TYX CORPORATION

Productivity Enhancement Systems

PAWS Studio Release Notes

Version 1.38.3

September 23, 2010

Table of contents

1 Paws Developer's Studio

Version 1.38.3
Release date: September 23, 2010

1.1 Critical Items

1.2 Known Limitations

1.2.1 1641CLCEx.exe: TPL Warning 014 and 015 inserted to expose two 1641CLCEx.exe limitations

The significance of these warnings of our 1641 carrier language and TPL translator into ATLAS follows.

TPL warning 014: The number of samples for the '_identifier_' sensor is assumed 1. This translator does not support values for the attribute 'samples' different than 0 or 1

This is a known limitation of the present translator. The following test case generates this warning.

```
class C
{
    public void main()
    {
        double d;
        <TPL>
        Setup
        <Signal xmlns:prf000="STDBSC" Out="sen1">
          <prf000:TwoWire name="conn1" hi="J1" lo="J2" channelWidth="1"/>
          <prf000:Instantaneous name="sen1" In="conn1" samples="5"/>
        </Signal>
        as s;
        Read s into d;
        Reset s;
        </TPL>
    }
};
```

As a work around this problem, this TPL section can be called repeatedly, in a loop, as below.

```
class C
{
    public void main()
    {
        double d;
        double[5] allResults;
        int i;
        for (i = 0; i < 5; i += 1)
        {
            <TPL>
            Setup
            <Signal xmlns:prf000="STDBSC" Out="sen1">
              <prf000:TwoWire name="conn1" hi="J1" lo="J2" channelWidth="1"/>
              <prf000:Instantaneous name="sen1" In="conn1" samples="1"/>
            </Signal>
```

```
            as s;
            Read s into d;
            Reset s;
            </TPL>

            // use the intermediate reading
            Console.WriteLine(d);
            allResults[i] = d;
        }
    }
};
```

If only the reading was intended to be repeated, the test might look as below.

```
class C
{
    public void main()
    {
        <TPL>
        Setup
        <Signal xmlns:prf000="STDBSC" Out="sen1">
          <prf000:TwoWire name="conn1" hi="J1" lo="J2" channelWidth="1"/>
          <prf000:Instantaneous name="sen1" In="conn1" samples="1"/>
        </Signal>
        as s;
        </TPL>
        double d;
        double[5] allResults;
        int i;
        for (i = 0; i < 5; i += 1)
        {
            <TPL>
            Read s into d;
            </TPL>

            // use the intermediate reading
            Console.WriteLine(d);
            allResults[i] = d;
        }
        <TPL>
        Reset s;
        </TPL>
    }
};
```

TPL warning 015: The ' identifier ' sensor does not support 'Z' (high-impedance) as triggering bit values for a digital event generator. The digital 'Z' (high-impedance) bits are replaced with digital 'L' (low) values

This is a known limitation of the present translator when the digital pattern is involved. The following test case generates this warning.

```
class C
{
  public void main()
  {
    <TPL>
    Setup
        <Signal Out='sen' >
            <DigitalBus
                name='port'
```

```
                pins='D1 D2 D3 D4'
                channelWidth='4'/>
            <ParallelDigital
                name='reference'
                data='HLXZ'
                logic_L_value='0.2 V'
                logic_H_value='5.0 V'/>
            <Measure
                name='sen'
                samples='0'
                attribute='data'
                As='reference'
                In='port'/>
        </Signal>
    as evGen;

    // use evGen to synchronize or gate other signals

    Reset evGen;
    </TPL>
  }
}
```

In order to avoid this warning, the 'Z' bit has to be explicitly replaced with 'L' in the *data* constant:

```
class C
{
  public void main()
  {
    <TPL>
    Setup
        <Signal Out='sen' >
            <DigitalBus
                name='port'
                pins='D1 D2 D3 D4'
                channelWidth='4'/>
            <ParallelDigital
                name='reference'
                data='HLXL'
                logic_L_value='0.2 V'
                logic_H_value='5.0 V'/>
            <Measure
                name='sen'
                samples='0'
                attribute='data'
                As='reference'
                In='port'/>
        </Signal>
    as evGen;

    // use evGen to synchronize or gate other signals

    Reset evGen;
    </TPL>
  }
}
```

1.2.2 1641CLCEx.exe: The TSF Model attributes are always perceived as write-only attributes

In this implementation, while the translator expands the anonymous TSF model and explores the instances of other (named) TSF models in order to resolve the *configured signals tree*, all the attribute values are calculated before the TSF model definitions of its components are investigated. The assumption is that the attributes of all TSF model instances are always write-only. There is never an expectation that such a TSF model instance *returns* through one of its attributes a value as result of an applied action.

