Medusa: Testing Cardiac Devices

Sponsor: Boston Scientific

Purdue University

CS490M: Software Testing

Final Project Report

Team Medusa

Jordan Fleming – Team Leader

Ryan Landrum

Arjmand Samuel

John Valko

Colin Weaver-Johnson

Date: 12/8/06

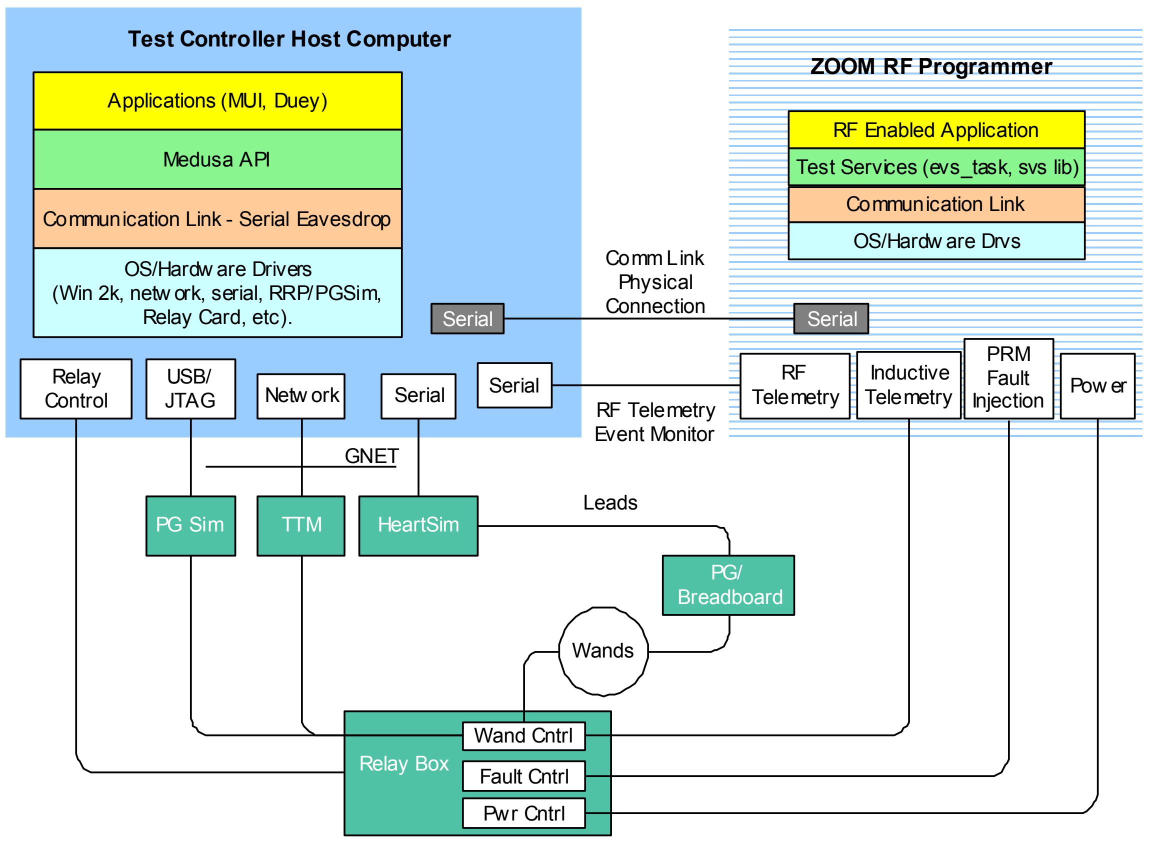
Table of Contents

1. Introduction

Medusa is a framework used internally by Boston Scientific to test its legacy, however still widely used, cardiac devices. As with any piece of software, over time issues have been found which need attention for various reasons. It was our appointed task to address these issues.

Provided to us were several pieces of hardware including a test station computer, PRM device, Insignia pacemaker device (with HP power supply), a MACHSIM heart simulator, Vitality AVT internal defibrillator, telemetry wands, and a relay box. Additionally, we were supplied a list of software change requests (SCRs) describing the details and severity of the issues as well as a synopsis of what had been done or considered already. The work on the actual Medusa system was carried out in Microsoft's Visual C++ 6 IDE using the provided source code and project workspace.

In addition to the proposed work suggested by Boston Scientific, we also derived a finite state machine to model the flow of the GUI interface on the PRM device.



- Image Courtesy of Boston Scientific -

2. Objectives

- ✓ Learn how to use Medusa from a user prospective
- ✓ Understand how Medusa works internally and its implementation
- ✓ Gain understanding of methods used for testing embedded systems
- ✓ Work on and resolve SCRs
- ✓ Test and verify SCRs
- ✓ Model PRM GUI with a finite state machine
- ✓ Merge cumulative changes and verify SCRs

3. SCR's Evaluated

SCR#	Team Member(s)	Status	Comments
963	Arjmand Samuel	Completed	Tested & Verified
1099	John Valko, Ryan Landrum	Dropped	AVT Failure
1194	Colin Weaver-Johnson, Jordan Fleming	Implemented	Not Verified
1220	John Valko	Completed	Tested & Verified
1228	Ryan Landrum	Completed	Tested & Verified
1239	John Valko	Completed	Tested & Verified
1254	Ryan Landrum	Completed	Tested & Verified

SCR #963

Synopsis

Medusa should detect the presence/absence of a heart sim and log appropriate error message and if a heart sim test is run on a non heart sim station an error mesage should be logged and the test should stop gracefully. Basically, if the test needs a heart sim but one is not present, the system currently throws an exception. This should be more gracefully handled as an error message and the test should be aborted. If the heart sim is present but not needed, no change is required to the existing functionality.

Files Changed:	Lines of code:
test_manager.cpp	1584 – 1592
heart_server.cpp	666 – 677

Verification

Start Medusa by switching off the Heart Simulator (simulating its absence). An error message will be displayed in red font in the details window as well in the summary window. A message box will appear in the main Medusa window and the system will block until OK of the message is pressed. If Medusa is started with heart simulator switched on (present), system will initialize normally.

SCR #1099

Synopsis

Under certain circumstances, the Medusa system fails to print certain values in its test report. As a result, there are verification errors. The proposed solution was to modify the template files to include a special character that would indicate that it may be omitted from the final test report and that such an omission should not warrant an error.

