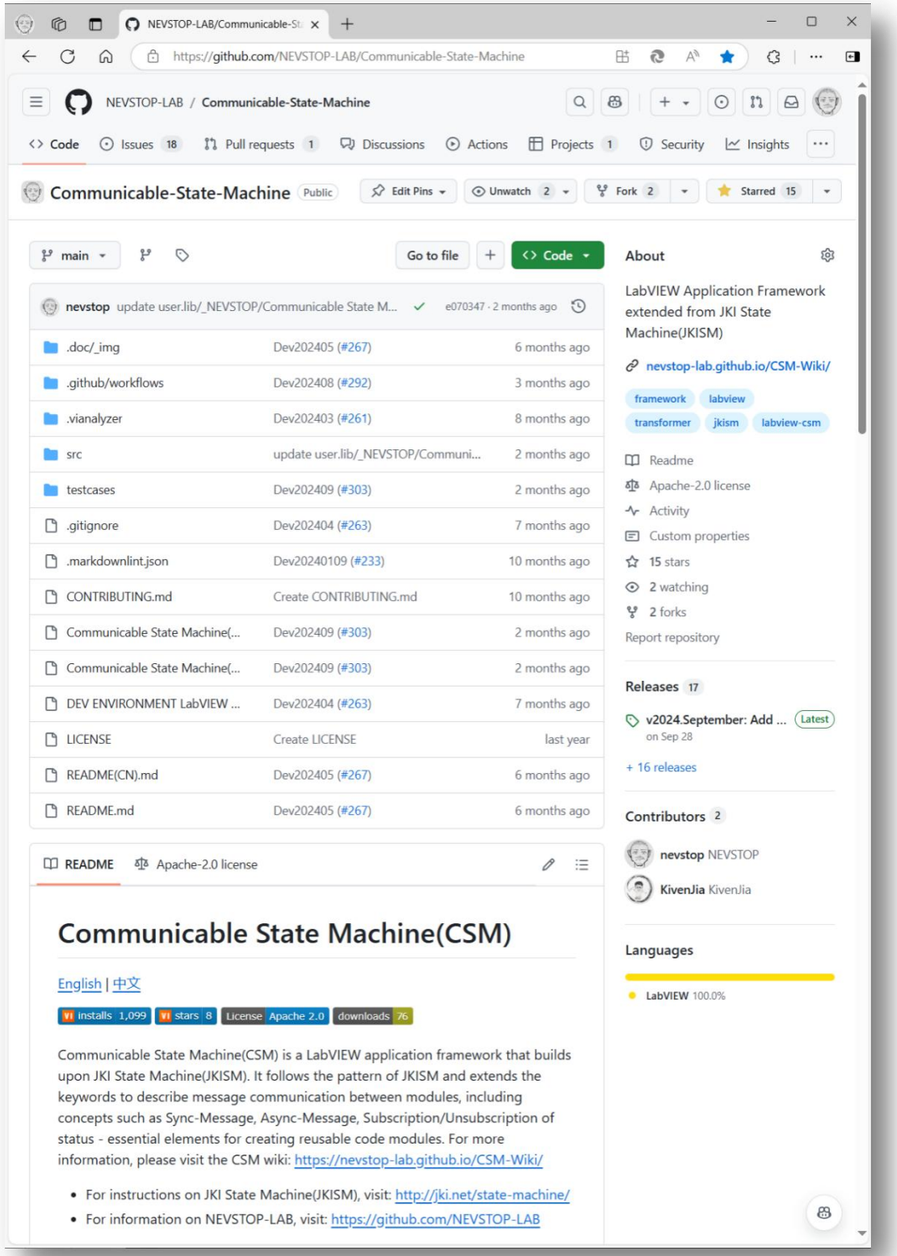
一种全新的开源 LabVIEW 编程架构: 可通讯状态机架构

李遥

Principal AE - NI, Emerson T&M

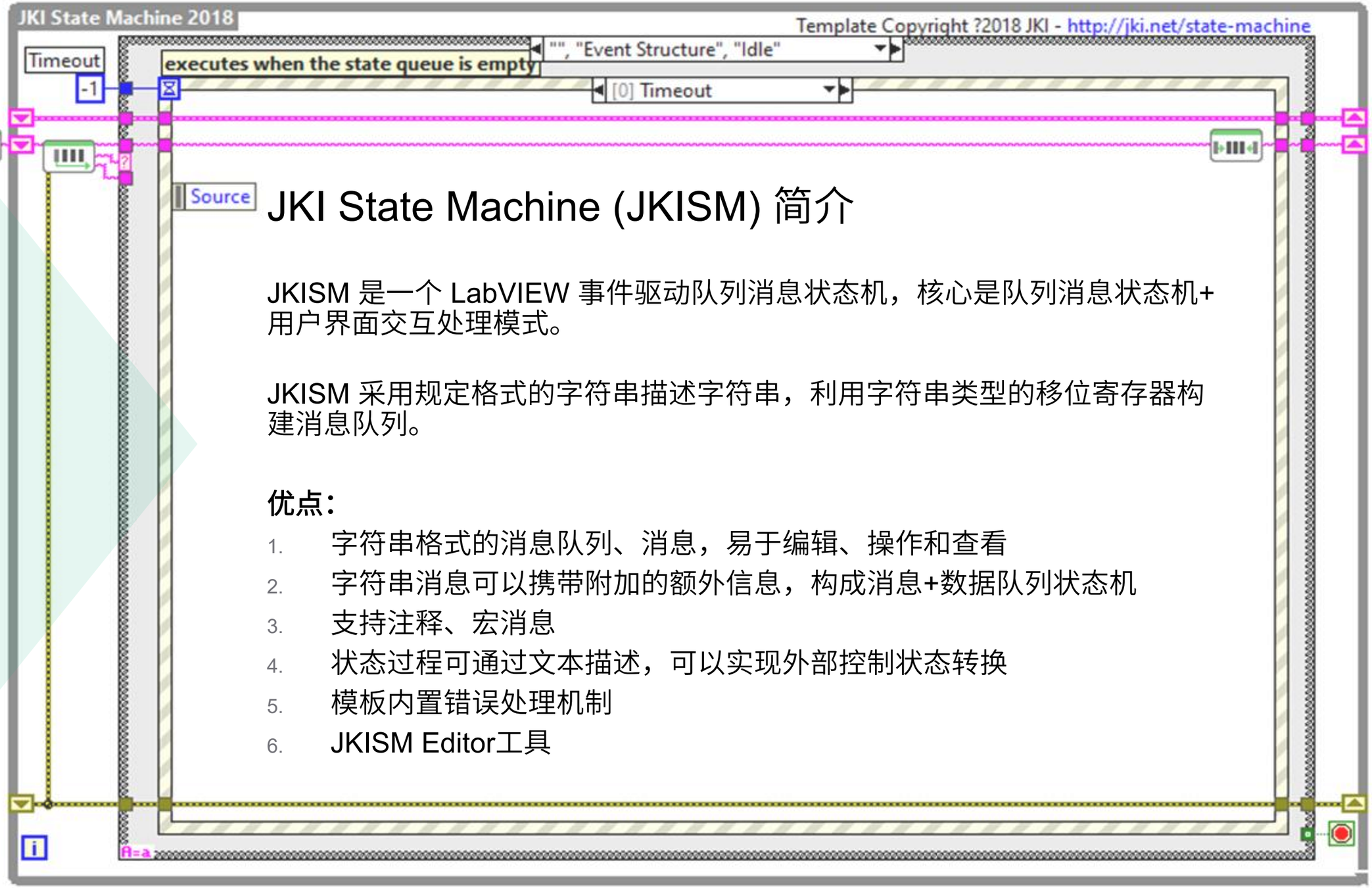


可通讯状态机框架 (Communicable State Machine)

Communicable State Machine(CSM) 是一个基于 JKI State Machine (JKISM) 扩展的开源 LabVIEW 应用开发框架。它遵循 JKISM 的字符串状态描述模式, 扩展了关键词, 用于描述模块之间的消息通信, 包括同步消息、异步消息、状态订阅/取消订阅等概念.

1. 完善的 LabVIEW 程序框架 (vs DQMH/SMO/AF …)
2. 全开源项目 (GitHub | gitee | vipm.io), MIT License
3. 中文技术支持
4. 持续的更新迭代

