Ytester

Getting Started with Ytester

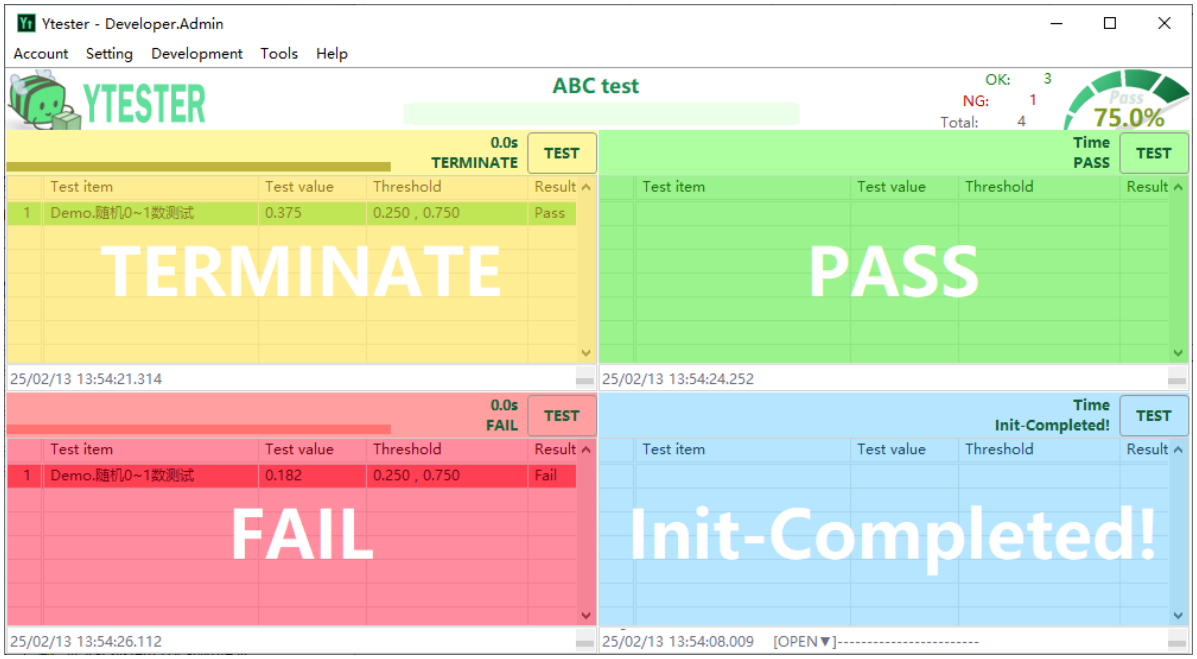
YYLL

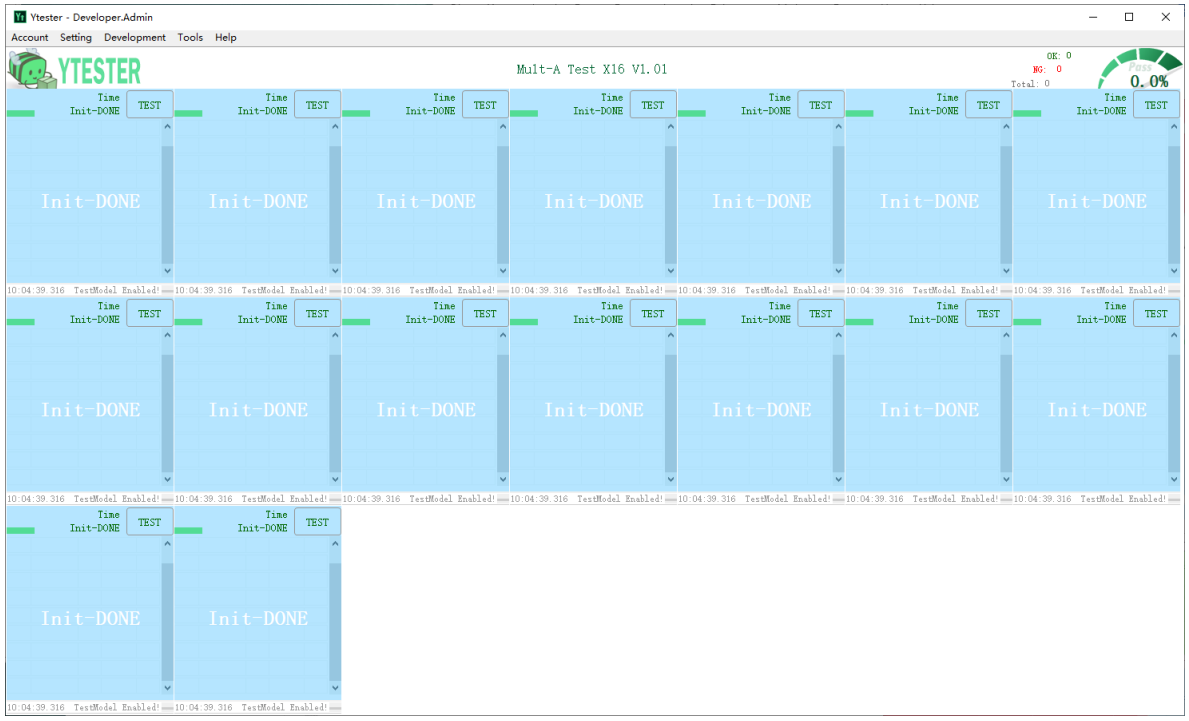
Contents

一、简介

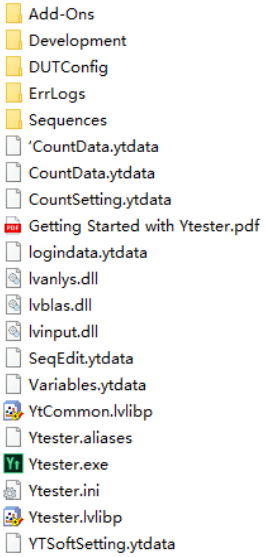
Ytester 是由 YYLL 独立开发的一款实验性的、支持二次开发的简单通用产测内核库软件（Ytester.lvlibp+YtCommon.lvlibp），以打包库的方式与用户开发的应用程序相结合发布使用，以下展示均为由 YYLL 提供 YtesterCreator.lvproj 项目打包发布的应用程序界面。

界面预览





目录结构

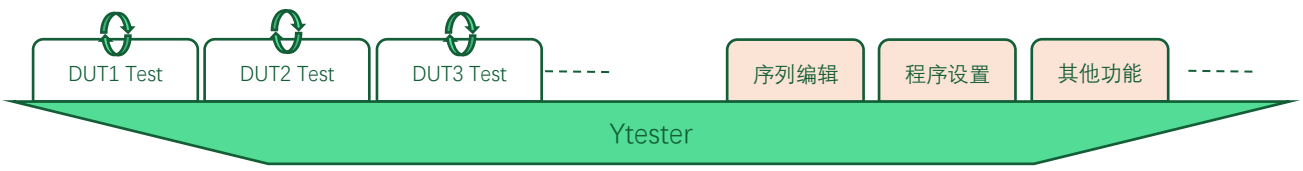


Ytester 目录结构如上图所示，以下表格描述其部分功能：

文件、文件夹	描述
Add-Ons	添加到菜单的插件文件夹
Development	存放二次开发步骤文件夹
DUTConfig	保存 DUT 测试单元配置文件夹