No files were changed (see below).

Why it was dropped

In the course of investigating this SCR we ran into severe complications. We received inconsistent behavior from the PRM and the Vitality AVT device. In the times that ensued we received many spurious errors of which we were unable to find the source. As a consequence we were unable to proceed in finding a solution to this problem. The following weeks were spent in an attempt to rectify the problems, but to no avail.

Eventually, we realized that it was possible to approach some of the SCRs without a properly functioning cardiac device, and so as a last resort we decided to move on and discontinue work on this SCR.

SCR #1194

Synopsis

This SCR deals with the logging of a particular info message that occurs very frequently. The frequency of this message loads down the CPU and takes up space on the hard drive often unnecessarily. So we have added a menu option to enable and disable the logging of the message.

Solution

The implementation involves using a global variable which is set in a listener function in the CmuiApp class and is read in the Logger class to determine whether the info message should be logged.

Files Changed:	Lines of code:
mui.h	182 - 185
mui.cpp	349 – 354
	1201 – 1216
Logger.h	244 – 251
Logger.cpp	209 - 212

Verification

The code is reached by having a message box created from the spot of the code where the info message is filtered out. We have run test scripts where we expected to see the info message displayed in the log and did not see it.

SCR #1220

Synopsis

The Medusa interface provides a toolbar allowing the user to issue various directives to the Medusa system. Among these is the ability to pause the system. However, if the system is paused, and any directive other than un-pause is issued, there is a deadlock and Medusa will crash. To remedy this problem, the other buttons are grayed allowing only the un-pause option to be selected. This was achieved by mapping appropriate message handlers in conjunction with use of a global variable currently used by Medusa in order to determine whether the system is currently paused.

Files Changed:	Lines of code:
mui.cpp:	318
	340-344
	772-773
	798-799
	878-883
	1099
	1111-1113
	1116-1158
	1255-1275

Verification

Start any test in Medusa. Pause said test. All buttons except the un-pause button should be grayed. If you un-pause the system, the buttons should return to their normal states.

SCR #1228

Synopsis

Improvements were needed to certain parts of the Medusa GUI element, specifically regarding the dialogs question_box and message_box. These two boxes are usually called within an actual test case to either ask the user a question on how to proceed, or to relay information about the current test. The enhancements that were needed were that question_box needed to have more functionality when displaying the messages, namely a vertical scroll bar to show the entire message. The message_box and question_box also needed the feature of always being displayed as the top most window, some test cases will push these windows to a non-visible area.

Solution

In order for the top most feature to be implemented, the function call that is used by message_box and question_box needed to use another overloaded implementation of that initialization call and pass in the global argument top_most, which was done completely. To implement the question_box functionality of a vertical scroll bar, it was necessary to rebuild the GUI with this feature added. Once the gui was rebuilt, the variables that hold the actual question and the ability to log any comments made by the user must be bound to the newly created resources. This has also been completed.

File Changed:	Lines of code:
mui.cpp	987 – 994
mui.rc	327-328
	336-337
	141

Verification

To test these particular issues, it would be impractical to test Microsoft's implementation of the .net gui interfaces, so it was decided to only test the ability of question_box and message_box to display when it was appropriate to do so. Both question_box and message_box were initialized manually at run-time to be sure that both were giving and using the correct functionality, which they were. Then actual medusa test files were run that would use both question_box and message_box within the actual tests. This also ran successfully.

SCR #1239

Synopsis

When running tests that use the internal printer on the PRM it is often desirable to print these jobs to a dummy device so as to not unnecessarily print. In order to do this, the computer is equipped with an electronic paper loop (EPL) card that allows printing to be redirected to a null device. This functionality is already available, however it must be enabled by any test wishing to use it and there is no interactive way to control it. In order to allow the user to have interactive control over the EPL, a checkable menu item is added in such a way that a user may monitor and change the status of the EPL device.

Solution

To achieve this we ask the relay box to redirect the printer output to the EPL. Additionally, test cases may also enable or disable the EPL and as such it is also necessary to probe the status of the relay box whenever displaying the status in the GUI.

To allow the user to select the EPL, a checkable menu was added in Options->Use EPL. When checked no printing will occur on the PRM. Both the status updates and the requests to enable/disable the EPL are handled through appropriate GUI callback functions. Additionally, the API code was slightly modified according to the attachment included with the SCRs.

File Changed:	Lines of code:
printer.cpp	78-91
	372-423
mui.cpp	345-346
	1275-1340
mui.rc	135-138

Verification

Open a test that uses the PRM printer. When the test comes to the section that utilized the printer, enable or disable the EPL and the PRM printer should stop or start respectively. It should also be observable in the menu when the test enables or disables the EPL itself.