EMERSON



JKI State Machine (JKISM) 简介

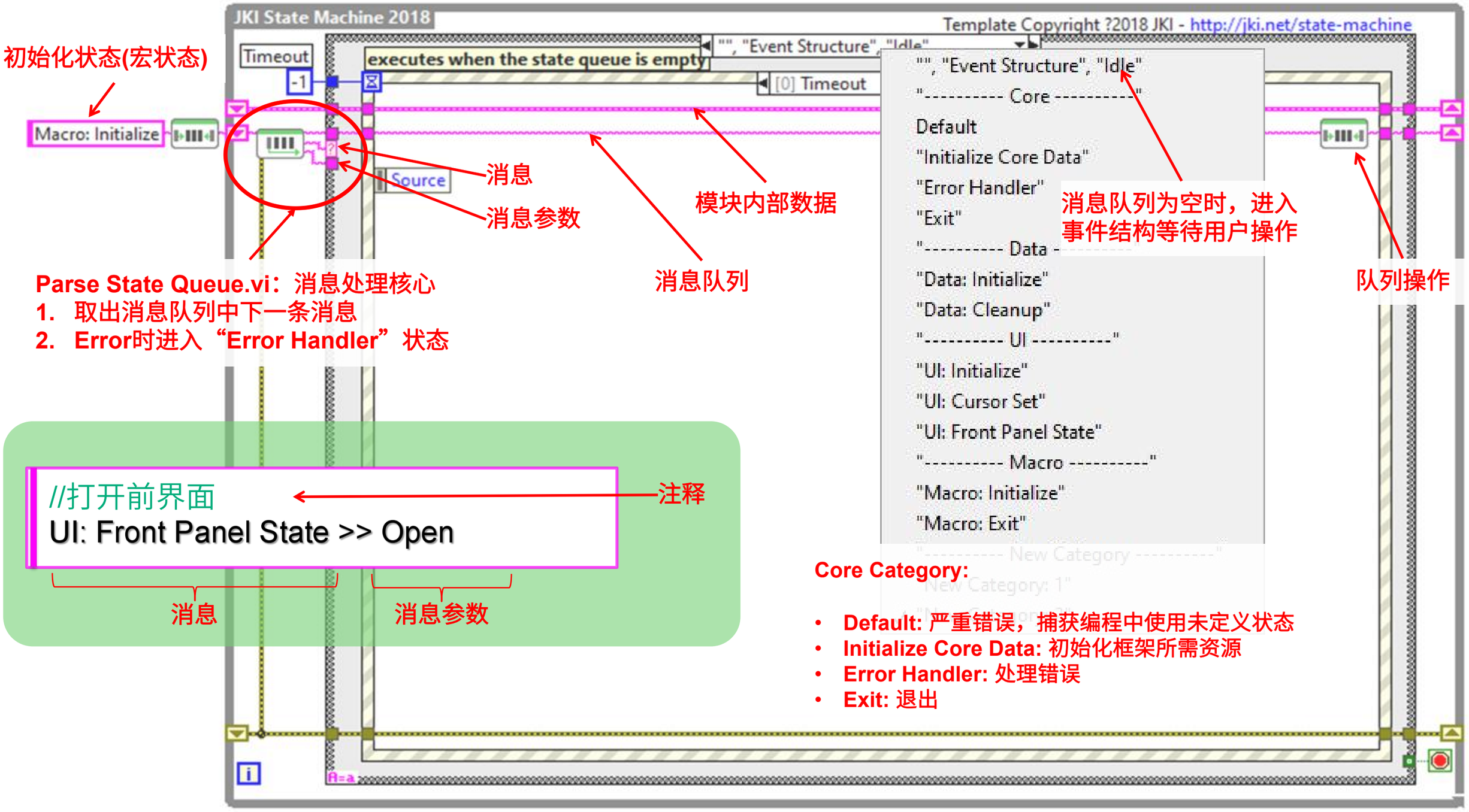
JKISM 是一个 LabVIEW 