ErrLogs	保存报错文件夹
Sequences	保存序列文件夹
* .ytdata	Ytester 数据文件

二、构架

Ytester 是由多个 DUT 测试单元（预设 16 个子面板模式、或无限个子窗口模式)、人机交互接口 (键盘、扫码枪、鼠标输入)、Ytester 设置、DUT 测试单元设置、重载 DUTs 测试单元、卸载 DUTs 测试单元、账户管理、变量操作、序列开发、统计、工具插件等组成的简单通用产测软件。其构架如下：



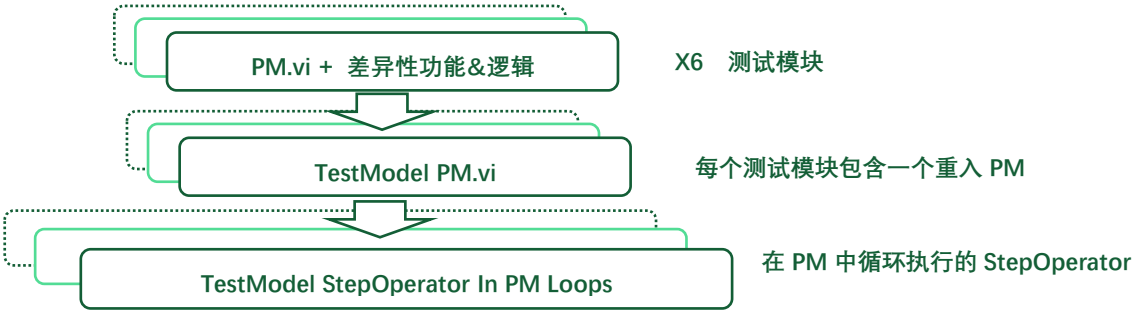
现介绍几个主要的功能模块：

DUT 测试单元

DUT 测试单元包含 UI、与 Yt 主程序交换等线程，及与测试相关 6 个测试模块构成。如下图所示 6 个测试模块：

在 6 个测试模块中开放了自定义步骤集，可由序列开发调试器编辑运行的步骤。LOOPREADY 模块用于每次测试前准备执行（在 UI 更新上不算做测试），也决定是否执行后面 3 个模块；TEST 模块是测试输出核心；OVERSAVE、CLEANUP 模块用于辅助测试；OPEN、CLOSE 模块只在 DUT 载入&释放时执行一次。