SCR #1254

Synopsis

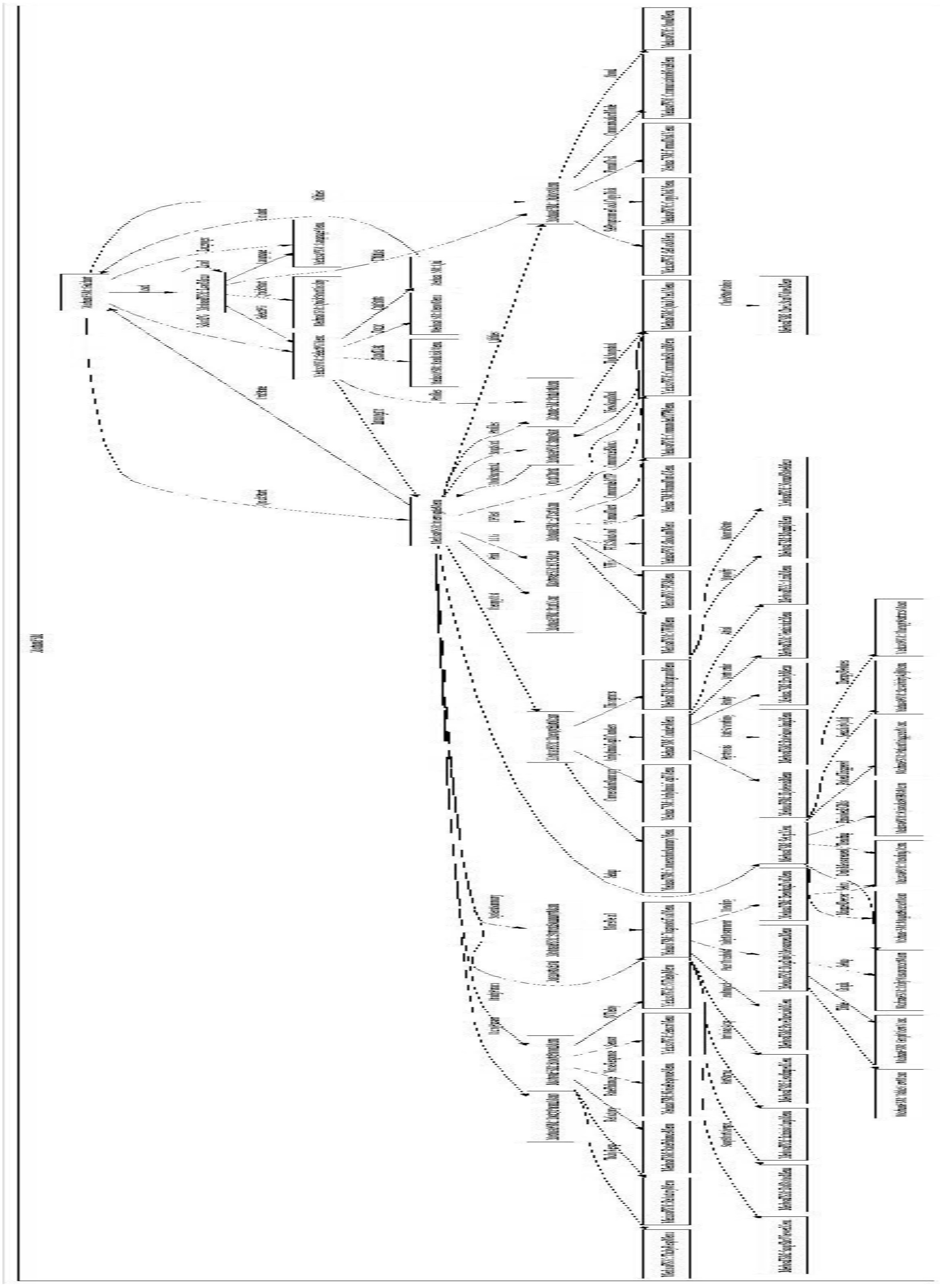
This SCR was a problem with the termination of batch runs within Medusa. Usually, the user had only one option when performing a batch run and wanting to quit. They would stop the batch run and lose the current test they were on. So, in order to remedy this situation, a implementation was needed to allow the user to tell medusa to stop the batch run after the last test completes, that way there is no lost logs and the batch run can be continued at a later time.

Solution

File Changed:	Lines of code:
test_manager.cpp	569
	2573-2579
mui.rc	707
	127
mui.cpp	339-343
	1219-1229

Verification

This SCR has not been thoroughly tested, although it has been implemented. The problem is that sometimes the batch run will not stop using either Guidant's implementation, or our implementation. Guidant's implementation of the batch break is almost identical to the implementation that we've done, it is only done later to allow the log to be written and the batch run file to be updated. It has been verified their implementation also doesn't stop batch files properly. But, on some occasions, both our implementation and Guidant's will stop the batch run successfully using certain test cases. Another problem while testing this SCR was that the Vitality AVT device began to show inconsistent behavior again much like it had done earlier during the project, and it was not practical to continue troubleshooting the Vitality AVT so close to the end of the project.



4. Finite State Model (FSM)

Synopsis

In order to further allow testing of the PRM GUI, we have constructed the above finite state machine. The machine provides a graphical representation of the transitions between different parts of the interface. Using this state machine it should be possible to systematically design tests to test the functionality of the PRM GUI. The diagram above is every state necessary to navigate each state within the PRM_GUI.

Reason for FSM

The reason we did this was to be able to test the FSM using the W method. The W method generates a minimal set of inputs to distinguish the different states in the FSM. You can then use this W set to traverse the FSM accordingly.

Tools Used.

In order to do this we used a State Machine Compiler (SMC) and also Diagram/Graph visualization tool (Graphviz). Further implementation would need Brandon Wuest's software, called Beastt, to generate the W set.

What more could be done

As it stands now, this implementation of the PRM GUI is not complete. This FSM covers only the transitions and states necessary to traverse the entire GUI of the PRM. Since about every state has a transition to every other state within the GUI, as well as a transition back to the original state, we felt it impractical to try and complete this FSM with the time remaining within the project.

In order for this FSM to be applicable for use, all these transitions need to be added. Along with this, you would use Brandon Wuest's software (Beastt) for producing the W set from a textual representation of the FSM. A W set is a minimum test set for testing the correctness of the PRM GUI. Once this W set has been completely generated, PRM GUI test scripts for can be automatically generated.

5. Other Topics Learned

Throughout our exposure to the Medusa project we have gained quite a bit of experience on subjects not covered in CS490M. Test cases, batch runs, DLL exports, direct experience with an oracle, and working on an embedded system are among some of the major topics. We have also taken concepts learned in the classroom and applied them to Medusa. We have created a Finite State Machine representing the PRM GUI. The SMC software used generates code in 9 different languages. We have chosen C++ for this application. Next, we used Beastt, a Brandon Wuest software solution for generating the W set. The W method generates a minimal set of inputs to distinguish the different states in the FSM. You can then use this W set to traverse the FSM accordingly.

6. Conclusion

In completing this project we have gained experience and learned some important lessons. We learned about how critical embedded systems are tested in the industry and how to test the software that tests embedded devices. Additionally we learned the importance of regular correspondence with project sponsors.

Throughout the period of working on the project, perhaps the most important lesson we learned was not to stick to one goal too long if that goal does not show any signs of progress. If we had thought about this sooner, it would have been possible to complete more of the SCRs and to tackle some of the more critical issues with the system.

Even though at times we found the project rather frustrating (for instance when we had issues initially understanding the architecture of the Medusa GUI or when we were unable to make progress because of strange problems with the devices), we will take away some useful knowledge and ideas about the importance of testing in the industry.

As for the completed SCRs, we are confident of our solutions and hope that they will be useful to the testing team at Boston Scientific. We would also like to express our thanks to Boston Scientific for providing this unique opportunity to experience testing applications in a real world industry environment.

7. Acknowledgments

Team Medusa would like to thank the following people for their contributions to our project. This learning experience could not have been a success without their support.

