Tester general utilities

# Tester basics

## Introduction

### Abbreviations

|  |  |
| --- | --- |
| ATR | Acceptance Test Results |
| EUT | Equipment Under Test |
|  |  |
|  |  |
|  |  |
|  |  |

### Location

The tester is available as a public repo at GitHub:

https://github.com/yahalit/TesterSkeleton.git

### How testing works

After the testing starts, all the checked tests run in sequence.

Each test has a descriptor, which defines among other things the specific handler function that implement the test. All the test functions are in the form

public bool TestSomething(CTestIdentifier a)

A test identifier has the following members:

public Func<CTestIdentifier, bool> TestFunc; // The function that runs the test

public CTestInfo TestInfo; // The test information

public List<string> ErrMsg; // List of error messages accumulated in the test

public int RowIndex ; // Location of result in the ATR

public string ExcelFileName = null ; // Name of the ATR file

public string TempFolder = null; // Folder to store temporary results

Of them the TestInfo struct summarizes:

public string TestIdentifier; // Test identifier, like 1.2.3

public string TestHeader; // Header for the test in the ATR

public string TestSheet; // Sheet in the ATR for test results

public string ErrorMessage; // Error message, in case of test failure

The handler function executes some sequence, and the outcomes may be:

* The procedure failed (e.g., some relevant communication could not be completed) , then the function returns false, with partial or no ATR fill. The test sequence will stop.
* The procedure is complete, the results obtained (which may be several ATR lines) are filled, as Pass, Fail, or NA. If there was a failure, and the sequencer is instructed to stop on failure, the test sequence will exit. Otherwise, the tester shall proceed to the next test.

On completion of the entire sequence, the tester will run the statistics log (how may tests of the total succeeded, failed, were NA, or not executed). If the sequence was not aborted, the resulting ATR file will be printed as PDF.

## The ATR file

### ATR file name

All the ATR files must be named as

[PROJ\_NAME]\_ATR\_SN\_[EUT Serial number]\_[year]\_[month]\_[day]\_[revision]

|  |  |
| --- | --- |
| PROJ\_NAME | Project name string like “PVS” |
| EUT Serial number | An integer in the range 1 to 100000 |
| Year | An integer in the range 2000-2200 |
| month | An integer in 1..12 |
| Day | An integer in 1..31 |
| revision | An integer in 1..1000 |

The year month and day represent the first generation of the ATR.

The revision may advance if tests are added, completed, or re-iterated

# Structs and Classes

|  |  |  |
| --- | --- | --- |
| AtrAttrib (struct) | The attributes of an ATR file | As parsed from its name |
| CComId (struct) | The attributes of a COM port |  |
| XLSGraph (class) | Static class – serves to add graphs to the ATR |  |
| CTestInfo (struct) | Location information for test in the ATR form |  |
| CTestIdentifier (class) | Identifier of a single test | Includes CTestInfo |
| CAtrStatistics (struct) | Summary of ATR passes, fails, etc. |  |
| CTestForm (struct) | The formal details in the ATR front page |  |
| AtrBoundSpecifier (struct) | User-driven tolerance boud specifier |  |

# C# Utilities

## COM ports

|  |  |  |
| --- | --- | --- |
| Utility | Description | Reference |
| Gadgets  UpdateAvailableComPorts | With a combo box as argument, update into it all the found COM ports | ‎2.1COM ports |
|  |  |  |
|  |  |  |
|  |  |  |

## Graphics utilities

|  |  |  |
| --- | --- | --- |
| Utility | Description | Reference |
| Gadgets  SetLedColor | Set the color for an indication LED |  |
| Gadgets  CalculateDropdownWidth | Calculate the width of a combo box required to display all its strings |  |
| Gadgets  AddToolTip2Control | Add a tool tip explanation to a windows control |  |

## Excel file utilities

|  |  |  |
| --- | --- | --- |
| Utility | Description | Reference |
| Gadgets  OpenInExcel | Open an Excel file by launching an excel application |  |
| Gadgets CheckLegitimateAtrFileName | Test that an ATR Excel file name is legitimate, and parse its attributes |  |
| Gadgets  IsExcelFileAccessible | Verify if a an excel file of a given name may be opened |  |
| Gadgets  SelectExcelFile | Select an Excel file from a given folder by an Open File dialog |  |
| Gadgets  IncrementFileVersion | For a given ATR file name, increment the version field |  |
| XLSGraph  CreateZedGraphChart | Draw a Graph and save it as PNG |  |
| XLSGraph  InsertImageIntoExcel | Put a PNG image into an Excel file |  |
| XLSGraph  PrintExcel | Print the Excel ATR file as PDF |  |