DUT 测试单元结构中 6 个测试模块亦可按下图理解：

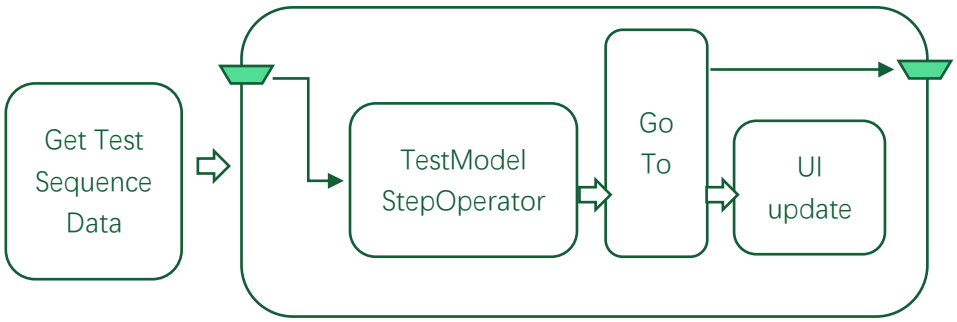


测试模块

测试模块由重入的 TestModel PM+差异性功能&逻辑构成；

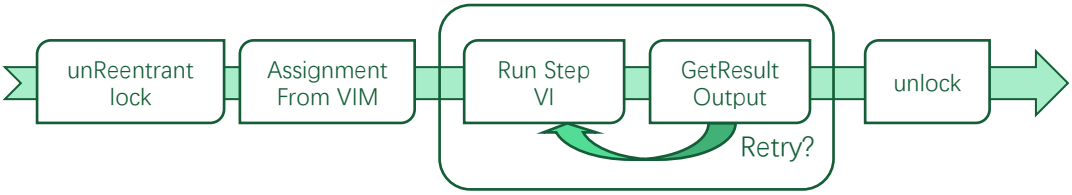
TestModel PM

TestModel PM 负责整个测试执行逻辑，加载用户编辑好的序列，按照测试逻辑去调用 TestModel StepOperator、对应的模块、UI 界面更新等等。



TestModel StepOperator

TestModel StepOperator 是实际步骤执行者，每个二次开发的步骤本质是包含特定控件的 VI，可基于 Development\ Steps\ StepTemplate.vit 模板创建，详见 Development 下《开发指南》文档。



VIM

为了加快测试，在此引入 VIM，其将在测试前会加载好步骤及其开放控件引用进入内存当中，VIM 处理 Step（VI）在内存中预加载&释放。其结构如下：

变量系统

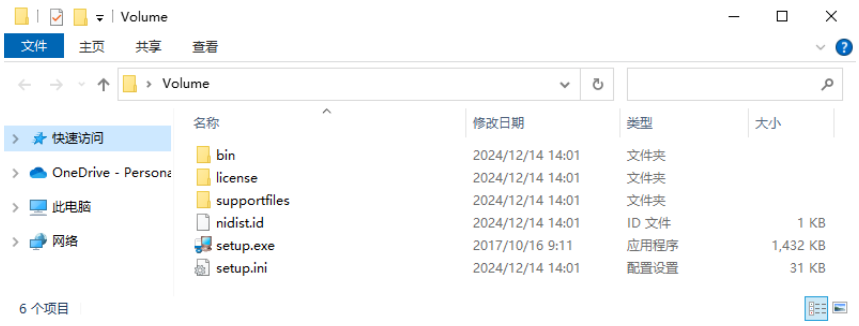
Ytester 存在自己一套的变量系统（存在于 YtCommon.lvlibp）。每一个 DUT 测试单元将分配独立的变量池，即不同的 DUT 测试单元中支持同名的变量，也可调用其他 DUT 测试单元的池中变量。Ytester 定义池“0“为全局变量，”1“为 DUT1 单元的池，以此类推。

其变量分为本地变量（创建时会保存在本地），临时变量（在运行中存在）。以“_”开头的变量表示在开始测试前会将其删除；以“#”开头的变量用于在开发时，通过功能按钮手动保存到本地方便观察开发，在正常测试时以临时变量方式使用，避免在测试中频繁向硬盘写入。

三、安装

Yt Runtime 安装

Ytester 运行需要安装 Yt Runtime 引擎支持。对已经安装过 Yt Runtime 引擎的电脑不需要再安装引擎可直接运行 Ytester。(当然，目标电脑存在 Labview2017RTE (或更高版本的 RTE) (32 位) 及 VISA、488-2 等也不需要再安装引擎）Yt Runtime 是一个安装包程序，执行 setup.exe 进行安装。



Ytester 安装

安装完成 Yt Runtime 后，只需要拷贝 Ytester 文件夹进目标电脑任何位置打开即可使用。

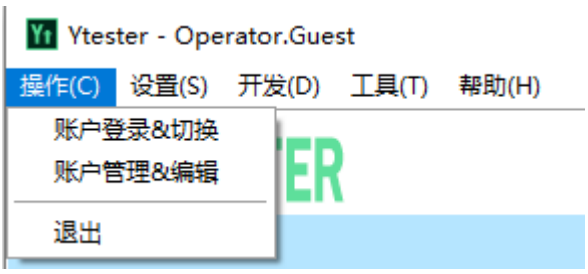
四、使用

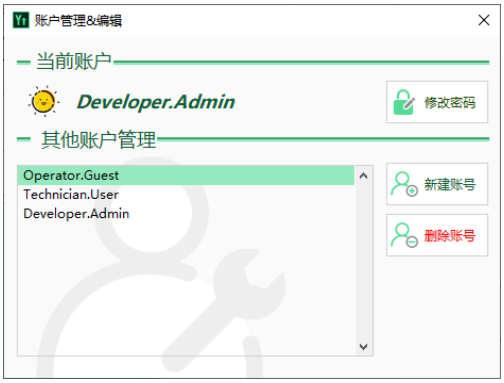
权限账户操作

第一次打开 Ytester 时会弹出登录界面。

本软件分为 3 类权限对应：操作员、技术员、开发者。目前技术员、开发者权限等同无区别，可在登陆高级权限后，在 Ytester 主设置中可以设置，是否使用不同类型权限及账号进行自动登录，如下：

