TYX CORPORATION

Productivity Enhancement Systems

PAWS Studio Release Notes

Version 1.39.14
February 4, 2013

Table of contents

1 Paws Developer's Studio

Version 1.39.14

Release date: February 4, 2013

1.1 Critical Items

1.2 Known Limitations

1.3 Enhancements

1.3.1 Display Information About Allocation Errors (TaskID 1135)

A double-click on an allocation error in Paws Studio opens the allocation input file or the SGL file.

If the allocation input file is a .pproc (generated by the preprocessor), the Studio still opens the original file. The only contexts in which no document is open are:

* The double-click action is not done on a genuine allocation error.

* The double-click action is done on an allocation error whose origin is an allocation file, but the allocation file is created from scratch by the preprocessor.

The .SGL file gets open when the allocation error specifies the signal name that could not be allocated, like:

1: Unable to Resolve S15 (No Candidates) (ERROR)

Paws Studio will scroll the .SGL file at the place where the failed assignment is reported on S15.

The allocation file is opened when the allocation error does not specify the name of the failed signal, like:

4: [DCP FNC 1|; end;] Assign: NOT a Candidate (ERROR)

In this case, the allocation input file is scrolled to line 4.

1.3.2 Unable to Debug When an Allocation Error is Missing (TaskID 1134)

When the Paws Project defined an allocinput file that was missing on the disk, any attempt to launch the debugger failed, with the error ”Project not Up to Date” . A RebuildAll action on the project did not resolve the issue.

This problem has been fixed in this release.