Sam Shabaneh	-	Boston Scientific
Dale Eason	-	Boston Scientific
Craig Abrahamson	-	Boston Scientific
Jim Maple	-	Boston Scientific
Dr. Aditya Mathur	-	Purdue University

8. Appendix

This appendix contains the .diff files from the original Medusa code, and our implementations within Medusa. These .diff files will be included in the revised Medusa code folder.

Legend

+	line that was added in new implementation
-	line that was taken out of original implementation
@@	line number of the original code, followed by the revised code line number

mui.cpp

```
@@ -184,9 +200,11 @@
 __declspec(dllexport) bool g_script_busy = false;                    // true when test is running
 __declspec(dllexport) bool g_abort_script = false;
 __declspec(dllexport) bool g_break = false;
+__declspec(dllexport) bool g_break_on_batch = false;
 __declspec(dllexport) bool g_break_on_string = false;
 __declspec(dllexport) string g_break_string;

+
 static g_debug_flags = 0;
 static CWinThread* test_thread_ptr= 0;  // pointer to thread running the test
 static DWORD exit_code;                                                  // medusa exit code
@@ -310,11 +328,13 @@
        ON_COMMAND(ID_FILE_RUN, OnFileRun)
        ON_COMMAND(ID_DEBUG_BREAK, OnBreak)
        ON_COMMAND(ID_DEBUG_BREAK_STRING, OnBreakString)
+
        ON_COMMAND(ID_TRACE_ENABLE, OnTraceEnable)
        ON_UPDATE_COMMAND_UI(ID_FILE_RUN, OnUpdateFileRun)
        ON_COMMAND(ID_FILE_STOP, OnFileStop)
        ON_UPDATE_COMMAND_UI(ID_FILE_STOP, OnUpdateFileStop)
        ON_COMMAND(ID_FIND_NEXTERROR, OnFindNexterror)
+       ON_UPDATE_COMMAND_UI(ID_FIND_NEXTERROR, OnUpdateFindNexterror)
        ON_COMMAND(ID_DEBUG_DATAMANAGERDUMP, OnDebugDatamanagerdump)
        ON_COMMAND(ID_DEBUG_STOPONERROR, OnDebugStoponerror)
        ON_UPDATE_COMMAND_UI(ID_DEBUG_STOPONERROR, OnUpdateDebugStoponerror)
@@ -336,12 +356,26 @@
        ON_COMMAND(IDD_PRMSNAP, OnToolsPRMScreen)
        ON_COMMAND(ID_TOOLS_START_TELEMETRY, OnToolsStart)
        ON_COMMAND(ID_TOOLS_STOP_TELEMETRY, OnToolsStop)
+
+       ON_UPDATE_COMMAND_UI(ID_FILE_OPEN, OnUpdateFileOpen)
+       ON_UPDATE_COMMAND_UI(ID_TRACE_ENABLE, OnUpdateTraceEnable)
+       ON_UPDATE_COMMAND_UI(ID_APP_ABOUT, OnUpdateAppAbout)
+       ON_UPDATE_COMMAND_UI(IDD_PRMSNAP, OnUpdatePRMSnap)
+
+       ON_COMMAND(ID_OPTIONS_USE_EPL, OnOptionsUseEPL)
+       ON_UPDATE_COMMAND_UI(ID_OPTIONS_USE_EPL, OnUpdateOptionsUseEPL)
+
+       ON_UPDATE_COMMAND_UI(ID_STOP_ONTEST, OnUpdateDebugBatchStop)
+       ON_COMMAND(ID_STOP_ONTEST,OnBatchStop)
+
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
        // Standard print setup command
        ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
```

```
+		//added by jordan fleming for the dynamics->associated text menu SCR #1194
+		ON_COMMAND(ID_DYNAMICS_ASSOCIATED_TEXT, ToggleAssociatedTextMessage)
 END_MESSAGE_MAP()

 /////////////////////////////////////////////////////////////////////////////
@@ -742,8 +776,10 @@
 // App command to run the dialog
 void CmuiApp::OnAppAbout()
 {
+
		CAboutDlg aboutDlg;
		aboutDlg.m_version = g_Test_Manager.get_version();
+		add_msg("Did about sequence (jvalko)\n");
		aboutDlg.DoModal();
 }
 /////////////////////////////////////////////////////////////////////////////
@@ -761,7 +797,8 @@
 {
		// TODO: Add your command update UI handler code here

-		pCmdUI->Enable(((m_test_file_name != "") && !g_script_busy) ? TRUE:FALSE);
+		pCmdUI->Enable(((m_test_file_name != "") && !g_script_busy
+								&& !g_pause) ? TRUE:FALSE);
 }

 void CmuiApp::OnFileStop()
@@ -786,7 +823,8 @@
 void CmuiApp::OnUpdateFileStop(CCmdUI* pCmdUI)
 {
		// TODO: Add your command update UI handler code here
-		pCmdUI->Enable((g_script_busy && !g_abort_script) ? TRUE: FALSE);
+		pCmdUI->Enable((g_script_busy && !g_abort_script)
+					&& !g_pause ? TRUE: FALSE);
 }


@@ -865,6 +903,11 @@
 }


+void CmuiApp::OnUpdateFindNexterror(CCmdUI* pCmdUI)
+{
+		pCmdUI->Enable(!g_pause);
+}
+

 void CmuiApp::add_msg(const char * msg)
 {
@@ -966,9 +1009,21 @@
 //		box.  Box stays on screen and medusa halts until user
 //		closes box by pressing OK or close buttons.  Return void.
 //		Function is a wrapper around MFC AfxMessageBox
+
+/****************************************
+*
+*		EDITED BY RYAN LANDRUM, CS490M PROJECT PURDUE UNIVERSITY 10/19/2006
+*
+*		AfxMessageBox(message);
+*					to
+*		AfxMessageBox(message, MB_SYSTEMMODAL);
+*
+*
+****************************************/
 __declspec(dllexport) void message_box(const char* message)
 {
-		AfxMessageBox(message);
+		AfxMessageBox(message, MB_SYSTEMMODAL);
+
 }

 //		Used to ask a yes or no question.  Caller specifies a
@@ -1081,7 +1136,7 @@
```

```
 void CmuiApp::OnUpdateDebugBreak(CCmdUI* pCmdUI)
 {
 	pCmdUI->SetCheck(g_break ? TRUE:FALSE);
-
+	pCmdUI->Enable(!g_pause);
 }


@@ -1093,9 +1148,53 @@

 void CmuiApp::OnUpdateDebugPause(CCmdUI* pCmdUI)
 {
+	/* jvalko: begin changes */
+	static bool wasPaused = false;
+
 	pCmdUI->SetCheck(g_pause ? TRUE:FALSE);
 	pCmdUI->Enable(g_script_busy ? TRUE:FALSE);
+
+#if 0 /* not needed ? */
+	if(wasPaused != g_pause)
+	{
+		wasPaused = g_pause;
+		CToolBar *pCTB = reinterpret_cast<CToolBar*>(pCmdUI->m_pOther);
+		CToolBarCtrl &toolCtl = pCTB->GetToolBarCtrl();
+		CString tempStr;
+		int index = 0;
+		TBBUTTON button = {0};
+
+
+		/* enumerate toolbar buttons */
+		for(index = 0;
+			index < toolCtl.GetButtonCount();
+			index++)
+			{
+				if(toolCtl.GetButton(index, &button))
+				{
+					tempStr = "Button[";
+					tempStr += index;
+					tempStr += "] = ";
+					tempStr += (button.fsState & TBSTATE_CHECKED) ?
+						"On" : "Off";
+					tempStr += "\n";
+					add_msg(tempStr);
+					if(button.idCommand != ID_DEBUG_PAUSE)
+					{
+						toolCtl.EnableButton(button.idCommand, !g_pause);
+					}
+				}
+				else
+				{
+					add_msg("Failed to get toolbar button!\n");
+				}
+			}

+		AfxMessageBox("are buttons greyed?", MB_OK, 0);
+
+		/* jvalko: end changes */
+	}
+#endif
 }
 #include "logger.h"
 void CmuiApp::OnDebugDumpObject()
@@ -1191,4 +1290,115 @@
 void CmuiApp::OnToolsStop()
 {
 	g_Telemetry_Data.get_data();
+}
+
+void CmuiApp::OnUpdateFileOpen(CCmdUI *pCmdUI)
+{
+	pCmdUI->Enable(!g_pause);
+}
```

```
+
+void CmuiApp::OnUpdateTraceEnable(CCmdUI *pCmdUI)
+{
+	pCmdUI->Enable(!g_pause);
+}
+
+void CmuiApp::OnUpdateAppAbout(CCmdUI *pCmdUI)
+{
+	pCmdUI->Enable(!g_pause);
+}
+
+void CmuiApp::OnUpdatePRMSnap(CCmdUI *pCmdUI)
+{
+	pCmdUI->Enable(!g_pause);
+}
+
+#include "Printer.h"
+#include "RelayBox.h"
+#include "relay.h"
+
+void CmuiApp::OnOptionsUseEPL()
+{
+	if (g_Relay_Box.isPresent())
+	{
+		if(!g_Printer_ptr)
+		{
+			AfxMessageBox("g_Printer_ptr was NULL!", MB_ICONEXCLAMATION, 0);
+			return;
+		}
+		if (g_Printer_ptr->epl_installed)
+		{
+			if(!g_Relay_Box.get(Relay(EPL_RELAY)))
+			{
+				if (g_Printer_ptr->enable_electronic_paper_loop())
+				{
+					g_log.log(Logger::INFO, "Electronic paper loop is available....
excellent!");
+				}
+				else
+				{
+					g_log.log(Logger::INFO, "Electronic paper loop is not available....
bummer!");
+				}
+			}
+			else
+			{
+				if (g_Printer_ptr->disable_electronic_paper_loop())
+				{
+					g_log.log(Logger::INFO, "Electronic paper loop has been
disabled.");
+				}
+				else
+				{
+					g_log.log(Logger::INFO, "Electronic paper loop has not been
disabled!");
+				}
+			}
+		}
+	}
+}
+
+void CmuiApp::OnUpdateOptionsUseEPL(CCmdUI *pCmdUI)
+{
+	if (g_Relay_Box.isPresent())
+	{
+		if(!g_Printer_ptr)
+		{
+			AfxMessageBox("g_Printer_ptr was NULL!", MB_ICONEXCLAMATION, 0);
+			return;
+		}
+		if (g_Printer_ptr->epl_installed)
+		{
```

```
+                                       pCmdUI->SetCheck(g_Relay_Box.get(Relay(EPL_RELAY)));
+                               }
+                               else
+                               {
+                                       pCmdUI->Enable(false);
+                               }
+               }
+               else
+               {
+                       pCmdUI->Enable(false);
+               }
+
+}
+
+void CmuiApp::OnUpdateDebugBatchStop(CCmdUI* pCmdUI)
+{
+               pCmdUI->SetCheck(g_break_on_batch ? TRUE:FALSE);
+}
+void CmuiApp::OnBatchStop()
+{
+               message_box("Break has been set");
+               g_break_on_batch = !g_break_on_batch;
+
+}
+//added by jordan fleming SCR #1194
+void CmuiApp::ToggleAssociatedTextMessage()
+{
+               g_toggle_associated_text = !g_toggle_associated_text;
+               //create a message window that reports whether the messages should be present
+               string message = "Reporting of \"There is no associated text defined in the database for tag\" has been ";
+               if(g_toggle_associated_text){
+                       message += "enabled.";
+               }else{
+                       message += "disabled.";
+               }
+               message_box(message.c_str());
+
 }
```