在登录后点击菜单“操作”-“账户管理&编辑”可进行密码修改、删除、新建，点击菜单“操作”-“账户登录&切换”可对账户进行切换。

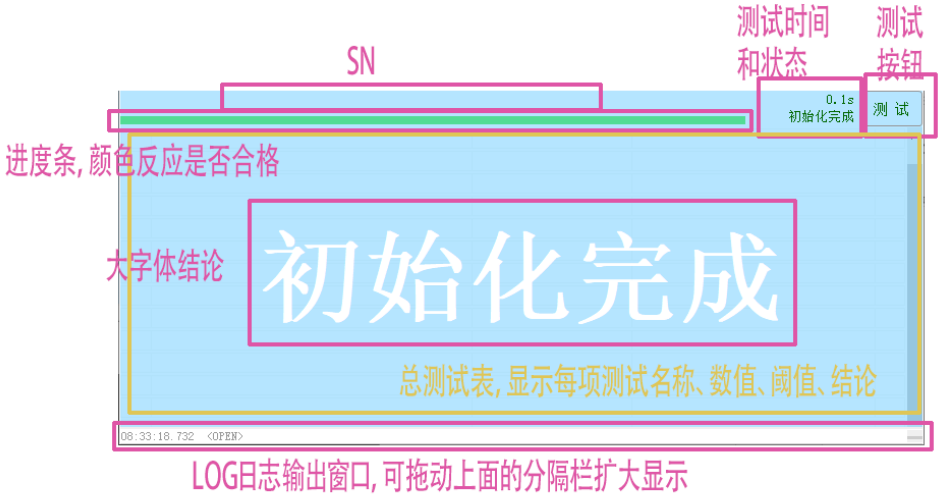




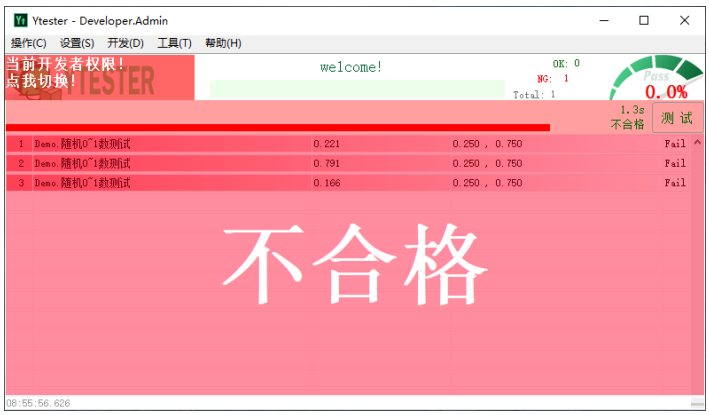
软件界面简介

当用户成功登录后，软件界面分为以下几大部分如下：

“DUTs 加载区”由 1 个或者多个 DUT 窗口构成，每个 DUT 窗口构成如下：



测试效果如下：

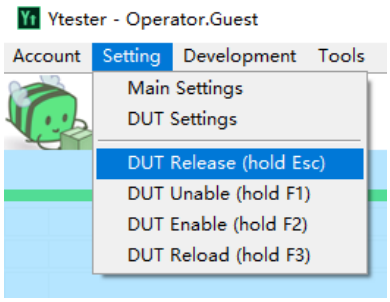


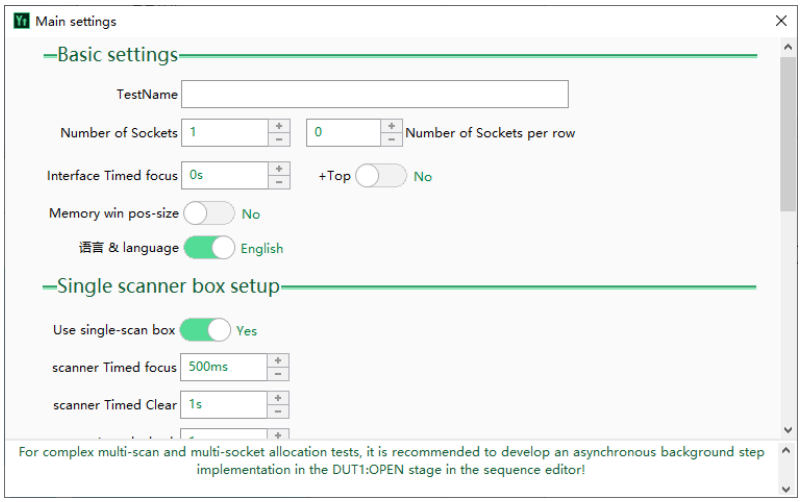
不合格界面

合格界面

软件设置

点击菜单设置，可以对 Ytester 主程序和 DUT 测试单元进行设置





以下简单介绍：

1) TestName 为 Ytester 主程序界面中间上方显示的本测试站名称含义；

2) Number of Sockets 为需要使用多少个 DUT 测试单元，右边控件决定测试单元排版；

3) Interface Timeed focus 为非测试时主程序周期获取焦点置顶；

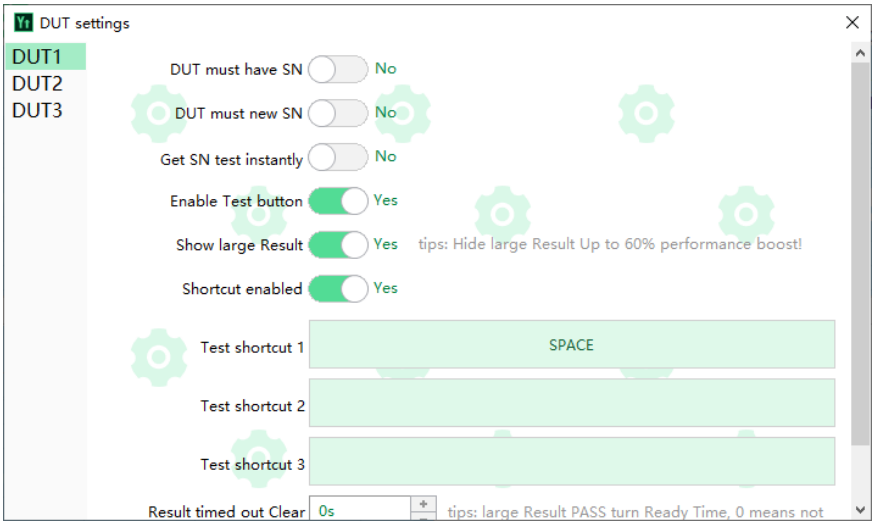
4) Memory win pos-size 为启用每次打开位置是上次关闭的位置；

5) 语言&language 为切换语言；