事件驱动队列消息状态机，核心是队列消息状态机+用户界面交互处理模式。

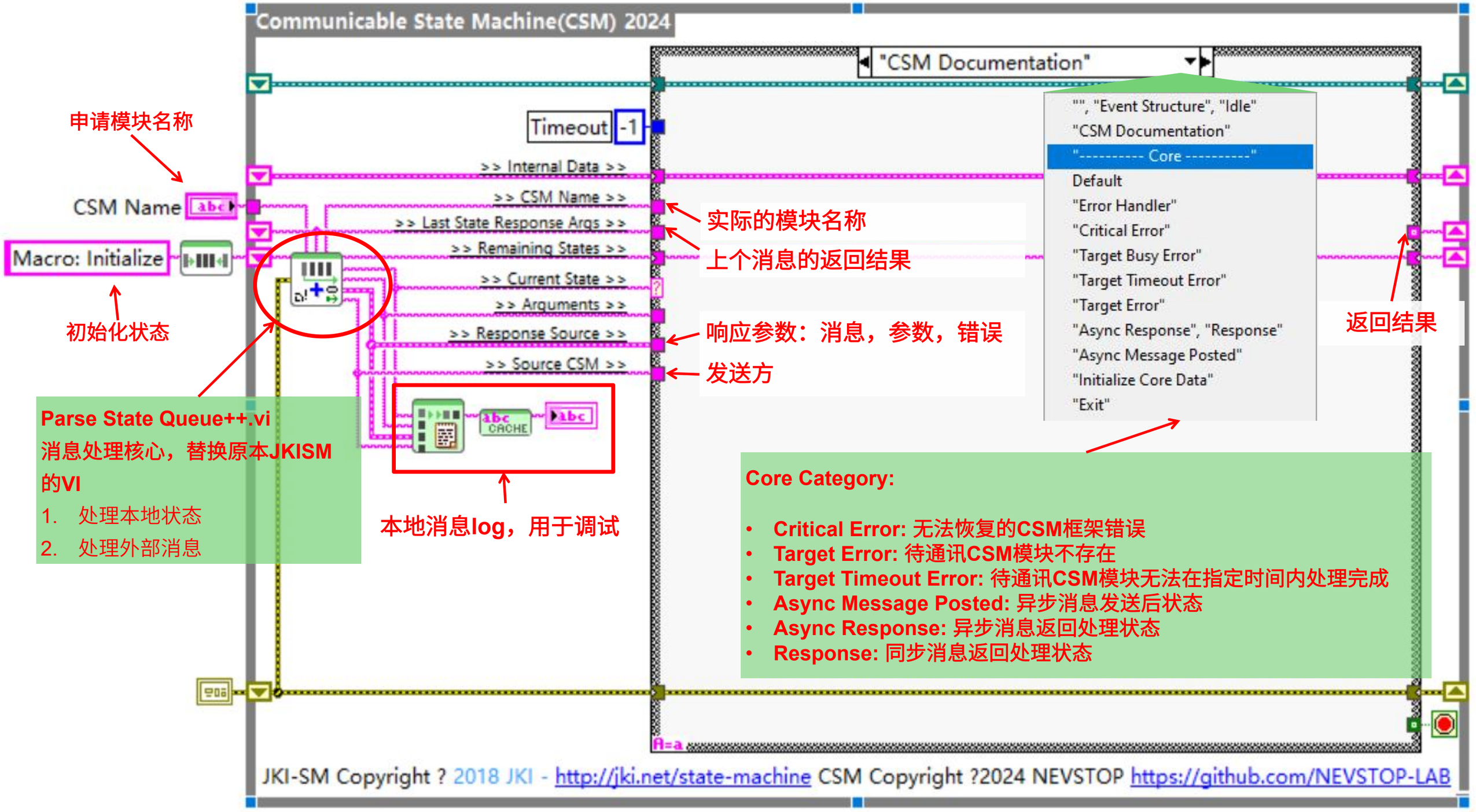
JKISM 采用规定格式的字符串描述字符串，利用字符串类型的移位寄存器构建消息队列。

