**Remote Debugging for Visual Studio 2010**

***Introduction:***

*This document describes how to remotely debug the C# VisionPro application that runs on Unilever’s Popsicle machines. Both the local and remote machines are running the 64 bit version of Windows7. Visual Studio 2010 runs on the local machine while the application runs stand alone on the remote machine. The debug procedures outlined in this document should be able to be used for setting up remote debugging of any installed application running on a remote machine(providing a PDB symbol file exists for the installed application). The firewall should be turned off on both machines. For testing, an RJ-45 crossover cable was used to connect the two computers. The Host IP address was set at 192.168.142.252 (Local Area Connection) and the Remote IP address was set at 192.168.142.82 (Local Area Connection).*

**Create user accounts (See Figure 1.)**

* Go to **Control Panel**.
* Select **Administration Tools**.
* Select **Computer Management**.
* Open the **Local Users and Groups**.
* Right click in the middle pane and select ***New User***.
* Enter **RemoteDebug** for the user name.
* Enter **remote** (lower case) for the password and create the user. Follow the same procedure on the remote machine and create an identical user with an identical password. *This is a trick so that Visual Studio can authenticate the connection and also allows connections to machines that are on different domains. Each of these local accounts must be set to Administrator level.*

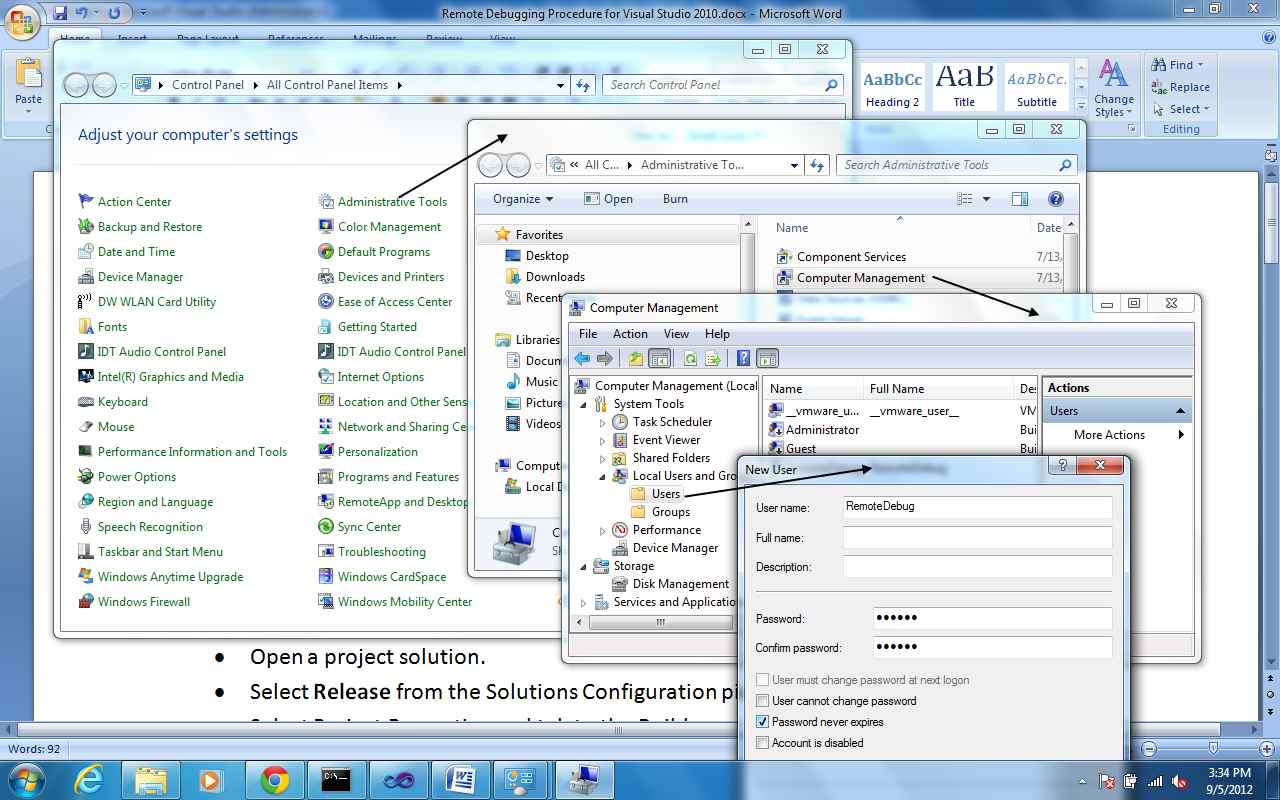


Figure 1. Create the RemoteDebug user account.