1.4 Problem Reports

2 Run Time System

Version 1.39.14

Release date: February 4, 2013

2.1 Critical Items

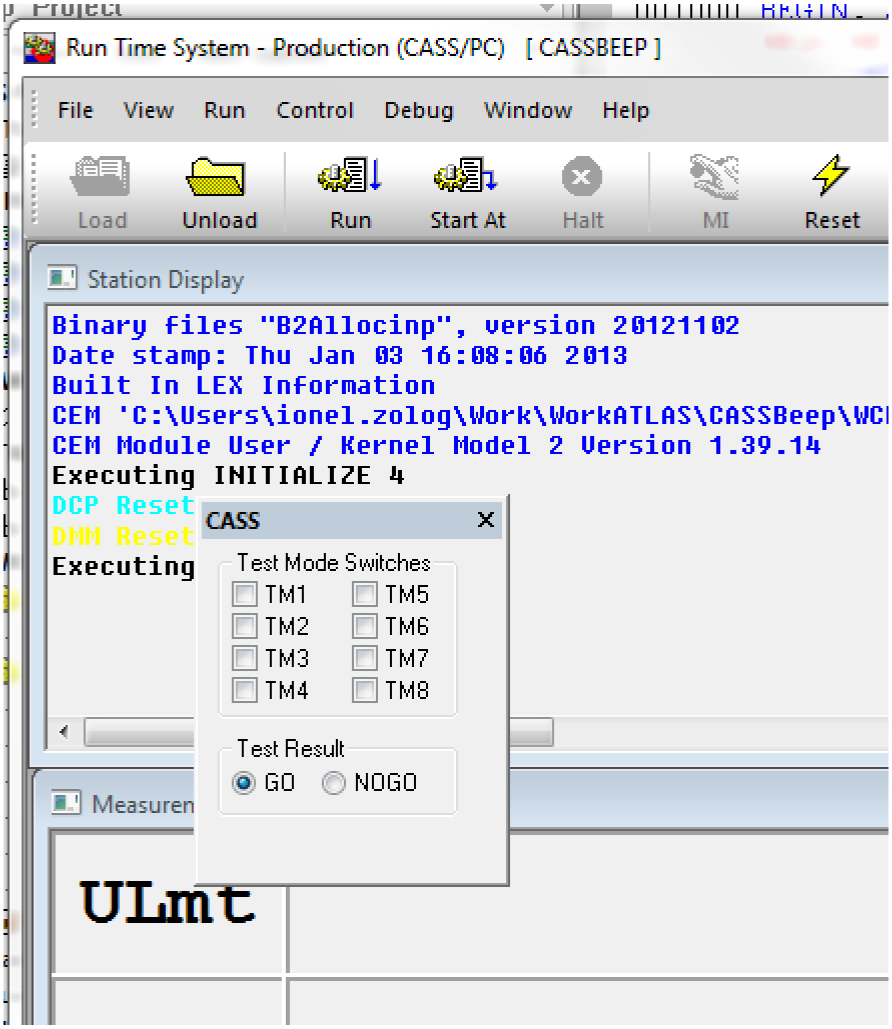
2.2 Known Limitations

2.3 Enhancements

2.3.1 RTS Populates CASS .OVERRIDES Global Variable Properly (Task ID 1132)

This release resolves the treatment of .OVERRIDES global variable, when a GO value is changed from FALSE to TRUE (this is what we call in the text below an OVERRIDE action). The increment of the OVERRIDE global occurs when the RTS executes the next statement.

An OVERRIDE action can be invoked from the RTS window by clicking the GO and NOGO radio buttons in the CASS status bar:



```
Binary files "B2Allocinp", version 20121102
Date stamp: Thu Jan 03 16:08:06 2013
Built In LEX Information
CEM 'C:\Users\ionel.zolog\Work\WorkATLAS\CASSBeep\WC
CEM Module User / Kernel Model 2 Version 1.39.14
Executing INITIALIZE 4
DCP Reset
DMM Reset
Executing
```