mui.h

```
--- C:\medusa\include\mui.h        Thu Aug 10 12:04:00 2006
+++ medusa\include\mui.h           Wed Dec 06 21:58:10 2006
@@ -8,8 +8,21 @@
  * FILE: mui.h
  * AUTHOR: D.Eason
  *
-* $Header: c:\cvsrepo\/Medusa/include/mui.h,v 1.1.1.1 2006/08/10 17:04:00 g032728 Exp $
+* $Header: C:\cvsrepo/Medusa/include/mui.h,v 1.5 2006/12/07 02:58:10 PURDUE Exp $
  * $Log: mui.h,v $
+* Revision 1.5  2006/12/07 02:58:10  PURDUE
+*
+* Checked in code for SCRs 1220,1239
+*
+* Revision 1.4  2006/12/07 02:23:14  PURDUE
+* landrum attempt uh...5
+*
+* Revision 1.3  2006/12/07 01:14:35  PURDUE
+* commit attempt jordan 1
+*
+* Revision 1.2  2006/10/22 18:51:26  PURDUE
+* no message
+*
  * Revision 1.1.1.1  2006/08/10 17:04:00  g032728
  * no message
  *
@@ -160,6 +173,7 @@
               afx_msg void OnFileStop();
               afx_msg void OnUpdateFileStop(CCmdUI* pCmdUI);
               afx_msg void OnFindNexterror();
```

```
+		afx_msg void OnUpdateFindNexterror(CCmdUI* pCmdUI);
		afx_msg void OnDebugDatamanagerdump();
		afx_msg void OnDebugStoponerror();
		afx_msg void OnUpdateDebugStoponerror(CCmdUI* pCmdUI);
@@ -176,8 +190,23 @@
		afx_msg void OnToolsPRMScreen();
		afx_msg void OnToolsStart();
		afx_msg void OnToolsStop();
+		afx_msg void OnUpdateFileOpen(CCmdUI* pCmdUI);
+		afx_msg void OnUpdateTraceEnable(CCmdUI* pCmdUI);
+		afx_msg void OnUpdateAppAbout(CCmdUI* pCmdUI);
+		afx_msg void OnUpdatePRMSnap(CCmdUI* pCmdUI);
+
+		afx_msg void OnOptionsUseEPL();
+		afx_msg void OnUpdateOptionsUseEPL(CCmdUI* pCmdUIm);
+
+		//added by ryan landrum SCR # 1254
+		afx_msg void OnUpdateDebugBatchStop(CCmdUI* pCmdUI);
+		afx_msg void OnBatchStop();
+		//added by jordan fleming SCR # 1194
+		afx_msg void ToggleAssociatedTextMessage();
+
		//}}AFX_MSG
		DECLARE_MESSAGE_MAP()
+
 private:
		CRect normalize_window(CRect rect);
		PROCESS_INFORMATION *mp_RemoteQNXProcessInfo;
```

