TYX CORPORATION

Productivity Enhancement Systems

PAWS Studio Release Notes

Version 1.39.2

February 24, 2011

Table of contents

1 Paws Developer's Studio

Version 1.39.2
Release date: February 24, 2011

1.1 Critical Items

1.2 Known Limitations

1.3 Enhancements

1.3.1 The <usr>\tyx\bin\ConfigureRegistry.exe tool has been changed

Through this change, this tool has become more helpful. The new command line arguments follow.

```
ConfigureRegistry.exe Version 1.39.2.
USAGE:      ConfigureRegistry.exe      -app<[Default:All]|PAWS|WRTS>      -
[Option]<Parameter>
 -[fromINI] update the Windows Registry configuration from INI
 -[toXML]   copy the Windows Registry configuration to <XML File Path>
 -[fromXML] update the Windows Registry configuration from <XML File Path>
 -[clean]   clean the Windows Registry configuration
 Examples:
1> Copy all configuration of TYX Applications to a XML file -->
ConfigureRegistry.exe -app -toXML"C:\Temp\MyConfiguration.xml"
2> Update configuration for PAWS application only from a XML file -->
ConfigureRegistry.exe -appPAWS -fromXML"C:\Temp\MyPawsConfiguration.xml"
3> Update configuration for WRTS application only from an INI file -->
ConfigureRegistry.exe -appWRTS -fromINI
4> Clean all configuration of TYX Applications in the registry -->
ConfigureRegistry.exe -app –clean
```

1.3.2 The Translation of Console.WriteLine 1641 carrier language statement into ATLAS fails if the argument is a constant string longer than 499 chars (BugID 731).

This issue has been fixed. In the present implementation, the translator generates consecutive OUTPUT ATLAS statements that output all the 499 characters long segments that build the original argument of the 1641 Console.WriteLine() instruction.

1.3.3 Signal Input connected to a TSF model through a SignalREF(S) type attribute (ChangeReq ID 732)

This version of our 1641 Carrier Language Translator into ATLAS supports the passage of a 1641signal reference to the internal TSF model definition through an attribute. Traditionally this has been done through the `In`, `Sync` and `Gate` TSF model fields. This traditional approach is still functional.

In order to pass signals through TSF Model attributes, the attributes have to be defined of type `SignalREF` or `SignalREFS`. At this time, just one signal per attribute can be passed.

The following test case demonstrates this approach and shows the resulting ATLAS code. The significant sections are colored in red.

.1641cl code:

```
class ASignal
{
  public void main()
  {
    double d;
    <TPL>
      Setup
      <Signal xmlns:prf000="urn:IEEE-1641:2010:SampleTSFModelsLib"
              Out="mySensorTSF">
        <prf000:SampleSensorTSF name="mySensorTSF"
                                inputAsSignalREF="myUUTPins"/>
        <TwoWire name="myUUTPins"
                 hi="J1"
                 lo="J2"
                 channelWidth="1"/>
      </Signal>
      as s1;
      Read s1 into d;
      Compare s1;
      Reset s1;
    </TPL>
  }
}
```

The TSF Model library:

```
<?xml version="1.0" encoding="UTF-8"?>
<tsf:TSFLibrary name="SampleTSFModelsLib"
                targetNamespace="urn:IEEE-1641:2010:SampleTSFModelsLib"
                xmlns:tsf="STDTSF"
                xmlns="STDBSC"
                uuid="CD3A1631-BD59-4172-B13B-32C20E5EEE73" version="1.0">
  <tsf:TSF name="SampleSensorTSF" uuid="4864F2D5-4AAD-4604-B8F4-88F63A4EC368">
    <tsf:interface xmlns:prf001="http://www.w3.org/2001/XMLSchema">
      <prf001:schema>
        <prf001:element>
          <prf001:complexType>
            <prf001:complexContent>
              <prf001:extension>
                <prf001:attribute name="inputAsSignalREF" type="SignalREFS"/>
              </prf001:extension>
            </prf001:complexContent>
          </prf001:complexType>
        </prf001:element>
      </prf001:schema>
    </tsf:interface>
    <tsf:model>
      <Signal Out="inst">
        <Ins>
          <In name="inputAsSignalREF" />
        </Ins>
        <Instantaneous name="inst"
```

```
                       samples="1"
                       nominal="range 0 V to 10 V"
                       UL="7.3 V"
                       LL="7.0 V"
                       In="inputAsSignalREF"/>
      </Signal>
    </tsf:model>
  </tsf:TSF>
</tsf:TSFLibrary>
```