CASS

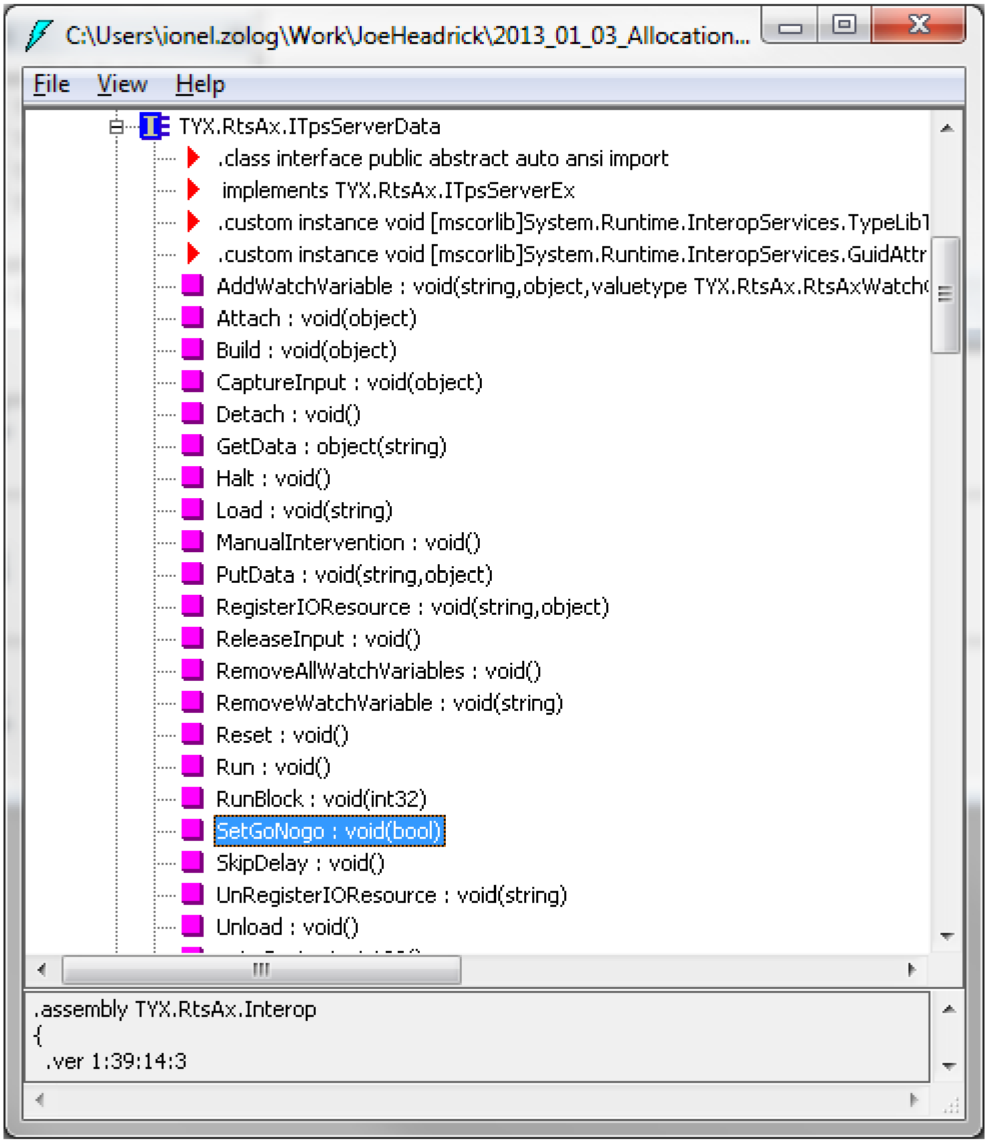
Test Mode Switches

☐ TM1 ☐ TM5
☐ TM2 ☐ TM6
☐ TM3 ☐ TM7
☐ TM4 ☐ TM8

Test Result

● GO ○ NOGO

An OVERRIDE action can be issued from a CASS client, by calling the new SetGoNogo method implemented in ITpsServerData:


C:\Users\ionel.zolog\Work\JoeHeadrick\2013_01_03_Allocation...
File
View
Help

```
TYX.RtsAx.ITpsServerData
    .class interface public abstract auto ansi import
    implements TYX.RtsAx.ITpsServerEx
    .custom instance void [mscorlib]System.Runtime.InteropServices.TypeLibT
    .custom instance void [mscorlib]System.Runtime.InteropServices.GuidAttr
    AddWatchVariable : void(string,object,valuetype TYX.RtsAx.RtsAxWatch(
    Attach : void(object)
    Build : void(object)
    CaptureInput : void(object)
    Detach : void()
    GetData : object(string)
    Halt : void()
    Load : void(string)
    ManualIntervention : void()
    PutData : void(string,object)
    RegisterIOResource : void(string,object)
    ReleaseInput : void()
    RemoveAllWatchVariables : void()
    RemoveWatchVariable : void(string)
    Reset : void()
    Run : void()
    RunBlock : void(int32)
    SetGoNogo : void(bool)
    SkipDelay : void()
    UnRegisterIOResource : void(string)
    Unload : void()
```

```
.assembly TYX.RtsAx.Interop
{
  .ver 1:39:14:3
```

The change of the GO value directly (as a debug variable from Paws Studio or from an RTS client) is not interpreted as an OVERRIDE action, so the expected increment of the .OVERRIDE global variable at the next atlas statement does not occur.

Instead, SetGoNogo(true) issues an OVERRIDE action if the value of the GO global variable at that time is FALSE.

2.3.2 PawsDebugger.dll Identifies Anywhere in TPS the Addresses of Any Global (TaskID 1137)

Our FindAddressAndTypeInformation API method used to search for an atlas variable address and type, in the current scope of the RTS only. After this release, our new FindAddressAndTypeInformation extends its search in the scopes of the first statements of all the other TPS modules, if the original search in the RTS current scope fails.

As a result of this search extension, our new FindAddressAndTypeInformation is able to return addresses of global variables that are declared in modules different than the module of the current RTS scope.

