper to be placed so that SCI boot capabilities are given without any external components needed.

See the **F280xxBootROM UserGuide** or **F283xxBootROM UserGuide** for further information.

Allowing a CCS Project to connect to the GUI

Only a few steps are necessary to allow the GUI to connect to any CCS project. The steps are:

1) Please ensure that GPIO-28 and GPIO-29 are set to SCI-RX and SCI-TX functionality. In the TI_28xxx_SysSW framework, this is found in [ProjectName]-DevInit_[Target]
2) Add SciCommsGui.c to your project.

   In the project window, right-click on your project, select “Add files to project”, then browse to find SciCommsGui.c. If you are using the system framework the path will be

   ```
   C:\TI_28xxx_SysSw\~SupportFiles\source\SciCommsGui.c
   ```

3) Add `SCIA_Init()` and `SerialHostComms()` to your function prototype list
4) Find/Set a CpuTimer that is running at approximately 1ms (ie. `CpuTimer0Regs.PRD.all = mSec1;`)
5) Add SerialCommsTimer as an int16 if it does not already exist.

Find the following code:

```
int16  VTimer0[4];   // Virtual Timers slaved off CPU Timer 0 (A events)
int16  VTimer1[4];   // Virtual Timers slaved off CPU Timer 1 (B events)
int16  VTimer2[4];   // Virtual Timers slaved off CPU Timer 2 (C events)
```

then add the line (if it does not already exist):

```
int16 SerialCommsTimer;
```

6) Declare the following variables if they do not already exist

```
//GUI support variables
// sets a limit on the amount of external GUI controls - increase as necessary
int16 *varSetTxtList[16];        //16 textbox controlled variables
int16 *varSetBtnList[16];        //16 button controlled variables
int16 *varSetSldrList[16];       //16 slider controlled variables
int16 *varGetList[16];           //16 variables sendable to GUI
int16 *arrayGetList[16];         //16 arrays sendable to GUI
```

7) Add the following to the project's initialization sequence:

```
SCIA_Init();

//"Set" variables
//---------------------------------------
// assign GUI variable Textboxes to desired "setable" parameter addresses
//varSetTxtList[0] = &Var;

// assign GUI Buttons to desired flag addresses
//varSetBtnList[0] = &Var;

// assign GUI Sliders to desired "setable" parameter addresses
//varSetSldrList[0] = &Var;


//"Get" variables
//---------------------------------------
// assign a GUI "getable" parameter address
//varGetList[0] = &Var;
```

8) Add `SerialCommsTimer++;` to the "Loop rate synchronizer" of the CpuTimer chosen in step 3
If Using CpuTimer0 to run at 1ms the code should look like:

```
void A0(void)
{
      // loop rate synchronizer for A-tasks
      if(CpuTimer0Regs.TCR.bit.TIF == 1)
      {
            CpuTimer0Regs.TCR.bit.TIF = 1;      // clear flag
            //-----------------------------------------------------------
            (*A_Task_Ptr)();        // jump to an A Task (A1,A2,A3,...)
            //-----------------------------------------------------------
            VTimer0[0]++;           // virtual timer 0, instance 0 (spare)
            SerialCommsTimer++;     // used by DSP280x_SciCommsGui.c
      }
      Alpha_State_Ptr = &B0;        // Comment out to allow only A tasks
}
```