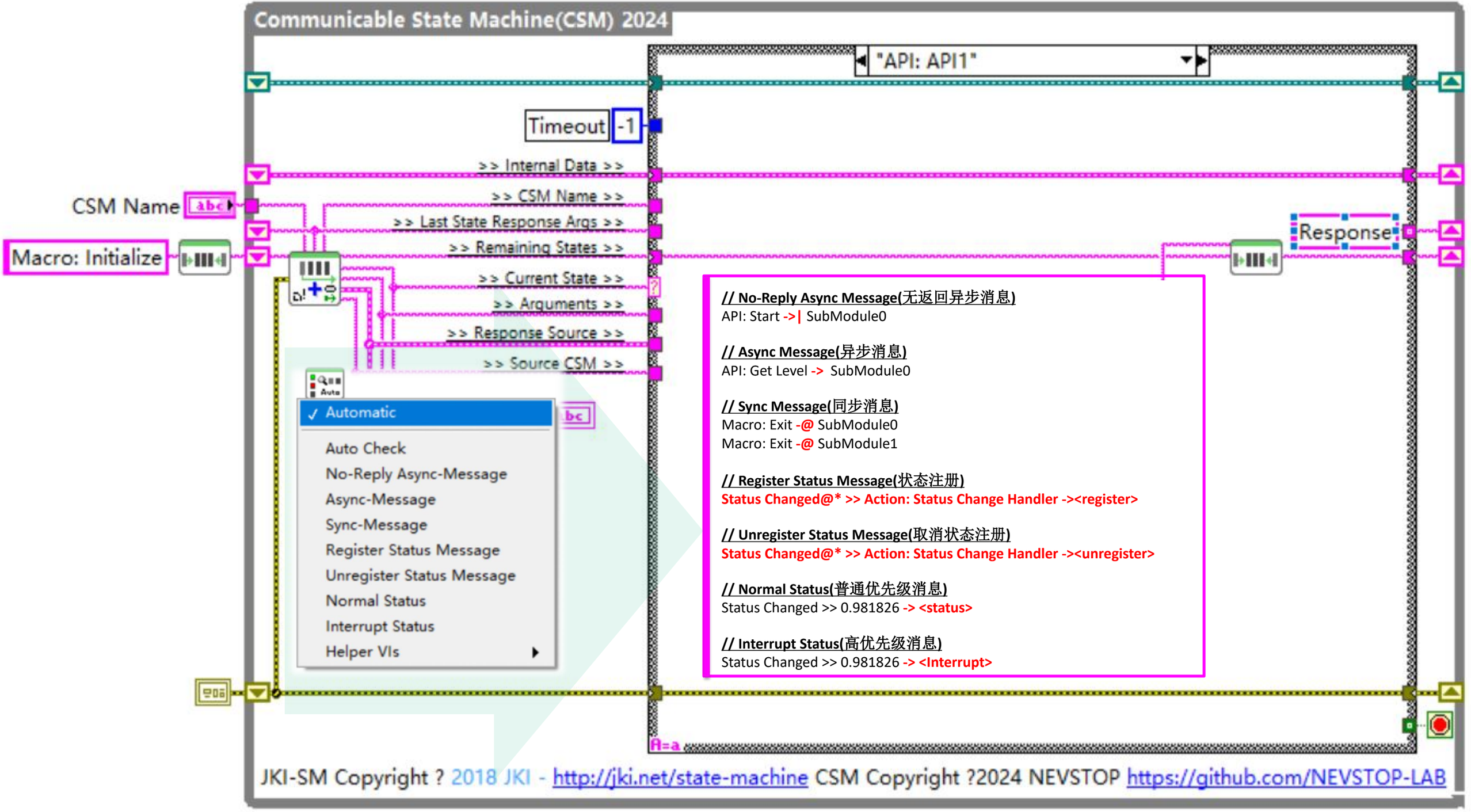
优点：

1. 字符串格式的消息队列、消息，易于编辑、操作和查看
2. 字符串消息可以携带附加的额外信息，构成消息+数据队列状态机
3. 支持注释、宏消息
4. 状态过程可通过文本描述，可以实现外部控制状态转换
5. 模板内置错误处理机制
6. JKISM Editor工具

JKISoftware/JKI-State-Machine: JKI State Machine (github.com)

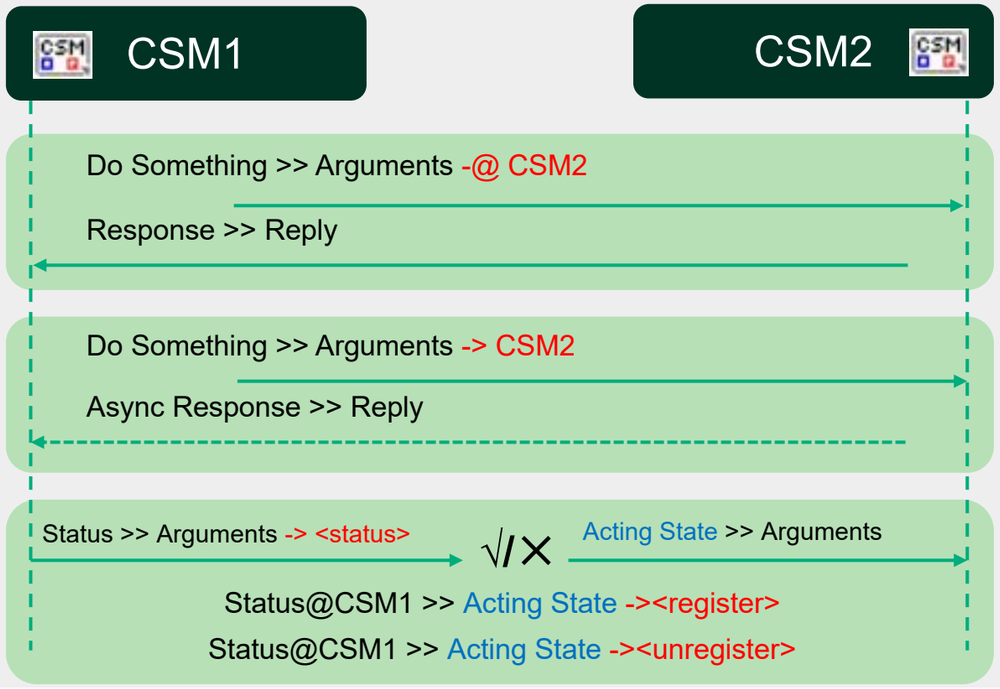

初始化状态(宏状态)
JKI State Machine 2018
Template Copyright ?2018 JKI - http://jki.net/state-machine
Timeout
-1
executes when the state queue is empty
[0] Timeout
Macro: Initialize
Source
消息
消息参数
模块内部数据
消息队列
消息队列为空时，进入
事件结构等待用户操作
队列操作
Parse State Queue.vi：消息处理核心
1. 取出消息队列中下一条消息
2. Error时进入“Error Handler”状态
"", "Event Structure", "Idle"
"---------- Core ----------"
Default
"Initialize Core Data"
"Error Handler"
"Exit"
"---------- Data ----------"
"Data: Initialize"
"Data: Cleanup"
"---------- UI ----------"
"UI: Initialize"
"UI: Cursor Set"
"UI: Front Panel State"
"---------- Macro ----------"
"Macro: Initialize"
"Macro: Exit"
"---------- New Category ----------"
"New Category: 1"
//打开前界面
UI: Front Panel State >> Open
注释
消息
消息参数
Core Category:
Default: 严重错误，捕获编程中使用未定义状态
Initialize Core Data: 初始化框架所需资源
Error Handler: 处理错误
Exit: 退出