6) 。。。。。。

主程序设置中在窗口底部有详细的功能介绍，此处不再赘述。如下图在 DUT 模型配置中，可对

每个 DUT 测试单元进行独立设置，包含 3 个测试触发快捷键、是否需要 SN、是否显示大字体结论、等等。相关设置浅显易懂此处简单介绍，不再赘述。

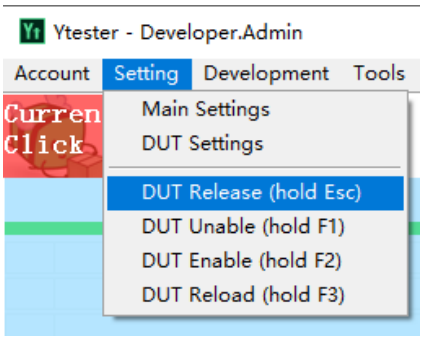


DUT 测试单元重载&释放&禁用&启用

在设置菜单中，DUT 重载是释放当前 DUT 测试单元后再重新加载 DUT 测试单元，这样目的可以重新执行 OPEN、CLOSE 模块；

DUT 卸载（长按 ESC）是仅释放当前全部 DUT 测试单元，方便用户点击开发。

DUT 禁用启用类似；

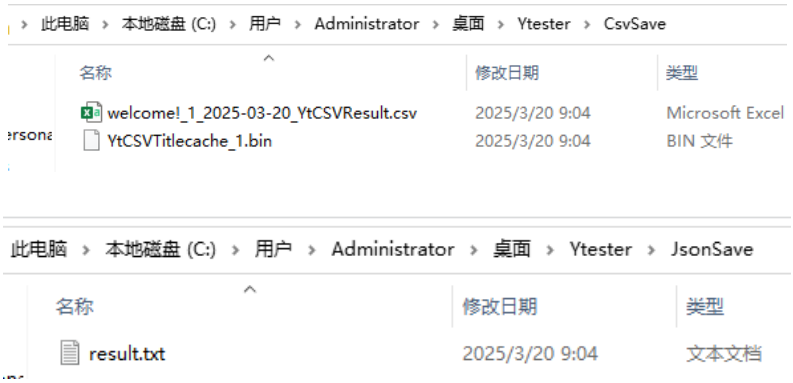


注意：在程序进行二次开发过程中往往存在不合理设置及意外，可能会导致卡死、无响应、互锁，因此在软件打开后会弹出停止加载 DUT 测试单元对话框，通过点击停止 DUT 测试单元加载，也可以通过**长按 ESC**，终止正在运行的 DUT 测试单元，经由序列开发调试器查找问题，避免意外。如下图所示软件打开时延迟加载 DUT 测试单元弹框：

测试数据的保存

测试数据如何保存是交由用户开发的步骤来实现，目前可以使用由作者默认提供开发的 2 个保存 STEP，在序列开发器中选择“OVERSAVE”测试阶段，添加“CSV 式保存”、“ JSON 式保存(LC mes)”2 个步骤；