* Next, open the desired project solution in Visual Studio.
* Select **Release** from the Solutions Configuration pick list.
* Select Project Properties and tab to the **Build** screen. See Figure 2.
* Click Advanced and change Debug Info to **pdb-only**.
* Select **Build>Rebuild Solution***. This will rebuild the Solution and update the PDB Symbol file which will match it to the exe file in the target output release folder. These matching files must exist on the remote machine in the installed directory of the application. The PDB file is critical for remote debugging, it contains all the debug information for breakpoints and allows us to single step through applications. If the PDB file does not exist in the install folder on the remote machine, copy and paste it in. The PDB file has been included in the installer for the Popsicle machines so they should be ready for remote debug.*

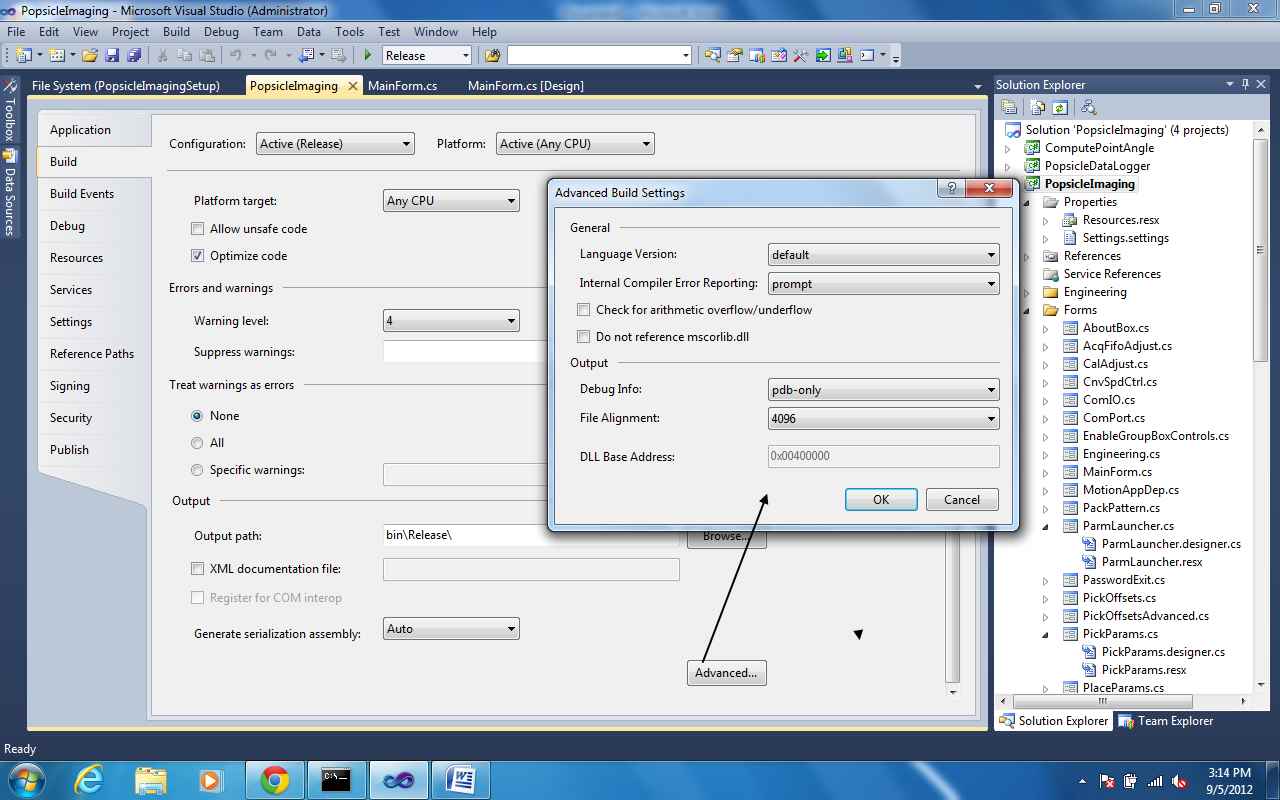


Figure 2. Advanced Build Options.