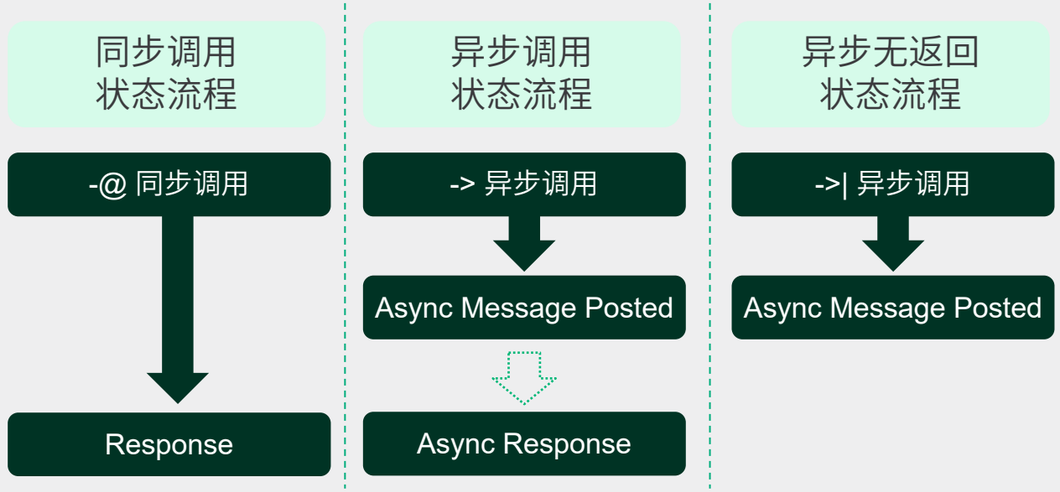

Communicable State Machine(CSM) 2024
"CSM Documentation"
申请模块名称
CSM Name
Macro: Initialize
初始化状态
Timeout -1
>> Internal Data >>
>> CSM Name >>
>> Last State Response Args >>
>> Remaining States >>
>> Current State >>
>> Arguments >>
>> Response Source >>
>> Source CSM >>
实际的模块名称
上个消息的返回结果
响应参数：消息，参数，错误
发送方
返回结果
"", "Event Structure", "Idle"
"CSM Documentation"
"---------- Core ----------"
Default
"Error Handler"
"Critical Error"
"Target Busy Error"
"Target Timeout Error"
"Target Error"
"Async Response", "Response"
"Async Message Posted"
"Initialize Core Data"
"Exit"
CACHE
Parse State Queue++.vi
消息处理核心，替换原本JKISM的VI
1. 处理本地状态
2. 处理外部消息
本地消息log，用于调试
Core Category:
• Critical Error: 无法恢复的CSM框架错误
• Target Error: 待通讯CSM模块不存在
• Target Timeout Error: 待通讯CSM模块无法在指定时间内处理完成
• Async Message Posted: 异步消息发送后状态
• Async Response: 异步消息返回处理状态
• Response: 同步消息返回处理状态
JKI-SM Copyright ? 2018 JKI - http://jki.net/state-machine CSM Copyright ?2024 NEVSTOP https://github.com/NEVSTOP-LAB

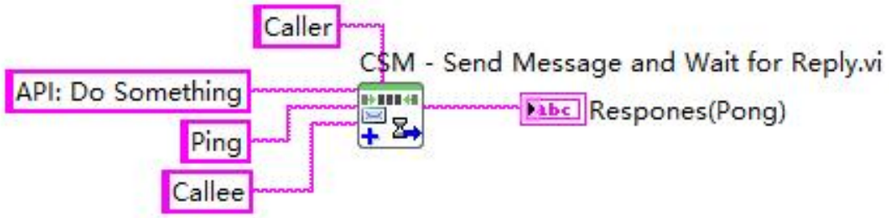

Communicable State Machine(CSM) 2024
"API: API1"
Timeout
-1
>> Internal Data >>
>> CSM Name >>
>> Last State Response Args >>
>> Remaining States >>
>> Current State >>
>> Arguments >>
>> Response Source >>
>> Source CSM >>
CSM Name
Macro: Initialize
Response
Automatic
Auto Check
No-Reply Async-Message
Async-Message
Sync-Message
Register Status Message
Unregister Status Message
Normal Status
Interrupt Status
Helper VIs
// No-Reply Async Message(无返回异步消息)
API: Start ->| SubModule0
// Async Message(异步消息)
API: Get Level -> SubModule0
// Sync Message(同步消息)
Macro: Exit -@ SubModule0
Macro: Exit -@ SubModule1
// Register Status Message(状态注册)
Status Changed@* >> Action: Status Change Handler -><register>
// Unregister Status Message(取消状态注册)
Status Changed@* >> Action: Status Change Handler -><unregister>
// Normal Status(普通优先级消息)
Status Changed >> 0.981826 -> <status>
// Interrupt Status(高优先级消息)
Status Changed >> 0.981826 -> <Interrupt>
JKI-SM Copyright ? 2018 JKI - http://jki.net/state-machine CSM Copyright ?2024 NEVSTOP https://github.com/NEVSTOP-LAB