heart_server.cpp

```
--- C:\medusa\heart_sim\heart_server.cpp	Tue Oct 17 08:57:53 2006
+++ medusa\heart_sim\heart_server.cpp	Wed Dec 06 22:11:26 2006
@@ -8,7 +8,7 @@
 // FILE:		heart_server.cpp
 // AUTHOR:	T. Lendway
 //
-// $Revision: 1.1.1.1 $
+// $Revision: 1.2 $
 //
 // $Modtime:   20 Nov 2003 14:55:56  $
 //
@@ -248,22 +248,6 @@
 {
   string conn_from_ini;

-
-
-
-
-//----------add by arjmand
-   g_log.log(Logger::INFO, "Arjmand: this is a test log.");
-
-//---------- end add by arjmand
-
-
-
-
-
-
-
-
   m_using_dpr = false; //Default setting
   if (g_Config.get_string("Peripheral", "ConnectionType", conn_from_ini))
   {
@@ -286,9 +270,17 @@
   if (g_Heart_Server.setup() == false)
   {
      g_log.log(Logger::INFO, "Heart Simulator driver unable to initialize.");
-     return;
+
+
+
+                return;
    }

+
+
+
+
+
    // Initialize heart sim hardware
    if (g_Heart_Server.heart_sim_available() == false)
    {
@@ -667,8 +659,24 @@

    if (g_Heart_Server.heart_sim_available() == false)
    {
-     g_log.log(Logger::ERR, "Heart Simulator not available (Check heartsim connection and power).");
-     return false;
+      g_log.log(Logger::ERR, "\Heart Simulator not available (Check heartsim connection and power).");
+
+
+
+//_---------------------------------added by purdue
+
+//              heart_sim_found=0; //to throw a mesage at the end of init
+
+                stringstream msg;
+           msg << "Heart Simulator not found. Cannot run test scripts which require heart sim.";
+           AfxMessageBox(msg.str().c_str());
+
+                g_log.log(Logger::ERR, "\Purdue: Donot run test scripts which require Heart Simulator!!");
+      g_log.log(Logger::ERR, "\Purdue: or add check_heart_sim() at the begining of test script!!");
+
+//_---------------------------------added by purdue
+
+                return false;
    }
    else // HW is available, get type and revision which are stored together
        // in a 16-bit word.  Type is in the top 2 bits, revision is the lower
@@ -676,6 +684,9 @@
        // assuming the hardware is correct (ini file error is tolerated).
    {

+//              heart_sim_found=1; //purdue
+
+
      if (m_using_dpr)
      {
        page_select(PAGE_0);
@@ -709,7 +720,7 @@
    else // success
    {
      stringstream msg;
-     msg << "aaaa Heart Sim scenario directory = " << m_scenario_dir;
+      msg << "Heart Sim scenario directory = " << m_scenario_dir;
      g_log.log(Logger::INFO, msg.str().c_str());
    }
    return true;
@@ -2238,5 +2249,32 @@
    return false;

 } // end of move_to_heart_sim
+
+
+
+bool Heart_Server::check_heart_sim()
+{
+if (g_Heart_Server.heart_sim_available() == false)
+{
+               return(false);
+
+}else
+
+{
+
+               return(true);
+
+}
+
+
+
+
+
+}
+
+
+
+
+

 // END OF SOURCE FILE **
```

logger.cpp

```
--- C:\medusa\logger\logger.cpp   Thu Aug 10 12:04:05 2006
+++ medusa\logger\logger.cpp    Thu Dec 07 01:12:50 2006
@@ -9,8 +9,17 @@
 // FILE:          Logger.cpp
 // AUTHOR:        Dale Eason
 //
-// $Header: c:\cvsrepo\/Medusa/logger/Logger.cpp,v 1.1.1.1 2006/08/10 17:04:05 g032728 Exp $
+// $Header: C:\cvsrepo/Medusa/logger/Logger.cpp,v 1.4 2006/12/07 06:12:50 PURDUE Exp $
 // $Log: Logger.cpp,v $
+// Revision 1.4  2006/12/07 06:12:50  PURDUE
+// Final changes
+//
+// Revision 1.3  2006/12/07 01:14:36  PURDUE
+// commit attempt jordan 1
+//
+// Revision 1.2  2006/10/01 03:12:58  PURDUE
+// no message
+//
 // Revision 1.1.1.1  2006/08/10 17:04:05  g032728
 // no message
 //
@@ -172,6 +181,7 @@
 __declspec(dllimport) bool g_break_on_string;
 __declspec(dllimport) string g_break_string;

+
 using namespace std;

 //****************************************************************************
@@ -202,6 +212,8 @@
 //
 //****************************************************************************
 __declspec(dllexport) bool g_pause = false;
+//added by jordan fleming SCR #1194
+__declspec(dllexport) bool g_toggle_associated_text = true;
 CString logger_first_error;
 //****************************************************************************
 //
@@ -373,7 +385,15 @@
 void Logger::log(const int type, const char * msg)
 {
		ASSERT(msg != 0);
-		ASSERT(type > 0);
+		ASSERT(type > 0);
+
+
+		if(type == Logger::INFO && g_toggle_associated_text == false){
+			if(0 == strncmp(msg,"There is no", sizeof "There is no")){
+				message_box(msg);
+				return;
+			}
+		}

		// increment this type statistics
		m_statistics[type]++;
```