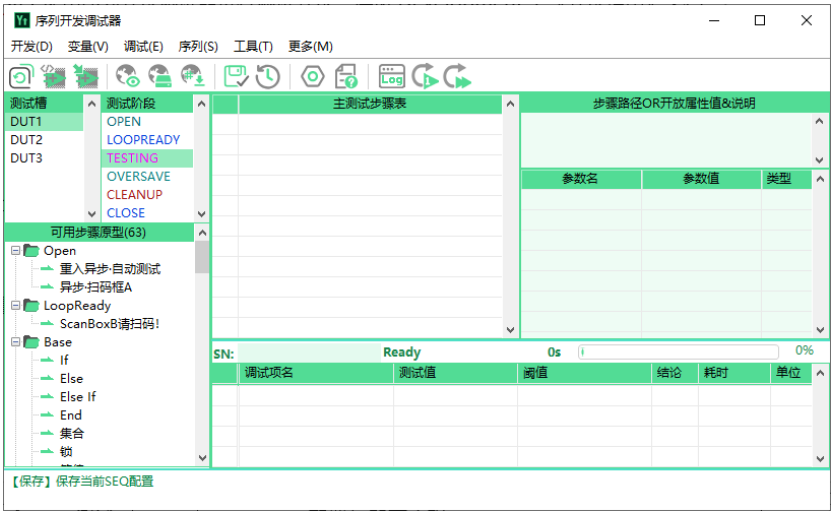
在测试完成后，会在更目录下生成测试记录，用户可以开发不同 STEP 适应自己需要的保存格式。



开发

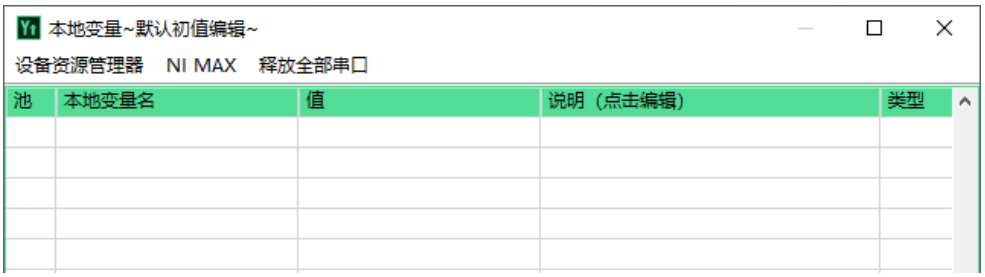
Ytester 使用序列开发调试器进行测试步骤的开发，序列开发调试器是集成开发&调试的 Yt 重要

模块，具体使用方法详见 Development 下《开发指南》文档。



本地变量配置

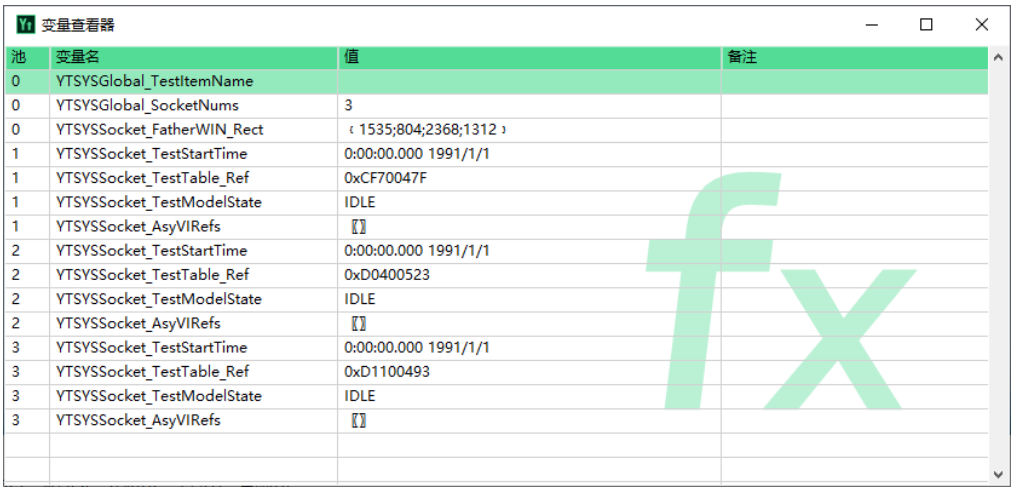
对本地变量进行修改，备注也可以修改，如下图所示。



以下情景中，在用户更换电脑移植程序后，或串口设备更换后，通过打开此窗口，点击菜单中“释放全部串口“解除对串口的占用。再通过菜单打开设备资源管理器，检查串口情况，对串口变量进行值修改。修改后自动生效保存在本地，此表格支持多选操作。

变量预览

对变量进行预览，如下图所示。



以下简单描述部分变量含义：

1) YTSYSGlobal_TestItemName 为测试站名称；

2) YTSYSGlobal_SocketNums 为载入 DUT 测试单元数量；

3) YTSYSSocket_TestStartTime 为测试开始时间；

4) YTSYSSocket_TestTable_Ref 为 DUT 测试单元测试表引用；

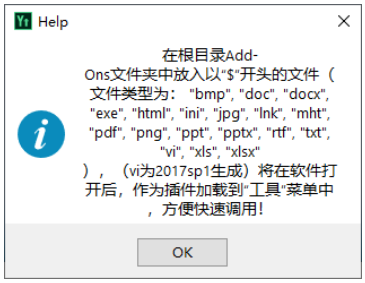
5) YTSYSSocket_TestModelState 为 DUT 测试单元执行状态；