The generated atlas:

```
C The following 's1.mySensorTSF.inst' 1641 signal is sensor and its 'Generic'
1641 configuration type has been mapped to the 'DC SIGNAL' atlas noun. $
            SETUP, (VOLTAGE-INST), DC SIGNAL,
                VOLTAGE-INST RANGE 0 V TO 10 V,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
C The following 's1.mySensorTSF.inst' 1641 signal is sensor and its 'Generic'
1641 configuration type has been mapped to the 'DC SIGNAL' atlas noun. $
            CONNECT, (VOLTAGE-INST), DC SIGNAL,
                VOLTAGE-INST RANGE 0 V TO 10 V,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
C The following 's1.mySensorTSF.inst' 1641 signal is sensor and its 'Generic'
1641 configuration type has been mapped to the 'DC SIGNAL' atlas noun. $
            FETCH, (VOLTAGE-INST INTO 'd'), DC SIGNAL,
                VOLTAGE-INST RANGE 0 V TO 10 V,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
            COMPARE, 'd', UL 7.3 LL 7.0 $
C The following 's1.mySensorTSF.inst' 1641 signal is sensor and its 'Generic'
1641 configuration type has been mapped to the 'DC SIGNAL' atlas noun. $
            REMOVE, (VOLTAGE-INST), DC SIGNAL,
                VOLTAGE-INST RANGE 0 V TO 10 V,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
```

1.3.4 Non-Sensor Configured Signals may not display default pins if they are roots in the Anonymous TSF model (ChangeReq ID 725)

The Non-Sensor Configured Signals are expected to have at their right side a connecting BSC. A Configured Signal is a TSF model instance or a BSC instance in the anonymous TSF model for which the 1641 carrier language translator into ATLAS has mapped an ATLAS noun selected from the available 1641 configuration files.

Before this release, the translator assumed that the Non-Sensor Configured Signals always worked in a Source configuration, and it supplied default UUT pins if these Signals were not explicitly connected at their right side through connecting BSCs (like TwoWire).

At this time, even in the case when the Non-Sensor Configured Signals are roots in the anonymous TSF model (so they are not connected to connecting BSCs at their right side), the translator may not supply the default pins if the atlas signal has enough input pins to satisfy the atlas syntax.

In the following test case, although the Load BSC is not connected in a Source configuration, no default pins are generated on its right side as long as Load BSC is connected at its input (on its left side) to non-default UUT pins.

.1641cl code:

```
class ASignal
{
  public void main()
  {
    double d;
    <TPL>
      Setup
      <Signal Out="myLoad">
        <TwoWire name="myUUTPins" hi="J1" lo="J2" channelWidth="1"/>
        <Load name="myLoad"
              In="myUUTPins"
              resistance="4.7 kOhm"
              reactance="100 Ohm"/>
      </Signal>
      as s1;
      Reset s1;
    </TPL>
  }
}
```

The generated atlas:

```
C The following 's1.myLoad' 1641 signal is source and its 'Load' 1641
configuration type has been mapped to the 'IMPEDANCE' atlas noun. $
            SETUP, IMPEDANCE,
                RES 4.7 * 1.0E3 Ohm,
                REACTANCE 100 Ohm,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
C The following 's1.myLoad' 1641 signal is source and its 'Load' 1641
configuration type has been mapped to the 'IMPEDANCE' atlas noun. $
            CONNECT, IMPEDANCE,
                RES 4.7 * 1.0E3 Ohm,
                REACTANCE 100 Ohm,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
C The following 's1.myLoad' 1641 signal is source and its 'Load' 1641
configuration type has been mapped to the 'IMPEDANCE' atlas noun. $
            REMOVE, IMPEDANCE,
                RES 4.7 * 1.0E3 Ohm,
                REACTANCE 100 Ohm,
                CNX ATLAS-SGN-P-HI J1 ATLAS-SGN-P-LO J2 $
```

1.4 Problem Reports

1.4.1 PR 11014 LEX License consumed when the Paws Subsets tab is hidden (ChangeReqID 729)

When the Paws Subsets Tab of the Project Workspace is hidden, the Build LEX license is not consumed. In a floating license environment, other Paws Studio applications in the network may consume this license.

1.4.2 PR 11008 Support request - Function GetStr

The essence of this Problem Report is that some unexpected linking errors were noticed when the customer's CEM code was built with the Microsoft Visual Studio 2003 compilers, and the CemEx.lib was an input static library. The solution to this problem is the distribution of our `LegacyCemEx.lib` in our `<usr>\tyx\lib` folder. All `Legacy*.lib` static libraries present in `<usr>\tyx\lib` are intended to be used when C\C++ compilers and linkers older than 2003 are invoked as CEM/NAM/UserBIFs building tools.

2 Run Time System

Version 1.39.2
Release date: February 24, 2011

2.1 Critical Items

2.2 Known Limitations

2.3 Enhancements

2.4 Problem Reports

2.4.1 PR 9113 The PLI process launched by the switch server fails to load the Project.SWX file

The RTS Switch Server failed to load an ITA DB if it was previously compiled with “UUT Wrap Around” options on. This problem was solved.