---

logger.h

```
--- C:\medusa\include\logger.h	Thu Aug 10 12:04:00 2006
+++ medusa\include\logger.h	Wed Dec 06 22:11:27 2006
@@ -12,8 +12,15 @@
 // FILE:		Logger.h
 // AUTHOR:		Dale Eason
 //
-// $Header: c:\cvsrepo\/Medusa/include/Logger.h,v 1.1.1.1 2006/08/10 17:04:00 g032728 Exp $
+// $Header: C:\cvsrepo/Medusa/include/Logger.h,v 1.3 2006/12/07 03:11:27 PURDUE Exp $
 // $Log: Logger.h,v $
+// Revision 1.3  2006/12/07 03:11:27  PURDUE
+//
+// Commited code for SCR 963
+//
+// Revision 1.2  2006/12/07 01:14:35  PURDUE
+// commit attempt jordan 1
+//
 // Revision 1.1.1.1  2006/08/10 17:04:00  g032728
 // no message
 //
@@ -241,10 +248,13 @@
 #   ifndef INSIDE_MEDUSA_DLL
 #      define _LOGGER_API __declspec(dllimport)
			__declspec(dllimport) bool g_pause;
-
+			//added by jordan fleming SCR #1194
+	__declspec(dllimport) bool g_toggle_associated_text;
 #   else
 #      define _LOGGER_API
-			extern bool_g_pause;
+			extern bool g_pause;
+			//added by jordan fleming SCR #1194
+			extern bool g_toggle_associated_text;
 #   endif
 #endif

@@ -314,6 +324,9 @@
 //
 //*****************************************************************************

+//int heart_sim_found=0;
+
+
 //*****************************************************************************
 //
 // TYPEDEFS AND ENUMERATIONS
@@ -354,6 +367,10 @@
 //
 //*****************************************************************************
 public:
+
+
+
+

  // log information
         struct log_info
```

mui.rc

```
--- C:\medusa\newmui\mui.rc      Thu Aug 10 12:04:05 2006
+++ medusa\newmui\mui.rc         Thu Dec 07 00:22:12 2006
@@ -122,16 +122,21 @@
     POPUP "Debug"
     BEGIN
         MENUITEM "Trace ",                  ID_TRACE_ENABLE
-        MENUITEM "Stop_on_error\tCtrl+E",       ID_DEBUG_STOPONERROR
+        MENUITEM "Stop on error\tCtrl+E",       ID_DEBUG_STOPONERROR
         , CHECKED
         MENUITEM "Pause",                   ID_DEBUG_PAUSE, CHECKED
         MENUITEM "Stop on next message\tCtrl+C", ID_DEBUG_BREAK, CHECKED
+        MENUITEM "Stop batch after current test", ID_STOP_ONTEST
         MENUITEM "Break on message string",     ID_DEBUG_BREAK_STRING
         , CHECKED
         MENUITEM SEPARATOR
         MENUITEM "Display object",          ID_DEBUG_DUMP_OBJECT
         MENUITEM "Data Manager Dump",           ID_DEBUG_DATAMANAGERDUMP
     END
+    POPUP "Options"
+    BEGIN
+        MENUITEM "Use &EPL",                ID_OPTIONS_USE_EPL
+    END
     POPUP "Tools"
     BEGIN
         MENUITEM "Stimulators\t...",        ID_TOOLS_STIMULATORS
@@ -139,6 +144,10 @@
         MENUITEM "Start Capture Telemetry",     ID_TOOLS_START_TELEMETRY
         MENUITEM "Stop Capture Telemetry",      ID_TOOLS_STOP_TELEMETRY
     END
+    POPUP "Dynamics"
+    BEGIN
+        MENUITEM "Associated Text",         ID_DYNAMICS_ASSOCIATED_TEXT
+    END
     POPUP "&Help"
     BEGIN
         MENUITEM "&About newmui...",        ID_APP_ABOUT
@@ -285,18 +294,19 @@
 END

 IDD_QUESTION_DLG DIALOG DISCARDABLE  0, 0, 324, 133
-STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_VISIBLE | WS_CAPTION |
-    WS_SYSMENU
+STYLE DS_SYSMODAL | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_VISIBLE |
+    WS_CAPTION | WS_SYSMENU
 CAPTION "Question Box"
 FONT 8, "MS Sans Serif"
 BEGIN
     DEFPUSHBUTTON   "YES",IDOK,79,54,50,14
     PUSHBUTTON      "No",IDCANCEL,165,55,50,14
-    LTEXT           "Static",IDC_QB_QUESTION,15,13,295,36
     LTEXT           "Note: (add a note to be placed in the log)",IDC_STATIC,
                     27,70,130,8
     EDITTEXT        IDC_EDIT1,24,85,275,37,ES_MULTILINE | ES_AUTOVSCROLL |
                     ES_AUTOHSCROLL | ES_WANTRETURN | WS_HSCROLL
+    EDITTEXT        IDC_EDIT2,13,13,298,39,ES_MULTILINE | ES_AUTOVSCROLL |
+                    ES_READONLY | WS_VSCROLL | WS_GROUP | NOT WS_TABSTOP
 END

 IDD_DIALOG1 DIALOG DISCARDABLE  0, 0, 404, 130
@@ -705,6 +715,7 @@
```

```
    ID_TOOLS_EAVESDROP      "Use eavesdrop to talk to the Programmer"
    ID_TOOLS_STIMULATORS    "Control stimulator state"
    ID_BUTTON_PRMSCREEN     "PRM Screen Image"
+   ID_STOP_ONTEST          "Stops batch run after current test"
    ID_MENU_GETTAG          "Get Tag"
    ID_MENU_VERIFY          "Verify GUI Item"
    ID_MENU_REFRESH         "Screen refresh"
```