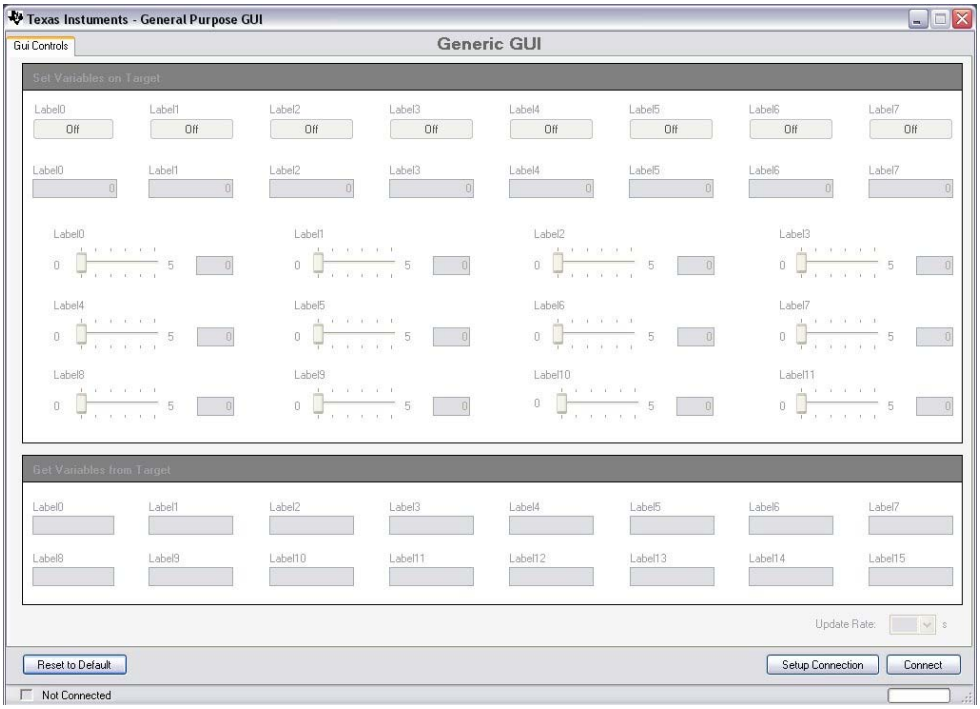
For example, if you are using CpuTimer1 running at 1ms add `SerialCommsTimer++;` to the B0 state.

9) In one of the tasks in the A,B, or C state (whichever represents the CpuTimer you chose in step 4) add the following line of code:
`SerialHostComms();`
10) Edit the code in Step 7 by changing &Var to a variable in the project which will connect to the control type specified, and then adding additional controls to the connection array as necessary. For example, to make a variable editable by a slider in the GUI, make it an element of the varSetSldrList array. The element chosen in the array corresponds to which slider will control the variable inserted into the array. Repeat this step for all variables that you would like the GUI to control.
11) After this your project will be able to connect to the GUI.

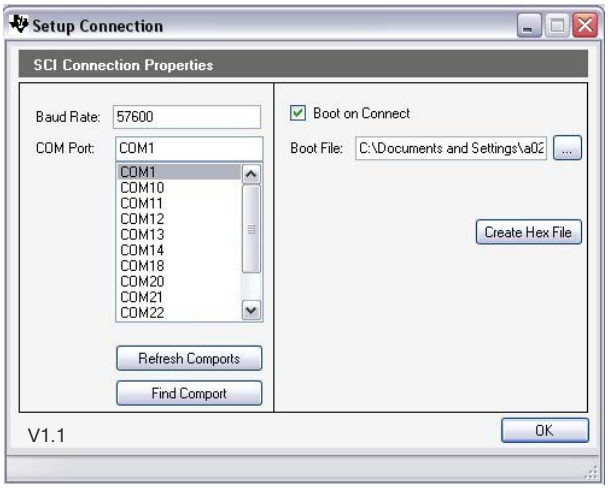
Running the Application

The GUI can connect to the hardware by booting a controlCARD with an .a00 file, by connecting to a flashed controlCARD target, or by connecting to a controlCARD running from RAM via an emulator. This guide discusses booting a .a00 file to the controlCARD.

1) Browse to `C:\TI_F28xxx_SysSW\GeneralPurposeGUI\` and double-click on GeneralPurposeGUI.exe. If the GUI does not open please ensure that Microsoft .NET Framework 2.0 is installed on your computer. The .NET framework is required for this program to run and is often installed through Microsoft Windows updates.