* The remote machine will need to be setup. In the local machine, goto Start>All Programs>Microsoft Visual Studio 2010>Visual Studio Tools>Visual Studio 2010 Remote Debugger Folder. Copy the x64 folder over to the desktop of the remote computer.
* Open the x64 folder and double click on the msvsmon file. For first time use, the firewall should be automatically configured to allow the connections. If not, refer to the link <http://msdn.microsoft.com/en-us/library/ee126350.aspx> for details on which ports should be opened.

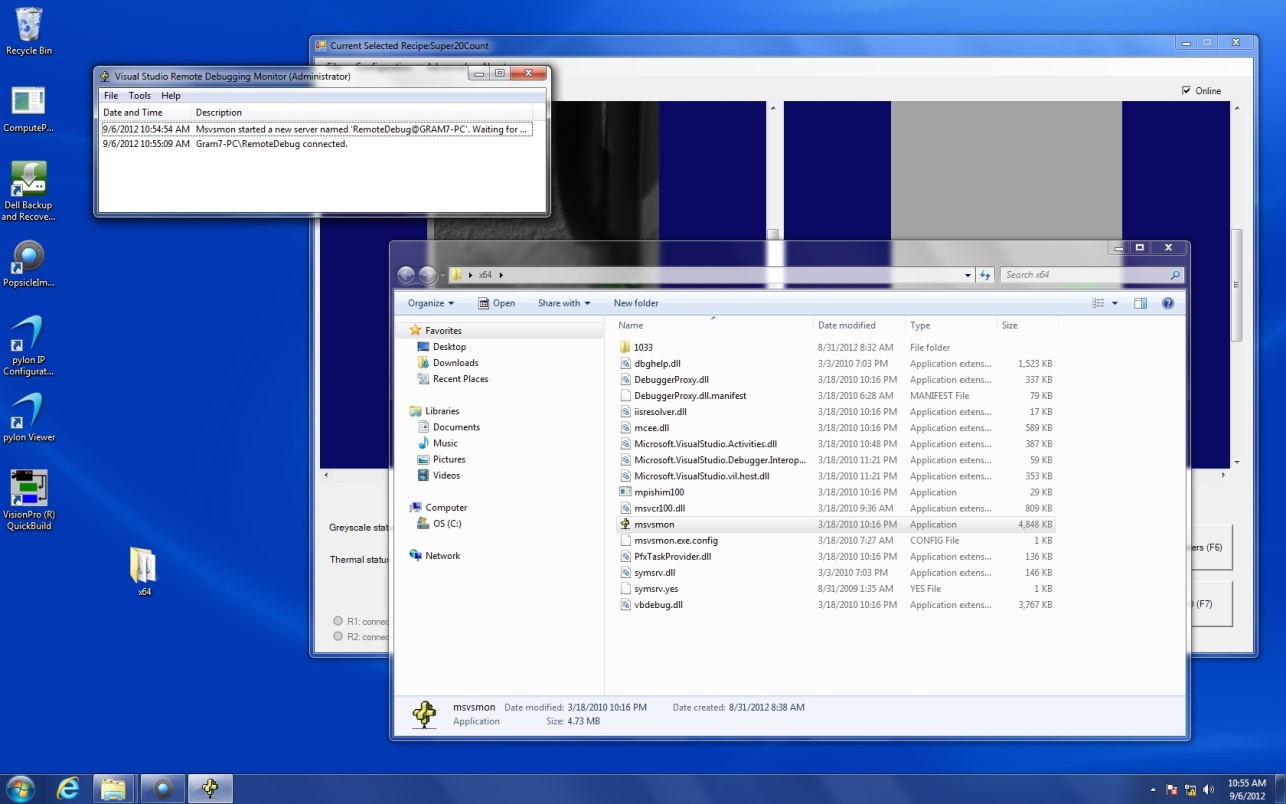


Figure 3. Remote PC Screenshot.