## Testing utilities

### General numeric

Static class VecOps:

|  |  |  |
| --- | --- | --- |
| Utility | Description | Reference |
| Any() | Find if a bool array contains any true |  |
| MaxAbs() | Find the maximum absolute value in a vector |  |
| Rms() | Find RMS of values in a vector |  |
| LinearRegression() | Fit a line coefficients[0] + coefficients[1] \* xdata ydata |  |
| CheckLinearRegression() | Check how well a linear relation fits xdata to ydata |  |
| CheckUnityRegression() | Check how well xdata matches ydata |  |
| CheckConstantFit() | CheckConstantFit: Check how well xdata \* level matches ydata |  |
| MeanOfRecords | For a jagged array of measurement sets, find the mean measurements set |  |

### Electrical testing

Static class VecOps:

|  |  |  |
| --- | --- | --- |
| Utility | Description | Reference |
| IsDigitalLow () | Does an analog voltage represent a Digital 0 (below 0.25 of logic supply) |  |
| IsDigitalHigh() | Does an analog voltage represent a Digital 1 (above 0.6 of logic supply) |  |

### User interaction

|  |  |  |
| --- | --- | --- |
| Utility | Description | Reference |
| CAtpMainWindow  GetUserValue() | Display an instruction;  Get a value (numeric or text), and/or check |  |

### ATR file handling

|  |  |  |
| --- | --- | --- |
| Utility | Description | Comment |
| CAtpExcel  SetResultInAtr | Write test result in the ATR form and evaluate Pass if necessary | Overloaded. Take bonds from Excel or dynamically specified |
| CAtpExcel  SetNAInAtr | Set a result in the ATR is Not Applicable |  |

# Place Holders

|  |  |  |
| --- | --- | --- |
| Interpreter.  GetVersionOffline() | Read the version of the EUT via ATP communication |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Communications

## COM ports

COM ports are associated with USB-UART adapters. They are generic software drivers that enable UART communication via USB.

The vendors of USB-UART adapters provide device drivers; these drivers must be pre-installed for the COM port to function.

COM ports are identified by:

* COM identifier string (like COM1, COM11). These are assigned by the operating system. The same USB physical device, when connected to different computers, or even when reconnected to the same computer, may be allocated different COM identifiers. They cannot be used as reliable identifiers in a tester setup
* Adapter identifiers. These are unique numbers given by the vendor to the adapters. Adapter identifiers should be marked (e.g., by a visible sticker) with their identifiers.

# Building a tester

## Introduction

This chapter gives step by step instructions to build a test.

Building a test involves editing a skeletal C# program.

As navigation aids, the code includes entry points like

// !! Do NOT remove or chage this Entry Point comment !!

// [EP1] Create a list of all the tests to perform

////////////////////////////////////////////

That you may easily find, E.g., search [EP1] for the above entry point.

## Defining tests skeletons

### Create set of test functions

First you create a list of tests.

Each test is a stand-alone sequence, testing something specific. It may be fully automatic or involve tester intervention. Following the sequence, at least one entry in the ATR form is generated.

The entry may be a number (like a measured voltage) that must be within a range to pass, or approval that something (E.g. a LED went ON) happened as planned to pass.

Next you edit the file AtpTests.cs

The file should include all (and only) the test skeletal functions. Each function looks like

public bool TestSomething(CTestIdentifier a)

{

bool TotalPass = true; // Optimistic

// Test body that will write something into the ATR form and set “TotalPass” // false on any test failure