2) Click “Setup Connection” on the GUI



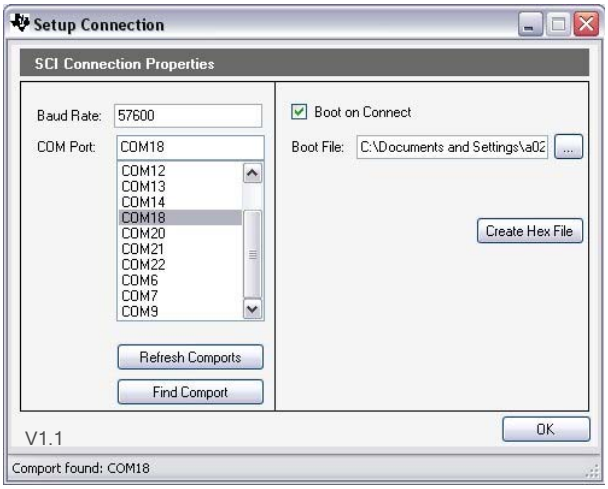
3) Ensure the Baud Rate is set to 57600

4) Next you will need to select your serial comport.
   a. If the comport that the target is connected to is known please select it.
   b. Otherwise, disconnect the emulator from the target board (if applicable). Next, click “Find Comport” then follow the instructions at the bottom of the window. (Note: the board’s main power is controlled by SW1) This will run through a short automated test to find the COM port that is connected to the EVM board. Once complete you should see “Comport Found: COMXX” near the bottom of the window. Once the comport is found you may turn the EVM board off and reconnect the emulator.
   c. If the GUI is still unable to find a valid comport after fixing/checking all errors received and then retrying, manually find the by going to:

      `Control Panel->System->Hardware tab->Device Manager->Ports(COM & LPT)`

      If using a serial port directly connected to a PC, look for a comport which shows up as “Communications Port” and select this comport in the Setup Connection window. If using a USB to Serial adapter look for the com port which shows “USB-to-Serial Bridge”, then select this comport in the Setup Connection window.

If the GUI is still unable to find a valid comport after fixing/checking all errors received and then retrying, manually find the by going to:
Control Panel->System->Hardware tab->Device Manager->Ports(COM & LPT).
If using a serial port directly connected to the PC, look for the com port which shows up as “Communications Port” and select this comport in the Setup Connection window. If using a USB to Serial adapter look for the com port which shows “USB-to-Serial Bridge”, then select this comport in the Setup Connection window.



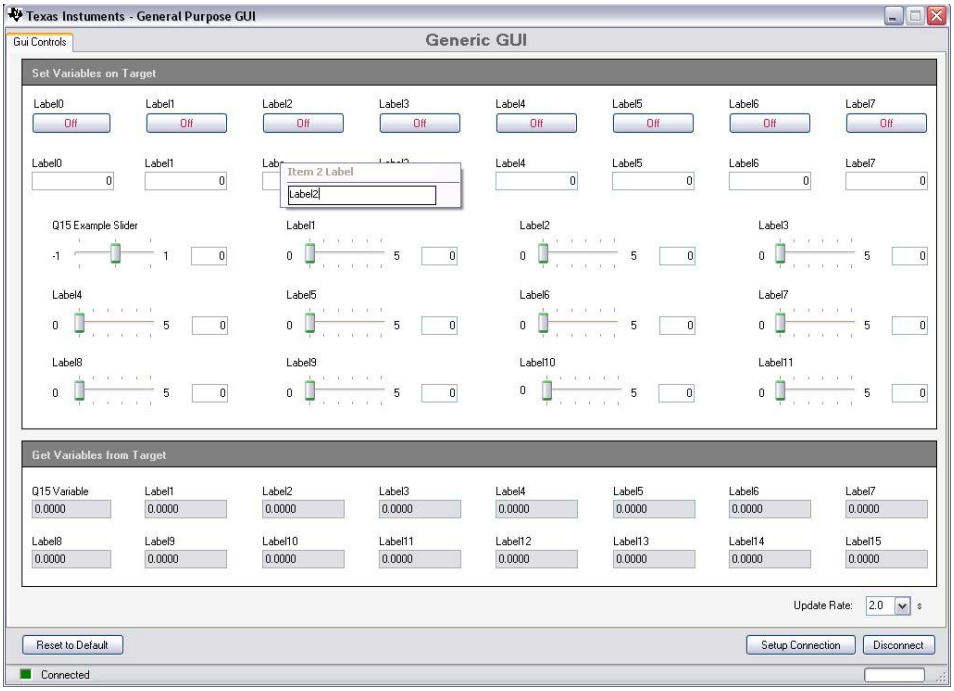
5) Check “Boot on Connect” to boot an .a00 file to the target F28xxx. To use the GUI in conjunction a project running from RAM in Code Composer Studio, please uncheck the checkbox and ensure the code is running.
6) If booting the F28xxx target from the SCI either create an .a00 file or choose one to be loaded. An .a00 file is a hexadecimal conversion of an .out file and is created with the Hex2000.exe utility. Please do one of the below to create this file.
   - Click “Create Hex File” to create the necessary .a00 file. Select the path to the Hex File Utility by browsing to `C:\CCStudio_v3.3\C2000\cgtools\bin\hex2000.exe` then click “Open”. Next, browse to find a valid .out file created by the project.