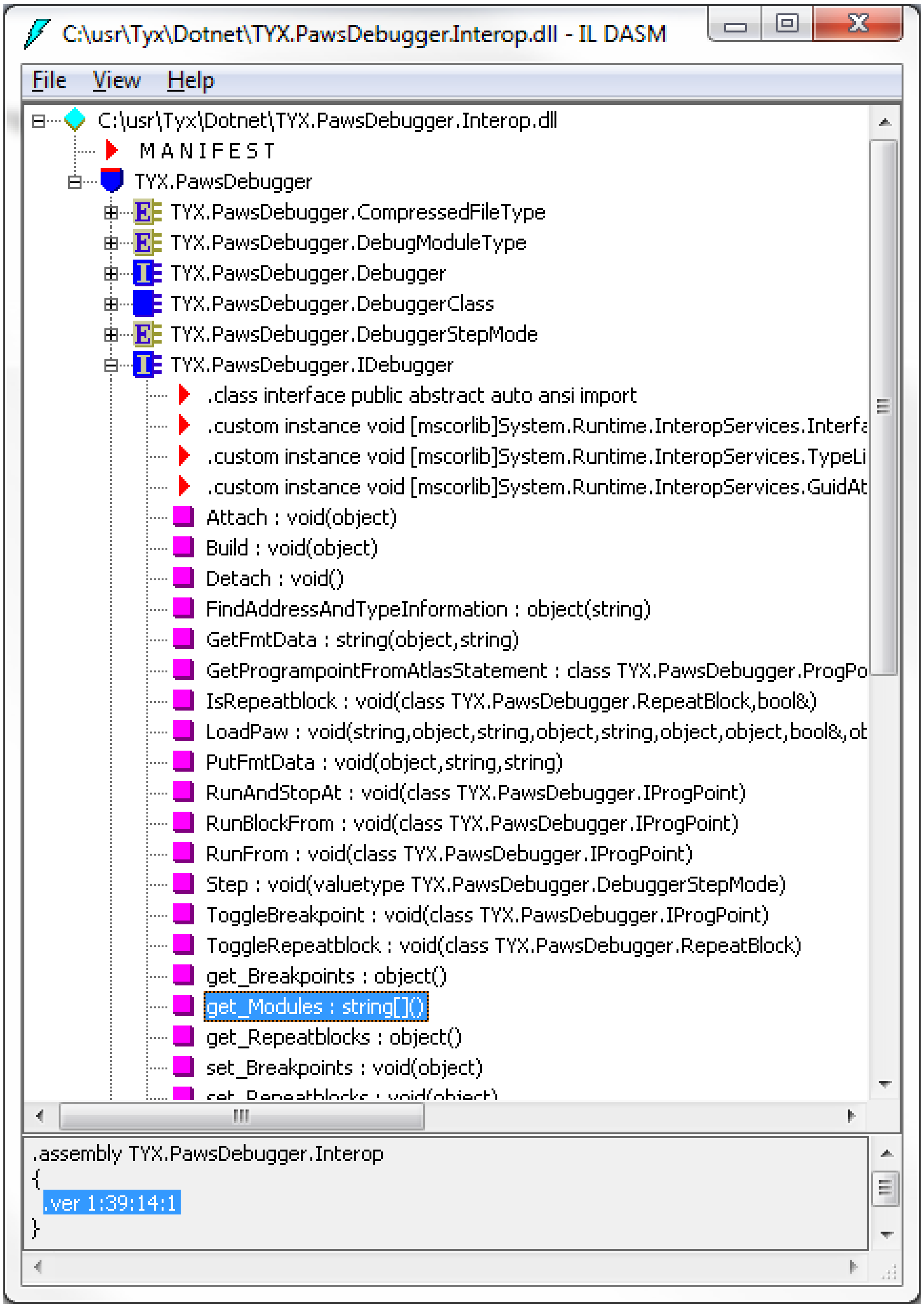
This approach allowed us to preserve the arguments of FindAddressAndTypeInformation, without the addition of a new "moduleName" argument.