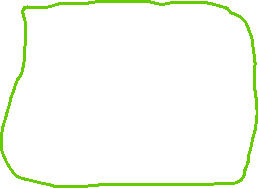
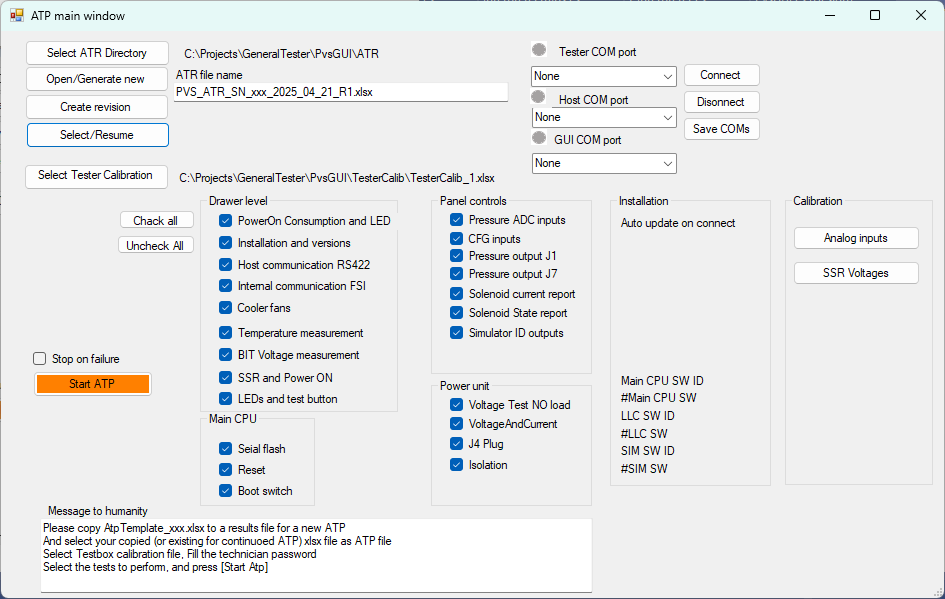
return TotalPass;

}

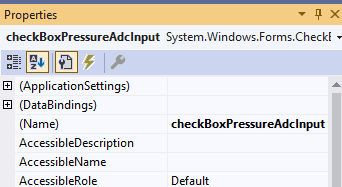
### Edit the main ATP form

Group your test functions into functional groups (it makes sense that each group will write its own sheet in the ATR)

In the ATP main form, AtpMainWindow.cs, edit the contents of the green circumference.



Replace the function group panes with your own, and place inside your own checkboxes, with your own texts and give them appropriate names.



Then, create a list checkboxes, assigning each to a test function. You will find the example at [EP1].

Each test you enter will look like

TestCheckboxList.Add(CheckTestSomething);

Then tag each of the checkboxes with the associated test information [EP2].

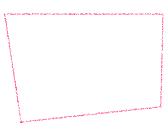
Each tag assignment will look like

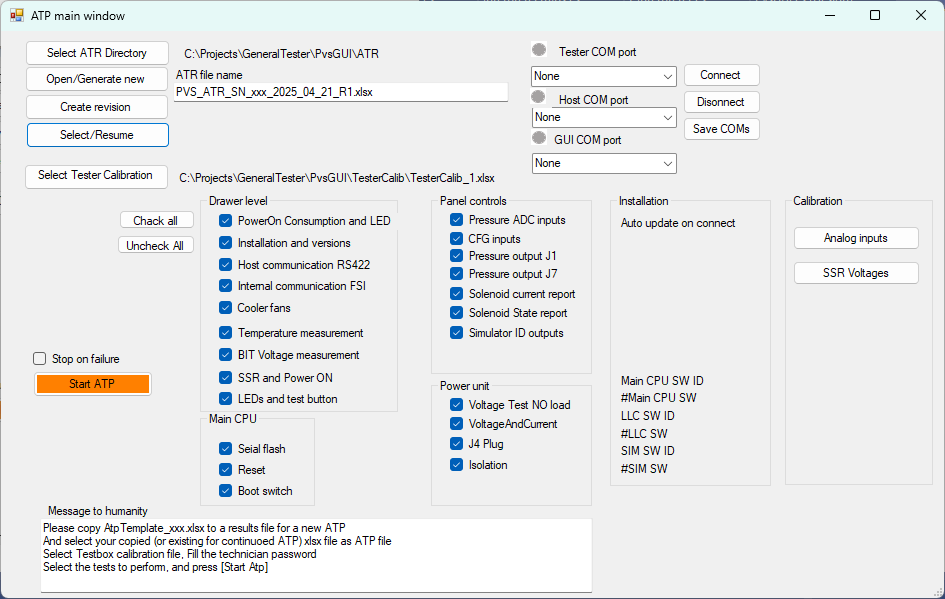
CheckTestSomething.Tag = new CTestIdentifier(TestSomething, "1.0.", Sheet: "Drawer Level tests", Header: "Power On, Power consumption, Power ON LED");

You can add tool tip helps to your checkboxes using something like

Gadgets.AddToolTip2Control(CheckTestSomething, new string[] {"This is what","I want"});

Next [EP3] you should redefine the communication selection combo boxes.





### Define calibration routines

In the calibration pane, set buttons for each required calibration procedure.

### Technician password

The tester may be free to run or require password.

If you want password protection, set its necessity and the password and a clue to itself at [EP4]

A password shall be demanded one time. Further testing will not require additional passwording.

## Building a distributable