Communication Between CSMs

CSM 模块间通过消息进行通讯, 内部依然满足状态机的逻辑

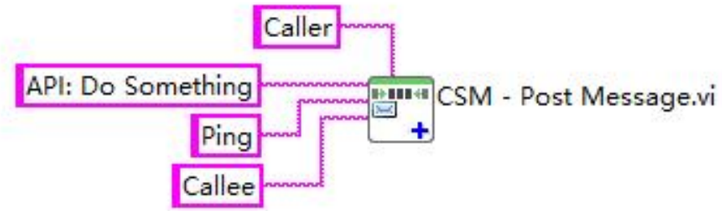






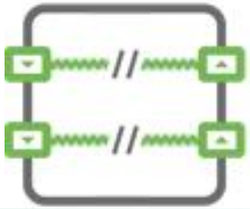
=

//Sync Call. Caller wait until getting “Pong”.
API: Do Something >> Ping -@ Callee



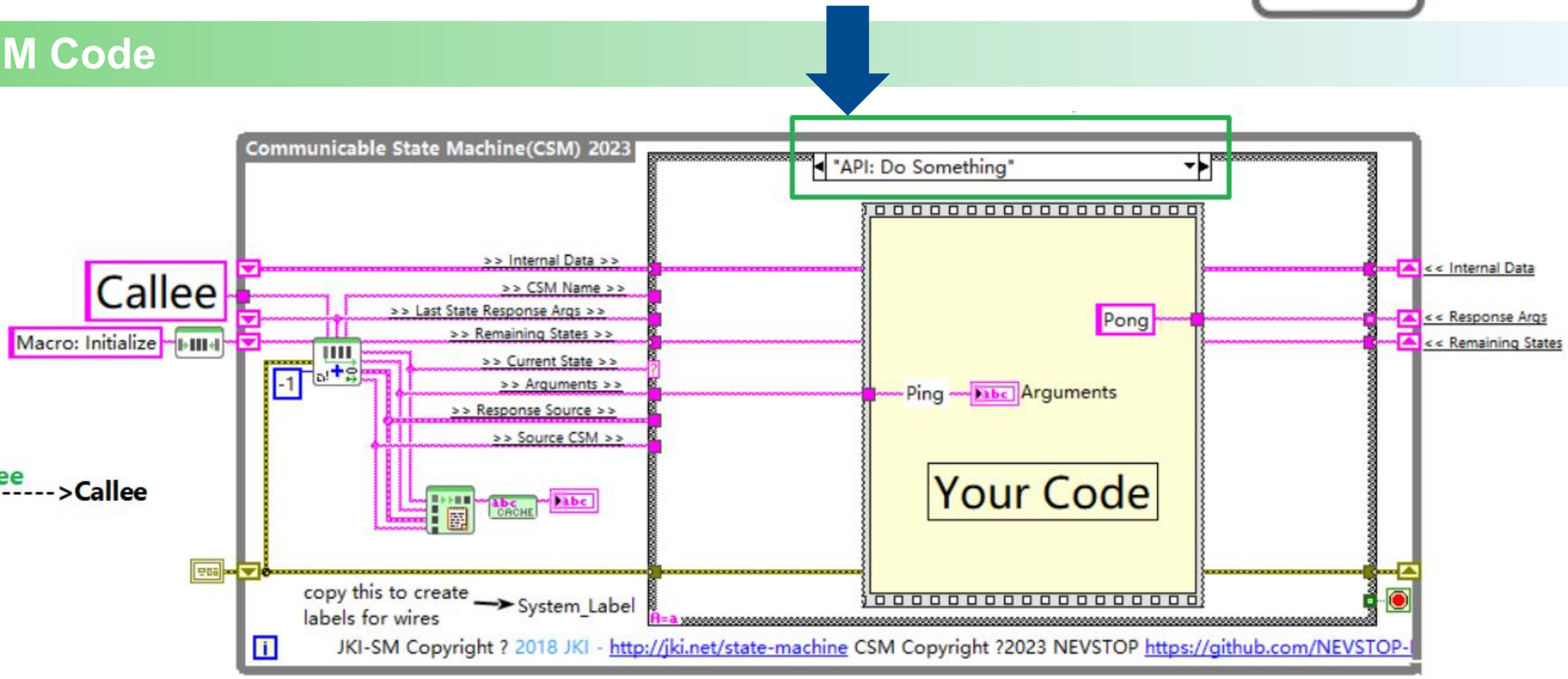
=

//Async Call. Caller will not wait.
API: Do Something >> Ping ->| Callee



Communication From Non-CSM Code

(Async Call)
API: Do Someting >> Ping -> Callee
Caller ----------> Callee
<----------
Async Response >> Pong
(Automatically Generated)