There is no risk of identifying a different global address by parsing the modules of a TPS, because any attempt to declare a GLOBAL atlas variable with the same name in distinctive modules is signaled as a linker error.

However, the present atlas compiler and linker allow the declaration of local variables (they are neither GLOBAL, nor EXTERNAL) in an atlas module. When our new FindAddressAndTypeInformation runs, such DECLARE statements of local variables may hide other DECLARE GLOBAL or DECLARE EXTERNAL statements of global variables in other modules. This is a current limitation of our present solution.

2.3.3 PawsDebugger.dll Provides the List of Modules After a TPS is Loaded (TaskID 1121)

The following picture shows the location and type of the new read-only Modules property:



```
C:\usr\Tyx\Dotnet\TYX.PawsDebugger.Interop.dll
    MANIFEST
    TYX.PawsDebugger
        TYX.PawsDebugger.CompressedFileType
        TYX.PawsDebugger.DebugModuleType
        TYX.PawsDebugger.Debugger
        TYX.PawsDebugger.DebuggerClass
        TYX.PawsDebugger.DebuggerStepMode
        TYX.PawsDebugger.IDebugger
            .class interface public abstract auto ansi import
            .custom instance void [mscorlib]System.Runtime.InteropServices.Interfa
            .custom instance void [mscorlib]System.Runtime.InteropServices.TypeLi
            .custom instance void [mscorlib]System.Runtime.InteropServices.GuidAt
            Attach : void(object)
            Build : void(object)
            Detach : void()
            FindAddressAndTypeInformation : object(string)
            GetFmtData : string(object,string)
            GetProgrampointFromAtlasStatement : class TYX.PawsDebugger.ProgPo
            IsRepeatblock : void(class TYX.PawsDebugger.RepeatBlock,bool&)
            LoadPaw : void(string,object,string,object,string,object,object,bool&,ob
            PutFmtData : void(object,string,string)
            RunAndStopAt : void(class TYX.PawsDebugger.IProgPoint)
            RunBlockFrom : void(class TYX.PawsDebugger.IProgPoint)
            RunFrom : void(class TYX.PawsDebugger.IProgPoint)
            Step : void(valuetype TYX.PawsDebugger.DebuggerStepMode)
            ToggleBreakpoint : void(class TYX.PawsDebugger.IProgPoint)
            ToggleRepeatblock : void(class TYX.PawsDebugger.RepeatBlock)
            get_Breakpoints : object()
            get_Modules : string[]()
            get_Repeatblocks : object()
            set_Breakpoints : void(object)
            set_Repeatblocks : void(object)
```

```
.assembly TYX.PawsDebugger.Interop
{
  .ver 1:39:14:1
}
```

The usage of this property is expected any time after a project is loaded (with LoadPaw), otherwise its value is null:

```
public partial class Form1 : Form
{
private TYX.PawsDebugger.Debugger debugger;
[...]
private void LoadClicked(object sender, EventArgs e)
{
[...]
// load a project anytime before this code runs, otherwise Modules
// property is null. After a project is loaded, at least one module (main module)
// is listed in the array.
string sAllModules = "";
if (this.debugger.Modules != null)
{
    foreach (string sModule in this.debugger.Modules)
    {
        if (sAllModules != "")
            sAllModules += ", ";
        sAllModules += sModule;
    }
}
System.Windows.Forms.MessageBox.Show("Modules :" + sAllModules);
[...]
}
[...]
}
```

2.4 Problem Reports

2.4.1 PawsDebugger.dll: Unable to Access the Type and Data Information for Debug Variables (BugID 1104)

Paws Studio is unable to access the type and data information for debug variables if the project is built with "Case Insensitive" in the ATLAS property page, turned on. It is also unable to show the type and data information for debug variables that contain a space in their names.

This release fixed this problem.