* Refer to Figure 3. Notice that the **msvsmon** debugger app is running in the upper left part of the screen. It shows that a connection is established with Visual Studio that is running on the local host machine. *This will occur when the process to debug is attached as explained in the next section.*
* Next, from Visual Studio, select **Tools>Attach to Process**. See figure 4.

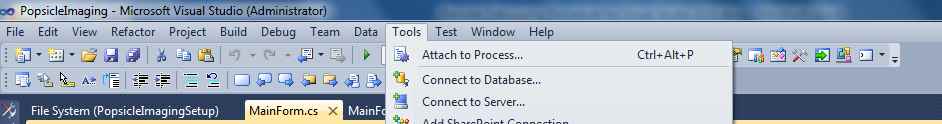


Figure 4. Selecting Attach to Process.

* Figure 5 shows the running processes on the remote machine*. Note: The user name and password must have already been setup on the remote machine and the user account must be logged in and running on the remote machine.*
* Notice the Qualifier field has the name RemoteDebug@GRAM7-PC. In this example, the GRAM7-PC computer name was selected after pressing the Browse button. *Note: It may take 2 or 3 times for the remote PC to show up.*

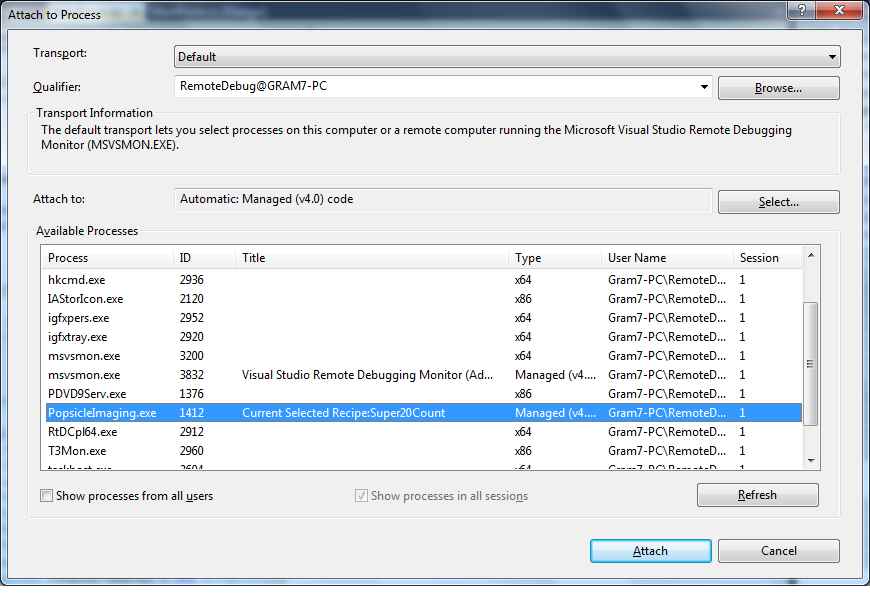


Figure 5. Active Running processes on the remote machine.

* Next, select your executable file. In this example, PopsicleImaging.exe is highlighted.
* Click on the **Attach** button. You should be able to set breakpoints and debug your application that is installed on the remote machine.

***Additional information***

* Figure 6 shows the path to the Modules window that is available while the debug session is running.
* Select Debug>Windows>Modules to bring up the Modules window.
* It is convenient to create a shared folder on the remote machine and map it. That way files can be transferred back and forth easily.