In the 1641 Standard, the *measurement* attribute of any *Sensor* is a typical example of a read-only attribute. This policy would allow a TSF Model to return a value through its attribute named "measurement" in a similar fashion like the *Sensor* BSC. The following test case shows this scenario.

```
class C
{
  public void main()
  {
    double ratio;
    <TPL>
    Setup
    <Signal Out="ratiosensor" xmlns:prf000="TSFModelsLib">
      <prf000:MultipleSensors name="ratiosensor"/>
    </Signal>
    as sen;
    Read sen into ratio;
    Reset sen;
    </TPL>
  }
}
```

TSFModelsLib:

```
<tsf:TSFLibrary name="TSFModelsLib" xmlns:tsf="STDTSF" xmlns="STDBSC">
  <tsf:TSF xmlns:tsf="STDTSF" name="MultipleSensors">
    <tsf:interface xmlns:prf000="http://www.w3.org/2001/XMLSchema">
      <prf000:schema>
        <prf000:element>
          <prf000:complexType>
            <prf000:complexContent>
              <prf000:extension>
                <prf000:attribute
                name="measurement"
                type="Ratio"
                default="v1.measurement / v2.measurement"/>
              </prf000:extension>
            </prf000:complexContent>
          </prf000:complexType>
        </prf000:element>
      </prf000:schema>
    </tsf:interface>
    <tsf:model>
      <Signal xmlns="" Out="v1 v2">
        <TwoWire name="v1Pins" hi="V1-1" lo="V1-2" channelWidth="1"/>
        <Average name="v1" In="v1Pins" samples="1" nominal="MAX 10 V"/>
        <TwoWire name="v2Pins" hi="V2-1" lo="V2-2" channelWidth="1"/>
        <Average name="v2" In="v2Pins" samples="1" nominal="MAX 10 V"/>
      </Signal>
    </tsf:model>
  </tsf:TSF>
</tsf:TSFLibrary>
```

This translator expects to calculate the default values for all its input attributes before it resolves the signals present in its TSF model definition. In the case above, it expects to calculate the default value of the measurement attribute as an input value for the TSF definition, before it resolves the ATLAS nouns for *v1* and *v2*.

This translator lists the missing feature of dealing with read-only TSF model attributes as a known limitation. Despite this shortcoming, it still performs the essence of the above measurement in a slightly different fashion, by having the ratio computation calculated outside of the TSF model, in the carrier language area:

```
class C
{
  public void main()
  {
    double ratio, d1, d2;
    <TPL>
    Setup
    <Signal Out="ratiosensor" xmlns:prf000="TSFModelsLib">
      <prf000:MultipleSensors name="ratiosensor"/>
    </Signal>
    as sen;
    Read sen into d1, d2;
    Reset sen;
    </TPL>
    ratio = d1 / d2;
    Console.WriteLine(ratio);
  }
}
C The following 'sen.ratiosensor.v1' 1641 signal is sensor and its 'Generic' 1641
configuration type has been mapped to the 'DC SIGNAL' atlas noun. $
            FETCH, (VOLTAGE-AV INTO 'd1'), DC SIGNAL,
                VOLTAGE-AV MAX 10 V,
                CNX HI V1-1 LO V1-2 $
C The following 'sen.ratiosensor.v2' 1641 signal is sensor and its 'Generic' 1641
configuration type has been mapped to the 'DC SIGNAL' atlas noun. $
            FETCH, (VOLTAGE-AV INTO 'd2'), DC SIGNAL,
                VOLTAGE-AV MAX 10 V,
                CNX HI V2-1 LO V2-2 $
…
C the beginning of the assignment section $
            CALCULATE, 'ratio' = ( 'd1' / 'd2' ) $
C the end of the assignment section $
            OUTPUT, 'ratio' $
```

1.3 Enhancements

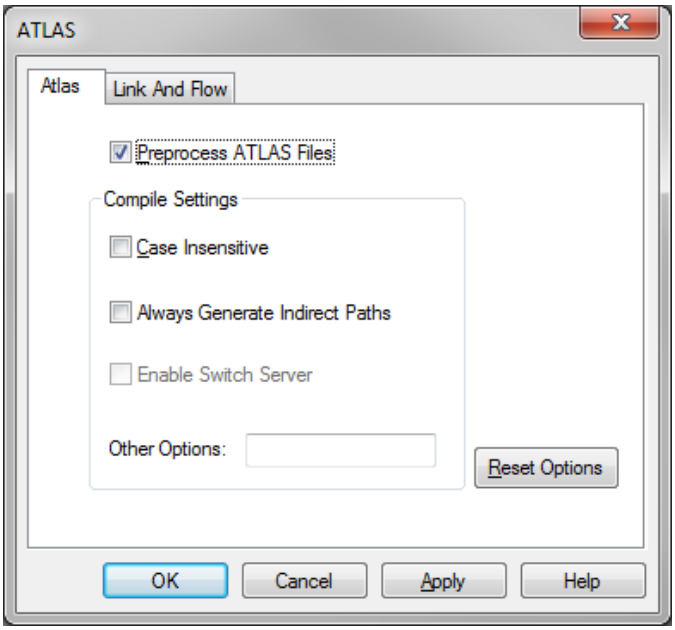
1.3.1 1641CLCEx.exe supports STDTSFLib.xml ver 2.01

Our present 1641CLCEx.exe carrier language and TPL translator into ATLAS supports the new STDTSFLib.xml. Its TSF Models are expected to be used under the “urn:IEEE-1641:2010:STDTSFLib” namespace. If both the olb and the new STDTSFLib.xml model libraries are expected to be used side by side, the old STDTSFLib.xml exposes its TSF models under the “STDTSFLib” namespace, while the new one exposes its TSF models under its new namespace.