Printer.cpp

```
--- C:\medusa\prm\printer.cpp     Thu Aug 10 12:04:14 2006
+++ medusa\prm\printer.cpp        Wed Dec 06 21:58:13 2006
@@ -8,11 +8,12 @@
 * FILE:                  Printer.cpp
 * AUTHOR:                Jignesh.parekh
 *
-* $Header: c:\cvsrepo\/Medusa/prm/Printer.cpp,v 1.1.1.1 2006/08/10 17:04:14 g032728 Exp $
+* $Header: C:\cvsrepo/Medusa/prm/Printer.cpp,v 1.2 2006/12/07 02:58:13 PURDUE Exp $
 *
 * $Log: Printer.cpp,v $
-* Revision 1.1.1.1  2006/08/10 17:04:14  g032728
-* no message
+* Revision 1.2  2006/12/07 02:58:13  PURDUE
+*
+* Checked in code for SCRs 1220,1239
 *
 //
 //    Rev 1.2   Feb 22 2002 15:27:28   bruce.chenoweth
@@ -78,6 +79,19 @@
 {
   //## begin Printer::Printer%.body preserve=yes
              g_Printer_Output_ptr->start();
+
+   string ini_file_entry_value;
+   epl_installed = false;
+
+   if (g_Config.get_string("PRINTER", "EPL_PRESENT", ini_file_entry_value))
+   {
+      if (strcmpi(ini_file_entry_value.c_str(), "false") &&
+          strcmpi(ini_file_entry_value.c_str(), "0") &&
+          strcmpi(ini_file_entry_value.c_str(), "no"))
+      {
+         epl_installed = true;
+      }
+   }
   //## end Printer::Printer%.body
 }

@@ -356,6 +370,57 @@
                           return false;
    }
  //## end Printer::clear_fault%912715469.body
+}
+
+
+bool Printer::enable_electronic_paper_loop()
+{
+   //## begin Printer::enable_electronic_paper_loop%912715470.body preserve=yes
+
+              if (g_Relay_Box.isPresent())
+   {
+      if (epl_installed)
+      {
+         g_Relay_Box.set(Relay(EPL_RELAY));
+         return g_Relay_Box.get(Relay(EPL_RELAY));
+      }
+      else
+      {
+         return false;
+      }
+  }
+  else
+  {
+     AfxMessageBox("Relay Box is not present\nPlease turn electronic paper loop ON",
+                  MB_OK | MB_ICONINFORMATION);
+     return true;
+  }
+  //## end Printer::enable_electronic_paper_loop%912715470.body
+}
+
+
+bool Printer::disable_electronic_paper_loop()
+{
+  //## begin Printer::disable_electronic_paper_loop%912715471.body preserve=yes
+  if (g_Relay_Box.isPresent())
+  {
+     if (epl_installed)
+     {
+        g_Relay_Box.reset(Relay(EPL_RELAY));
+        return !g_Relay_Box.get(Relay(EPL_RELAY));
+     }
+     else
+     {
+        return false;
+     }
+  }
+  else
+  {
+     AfxMessageBox("Relay Box is not present\nPlease turn electronic paper loop OFF",
+                  MB_OK | MB_ICONINFORMATION);
+     return true;
+  }
+  //## end Printer::disable_electronic_paper_loop%912715471.body
 }
```

test_manager.cpp

```
--- C:\medusa\testman\test_manager.cpp       Thu Aug 10 12:04:16 2006
+++ medusa\testman\test_manager.cpp         Wed Dec 06 22:11:47 2006
@@ -9,8 +9,15 @@
 * FILE: test_manager.cpp
 * AUTHOR: D.Eason
 *
-* $Header: c:\cvsrepo\/Medusa/testman/test_manager.cpp,v 1.1.1.1 2006/08/10 17:04:16 g032728 Exp $
+* $Header: C:\cvsrepo/Medusa/testman/test_manager.cpp,v 1.3 2006/12/07 03:11:47 PURDUE Exp $
 * $Log: test_manager.cpp,v $
+* Revision 1.3  2006/12/07 03:11:47  PURDUE
+*
+* Commited code for SCR 963
+*
+* Revision 1.2  2006/12/07 02:23:15  PURDUE
+* landrum attempt uh...5
+*
 * Revision 1.1.1.1  2006/08/10 17:04:16  g032728
 * no message
 *
@@ -558,6 +565,17 @@
 bool check_for_log_and_halt_after_test_runs = false;
 string prm_log_file_stats;

+/*******************
+*
+*Added by PURDUE
+*
+*  Ryan Landrum
+*  12/6/06
+*
+********************/
+__declspec(dllimport) bool g_break_on_batch;
+
+
 // Class Test_Manager
 //static
 Test_Manager Test_Manager::mg_Test_Manager;
@@ -1578,7 +1596,17 @@
      LocalFree( lpMsgBuf );
   }

-   g_log.log(Logger::INFO, "System Init complete\n");
+
+
+
+//purdue
+
+
+   //g_log.log(Logger::ERR, "\Purdue: Donot run test scripts which require Heart Simulator!!");
+
+
+
+   g_log.log(Logger::INFO, "System Init complete....\n");

   runUnitTests();

@@ -1633,6 +1661,9 @@
      for (int i = 0;i <iNumberTests; i++)
      {
        //Get the current test dll
+
+                              //INSERT CONDITION FOR STOP OF BATCH HERE
+
        stringstream currentTest;
        currentTest << test << i;
        string currentTestName = currentTest.str();
@@ -2547,6 +2578,22 @@
                // remove current test from test list
                test_list.pop_back();
                // get next test from excel
+
+                                                  /**************
+
+                                                            Added by Purdue
+
+                                                            Ryan Landrum
+                                                            12/6/06
+
+                                                   **************/
+                                                  if(g_break_on_batch)
+                                                  {
+
+                                                                    break;
+
+
+                                                  }
               CString tmp;
               if (excel_app.allocate_a_test(tmp, station_id.c_str(), MEDUSA_VERSION_NUMBER))
               {
@@ -2573,6 +2620,7 @@
          {
           break;
          }
+
        }

                              /* SCR 1181
```