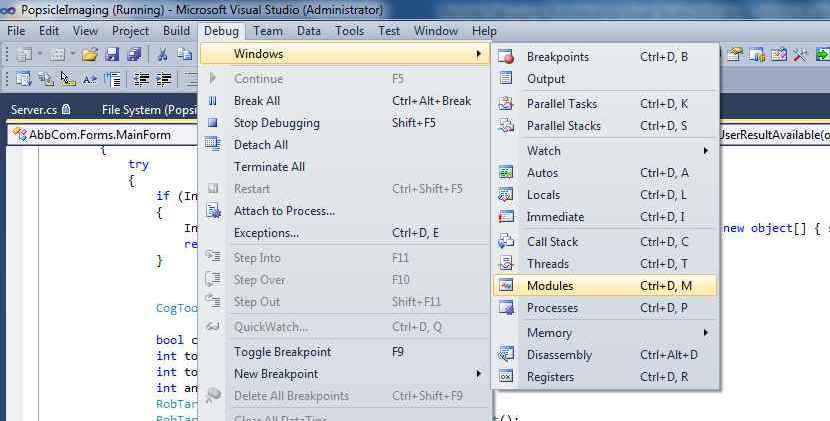


Figure 6. Selecting the Modules Window.

* Figure 7 shows the Modules window. Notice the Symbol Status shows Symbols Loaded for the PopsicleImaging.exe file. You can also view the paths where the Symbols files are loaded. This window can help track down issues related to the Symbol files as sometimes Visual Studio will complain about not being able to find the file and then it shows a hollow breakpoint that is unusable.

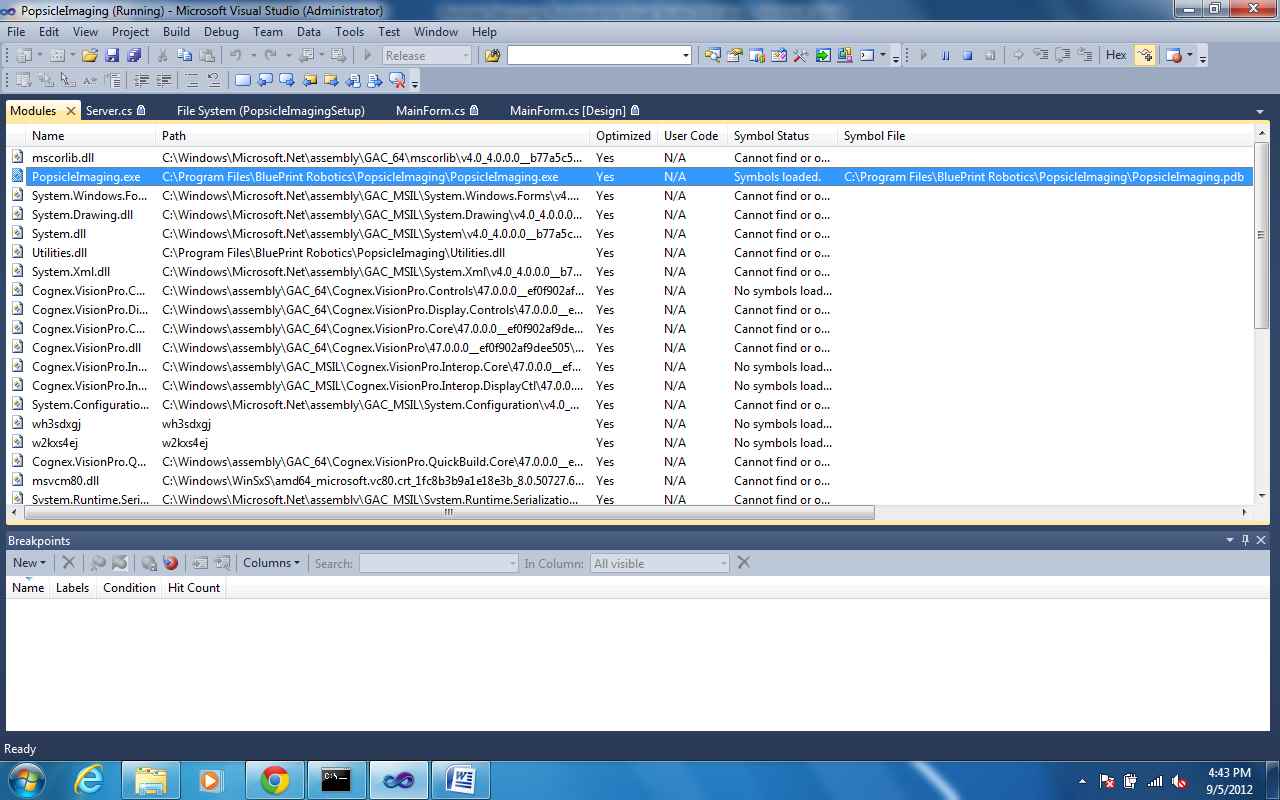


Figure 7. Modules Window.