1.3.2 Studio able to use a customized ATLAS Preprocessor

The set of rules that define the usage of a customized ATLAS preprocessor follows:

1)The preprocessor is invoked if only the settings of the ATLAS module has the Preprocess Atlas Files checkbox checked.



2)The preprocessor is invoked from Paws Studio right before the ATLAS compiling step. If the ATLAS module contains AsModules, the AsModules source files are concatenated, and fed as inputs to the preprocessor. The output of the preprocessor is the input file for our ATLAS compiler. Any preprocessor should not break the lines alignment in between its input and output files. A genuine ATLAS compiler error detected in the output file of the preprocessor is shown in the input file of the preprocessor, when the user double-clicks on the error in the Output Window of Paws Studio.

3)The preprocessor has to have the following name: <usr>\tyx\bin\ATLAS<subset><station>Preprocessor.exe, where <usr> is c:\usr, and <subset> and <station> are the names of the current subset and station of the loaded project. For a IEEE716.89\PAWS station, the preprocessor name would be c:\usr\tyx\bin\ATLASIEEE716.89PAWSPreprocessor.exe.

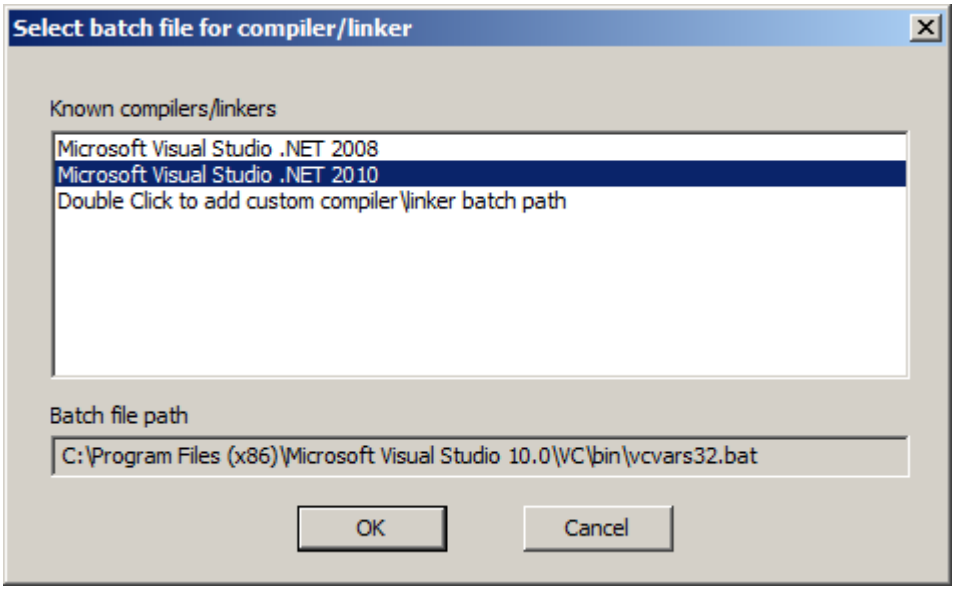
4)The first argument of this preprocessor represents all the ATLAS input files, at the same time. This allows the preprocessor to handle linking errors.

5)The second argument of this preprocessor defines the output folder of the preprocessed files. The short file names of the output files are the short file names of the input files, whose extensions are replaced with ".pproc"

6)The third argument (which is the last argument) of this preprocessor defines the report file name. The preprocessor is expected to write in this file the user-messages that get displayed at building time in the Paws Studio Output Window.

1.3.3 CEM modules built with Microsoft Visual Studio 2010 C++ compilers

In Paws Developer's Studio, the user selects a CEM module, right click, Settings. When the "Batch File That Sets the Compiler\Linker Environment Variables" control needs to point to the Microsoft Visual Studio 2010 script, the user has to click Browse, and the Microsoft Visual Studio 2010 is visible in the compilers list as an option.



1.4 Problem Reports

1.4.1 PR 10037: LiteDebugger property sheet is not visible when RTS is not attached.

This problem was fixed.

2 Run Time System

Version 1.38.3
Release date: September 23, 2010

2.1 Critical Items

2.2 Known Limitations

2.3 Enhancements

2.3.1 TYX.ATMLTestResultsLogger.Recorder generates valid 2010 schemas documents

When the RTS is configured to use the TYX.ATMLTestResultsLogger.Recorder datalogger, it generates 2010-schema compliant documents. The updated schemas are present in c:\usr\tyx\datalog\TestResults\Schemas. The place where the documents are created is c:\usr\tyx\datalog\TestResults.

2.4 Problem Reports

2.4.1 PR 10060: Test Diagram Generator fails to load when SigBase turns on its RTS

When SigBase turns on the RTS to run a Test Procedure, and the RTS is configured to use its Test Diagram Generator, the latter failed to load the project switch database.

This problem was resolved.