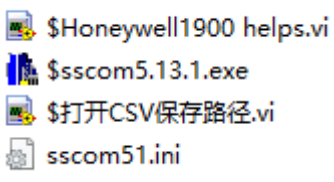
6) YTSYSSocket_AsyVIRefs 为 DUT 测试单元异步 Vis 引用收集，用于测试结束后 CLOSE 阶段释放这些异步的 Vis；

7) ……

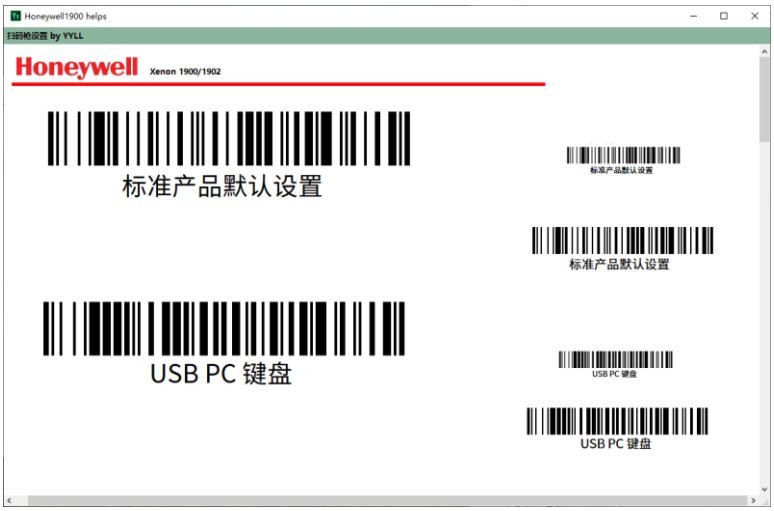
插件

Ytester 支持用户添加特定文件到 Add-Ons 文件夹，这类文件将添加到 Ytester 的工具菜单中方便调用。Ytester 默认添加了 3 个如图所示插件。





如 Honeywell1900 helps.vi 提供该系列扫码枪的功能切换；sscom 提供第三方串口调试工具；用户可以自行拓展插件；

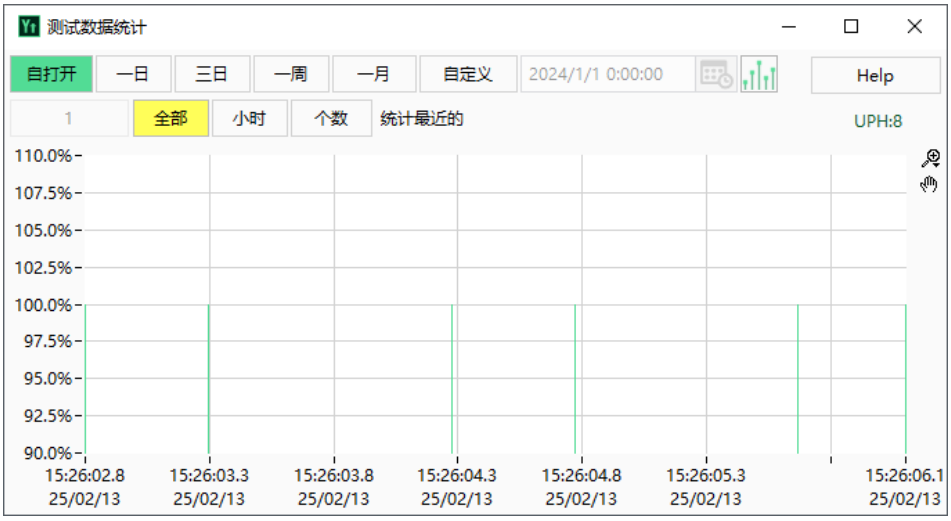


测试数据统计

Ytester 支持对测试数据统计及回溯合格状态，其中对不同合格率会改变颜色。

对于终止的测试默认不记录合格率，可以通过 Ytester 主设置中开启对终止测试记录为 NG。

可以点击如下图图标打开或菜单中打开，测试数据统计界面如下：



该界面第一行表示总体统计开始的时间，其影响主界面上的统计显示。第二行表示下方图表要显

示多少，可以是最近 xx 小时的合格走势，或最近 xx 个测试的情况；右侧简易统计了最近一小时的 UPH。点击 HELP 会解释图表如何方便查看过往合格走势帮助分析。

LOGO 修改

界面 LOGO 修改方法:软件开机读取根目录下 interfacelogo.xxx 文件(xxx 为.png/.jpg/.bmp)，未找到使用默认 LOGO 界面如下，图片大小建议为 240X48 左右。