***Alternative method to attaching to a process for remote debug***

After identical user accounts are setup on the computers and the **msvsmon** debug application is running on the remote computer, you can enter the application path in the ‘Start external program’ field and the remote machine name in the project properties debug tab. See Figure 8.

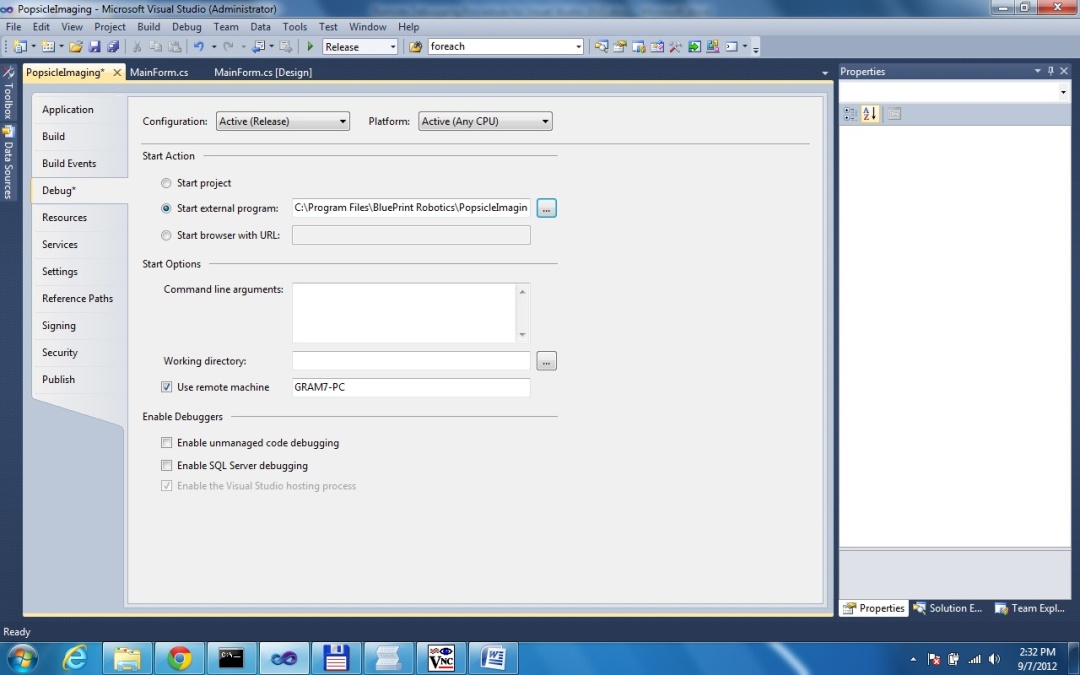
****

Figure 8. Project properties Debug Tab.

Select Release mode and press F5. You should see the application start on the remote computer. When debugging is complete, press Shift-F5 to stop everything. Note that if you clicked the browse button, you were actually browsing on the local machine, the path and file that you choose must exist on the remote machine. Or enter the path by hand if needed.

**Debugging QuickBuild scripts**

VisionPro QuickBuild scripts can also be debugged remotely. See the documentation in VisionPro for details. Here is a quick summary of the setup.

* Start the msvsmon program on the remote computer and shutdown QuickBuild if it is running.
* Create an empty project in Visual Studio on the local machine.
* Go to the project properties and select the debug tab.
* Select **Start External Program** and enter path or browse to QuickBuild.exe.
* Check **Use Remote Machine** and enter in remote computer name.
* Press F5, QuickBuild should start on the remote PC
* Open the desired QuickBuild job.
* Go to the script and select **Debug** mode.
* Run the QuickBuild job. The script should appear in Visual Studio and you should be able to single step and debug.
* When you are done, put the script back in **Release** mode.