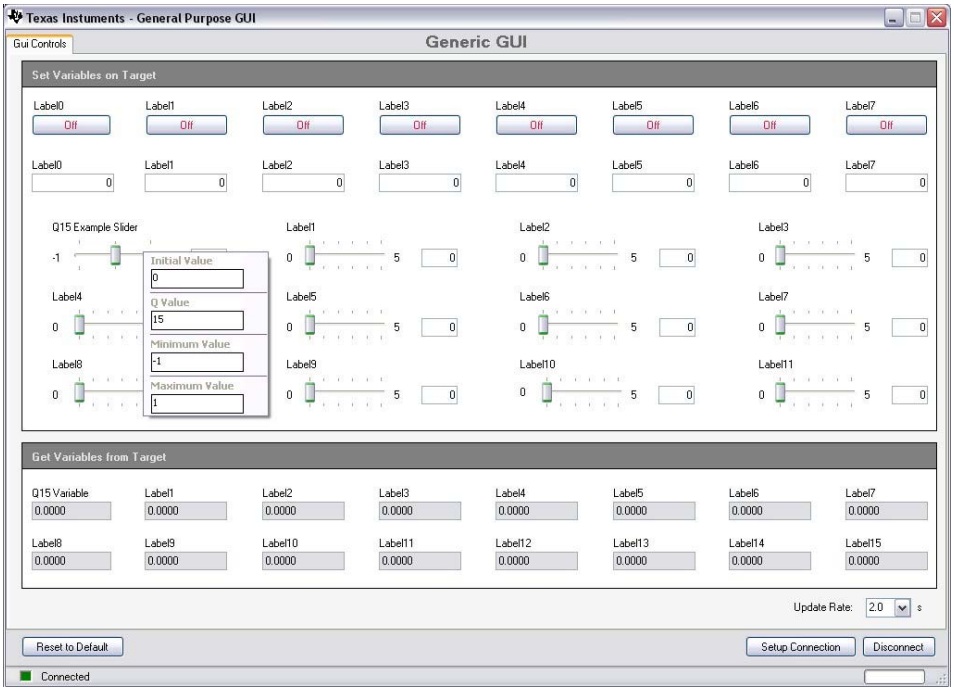
- Click "…" to select an .a00 boot file.

Please see **TMS320x2833x Boot ROM Reference Guide** for more information on this topic. Click "OK"

7) On the Main Window click "Connect". When asked to power cycle the board, turn the main power off and then back on.
8) When complete press "Connect" again when the GUI asks. A progress bar in the bottom right hand corner of the main window will show the progress of the boot procedure. When the boot procedure is finished the program will connect to the target. Please note that booting to the DSC will not write the program into FLASH.
9) Customize the GUI by right-clicking on controls:
   - The first textbox corresponds to varSetTxtList[0] in the project, the second textbox to varSetTxtList[1], and so on
   - The first slider corresponds to varSetSldrList[0], the first
   - Set Variables:
     - Right-clicking on a label changes its name
     - Right-clicking on a button changes its initial value
     - Right-clicking on a textbox changes its Qvalue and initial value
     - Right-clicking on a slider changes its Qvalue, initial value as well as minimum and maximum values allowed for the control
   - Get Variables
     - Right-clicking on a label changes its name
     - Right-clicking on a textbox changes its Qvalue

   NOTE: To edit the configuration of the GUI without needing to connect to the GUI. The file found at `C:\TI_F28xxx_SysSW\GeneralPurposeGUI\Settings.txt` can be edited as desired and will make the necessary changes to the GUI.





References

For more information please see the following guides:

F28xxx User's Guides –
http://www.ti.com/f28xuserguides

F280xx Header Files and Peripheral Examples –
http://www-s.ti.com/sc/techzip/sprc191.zip

F283xx Header Files and Peripheral Examples –

http://focus.ti.com/docs/toolsw/folders/print/sprc530.html