帮助

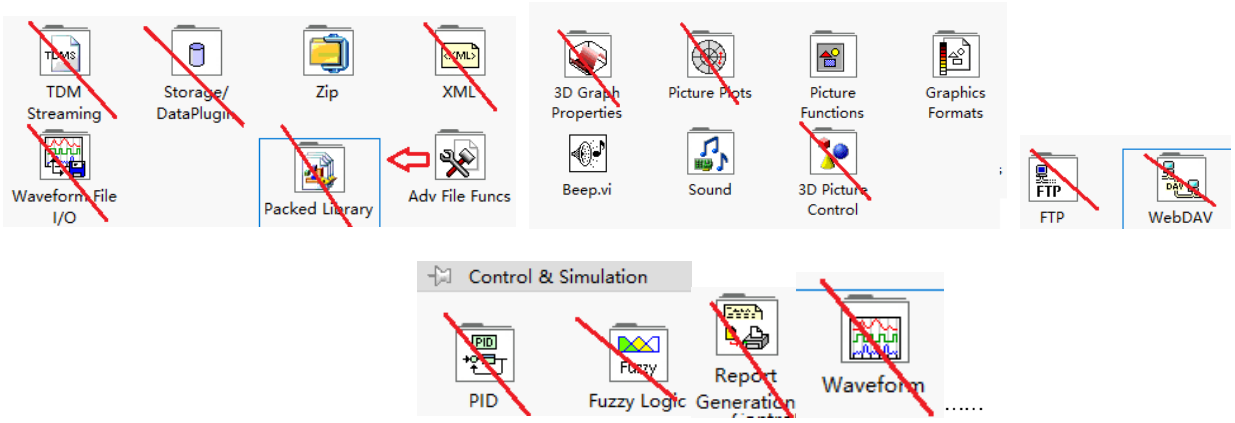
在帮助菜单中可以查看软件历史及版本。

五、二次开发

关于步骤 VI 的开发，详见 Development 下《开发指南》文档。

开发须知：步骤 VI 中子 VI 依赖问题

在 Labview 开发环境中，编程面板中的很多很多很多很多 VI 是不包含在 RTE（引擎）中的。（基本能打开的就是不在引擎中），绝大部分存在于 LabVIEW 根目录 instr.lib、user.lib、vi.lib 中。



Ytester 经由 RTE（引擎）执行，作为支持二次开发的平台，无法预料用户开发的 STEP 中子 VI 依赖问题，这导致用户开发的 STEP 无法执行。

比如在开发模式下可以运行，Ytester 中无法加载的步骤。其根本原因是子 VI 依赖问题，labview 会加载 instr.lib、user.lib、vi.lib 中的依赖，而 Ytester 无法加载这些类<vi.lib>路径开头的子 VI，

Ytester 也没有开发类似 TestStand 提示用户提供缺失子 VI 路径功能。

这个问题一直存在，如下链接提及：

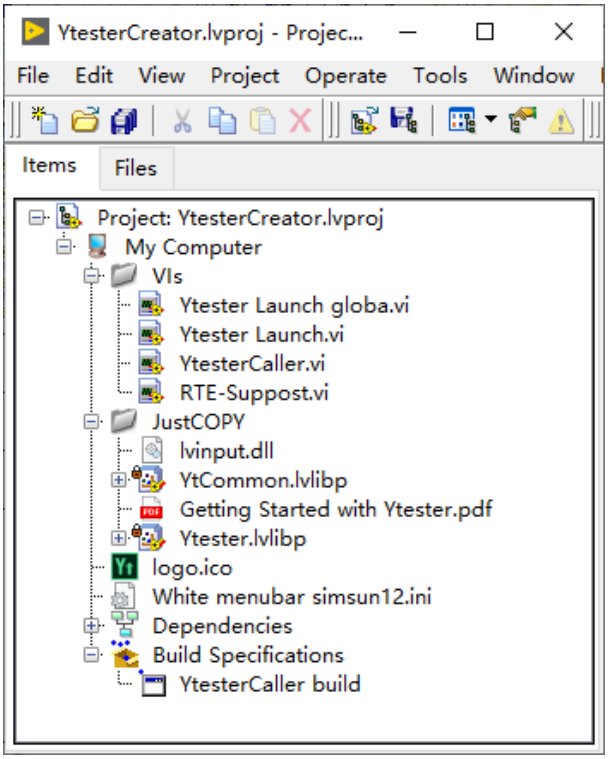
https://lavag.org/topic/3292-calling-an-external-labview-vi-from-an-exe/

当用户使用到的步骤 VI 故 Ytester 最开始采用的是预先封装好这些 VI（某些 labview 安装自带的 VI 又不在引擎中）这里 Yt 内部导入了 Ytester RTE Fasterloading.vi 用于预先封装支持，当然这是一种伪二次开发。后续的不支持项需要联系作者在 Ytester 中预先封装；

因此 Ytester 出于兼容性、调试中串口占用等小问题、决定将 Ytester 在开发模式下运行，这是最好的兼容方式+最好的开发便利性；然而这种模式下也是一堆问题。

Ytester 应用发布

最后，Ytester 采用以打包库（Ytester.lvlibp+YtCommon.lvlibp）的方式与用户开发的应用程序相结合发布使用来解决这个依赖问题。对于 STEP 中预先要使用的 VI 可以在 YtesterCreator.lvproj 中 RTE-Suppost.vi 中直接添加用于支持。以下是 Ytester 应用发布提供的模板项目：



在此项目中，用户可以修改应用程序图标、扩展更多功能，Ytester 将作为内核被调用使用。

六、用例

暂无

七、其他

暂无