常见的 LabVIEW 编程框架比较

	DQMH	SMO	Actor Framework	CSM
优点	• 继承于最广泛应用的框架 QMH • 无OOP，对于大多数 LabVIEW用户比较友好 • 社区活跃度高，资料/工具众多	• 非常符合面向对象的概念 • 模块独立性好，操作调用直观 • SMO 最适用的场景还是模块复用，不与调用方一起开发	• 操作者、消息均解耦，解耦程度高，便于多人协作开发 • 熟悉 OOP 的开发人员使用方便	• 继承于JKISM，代码集中度高 • 无OOP,对于大多数LabVIEW用户比较友好 • 通过文档定义接口分工，无需代码 • 容易实现操作序列化 (Serialization) • 内置多种高级模式
缺点	• 代码接口冗余 • 两套模板实现 Singleton/Standalone模块	• 框架复杂度高，对用用户要求高 • 目前社区用户少，更新频率不高 • 资料比较少	• 框架复杂度高，对用用户要求高 • 项目需要架构师角色	• 所有消息必须使用 STRING 格式传递，有转换的开销 • 参数不通过 LabVIEW 数据类型定义，不严格要求 • 目前工具/资料还不完善

	DQMH	SMO	Actor Framework	CSM
外部接口	Script 辅助创建 API	类公共接口作为API	类封装的 Message 消息通讯	字符串格式的消息通讯，非必须创建API
状态反馈	User Event，外部需要存在事件结构添加订阅逻辑	User Event，外部需要存在事件结构添加订阅逻辑	消息处理中封装反馈消息	• 消息通过 JKISM 机制传递，无需 UserEvtStrucutre • 隐式传递，订阅无需修改侵入代码，可以外部调用。
可复制模块	复杂，两套模板	容易，类实现，框架不区分	容易，类实现，消息和模块独立，灵活性高	容易，VI属性决定
代码依赖	调用方需要依赖模块的 自定义事件和参数定义，需要预先开发框架定义部分的接口代码	调用方高度依赖模块实现，所以必须先开发框架接口部分	调用方可能有继承依赖和消息依赖，需预先定义消息类和方法	调用方开发无需依赖框架代码。传递消息、参数都使用字符串，不显式依赖，不容易 broken 代码。复杂参数需要依赖
参数传递	任意类型，使用 User Event 传递	任意类型，使用 User Event 传递	任意类型，需要封装 Message 类	字符串，复杂类型需转换
代码复杂度	框架逻辑简单，模块代码冗余度高，不依赖 script 增删比较复杂	框架特别复杂，模块逻辑基于JKI状态机，但开发需要理解JKI SMO 的概念逻辑	框架特别复杂，实现高度依赖OOP	继承 JKISM 的优点，代码逻辑集中，理解消息通讯状态跳转后，与JKISM 编程思路相同
生态	最佳	社区中仅有JKI提供模板和工具	一般，通常作为企业内部统一架构	待完善
OOP	无	有	有	无
执行效率	依赖 Event Structure	依赖 Event Structure/ LabVIEW OOP	LabVIEW OOP	依赖参数的解析
UI编写	支持	支持	支持	支持
RTOS	支持	不推荐	不推荐	支持
高级模式	自定义模板	可组合，模块能作为子模块	继承 Actor Core.vi 实现特殊的模式	工作者模式/责任链模式/自定义模板

CSM 独特之处

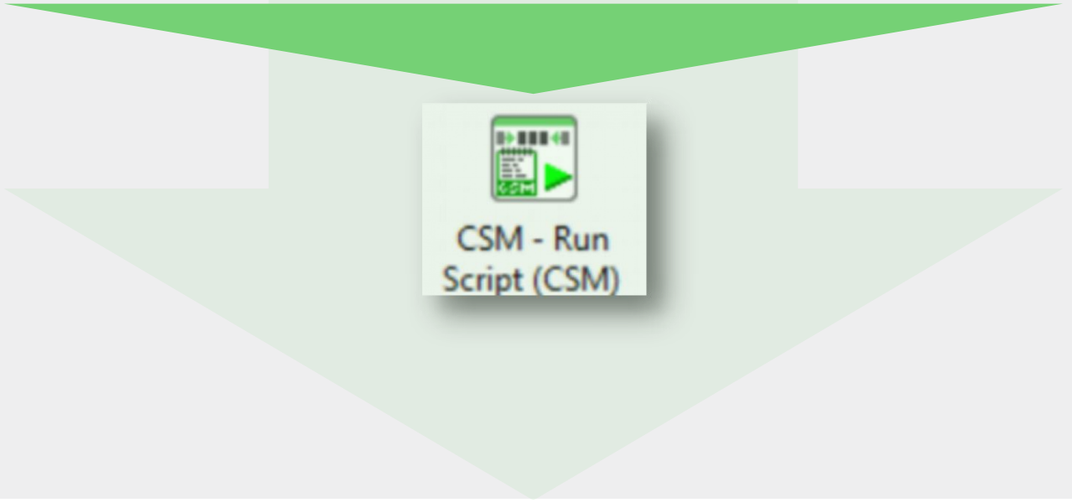
1. 纯文本的流程控制
2. 隐形的“通讯总线”
3. VI 即模块
4. 内置超级详细的检查接口
5. 拓展式的设计

纯文本的流程控制

- 便于架构师脱离代码进行架构设计
- 便于脚本化控制、测试
- 编译后动态修改程序的行为（动作行为、订阅状态等）
- 容易实现机器间的协议通讯

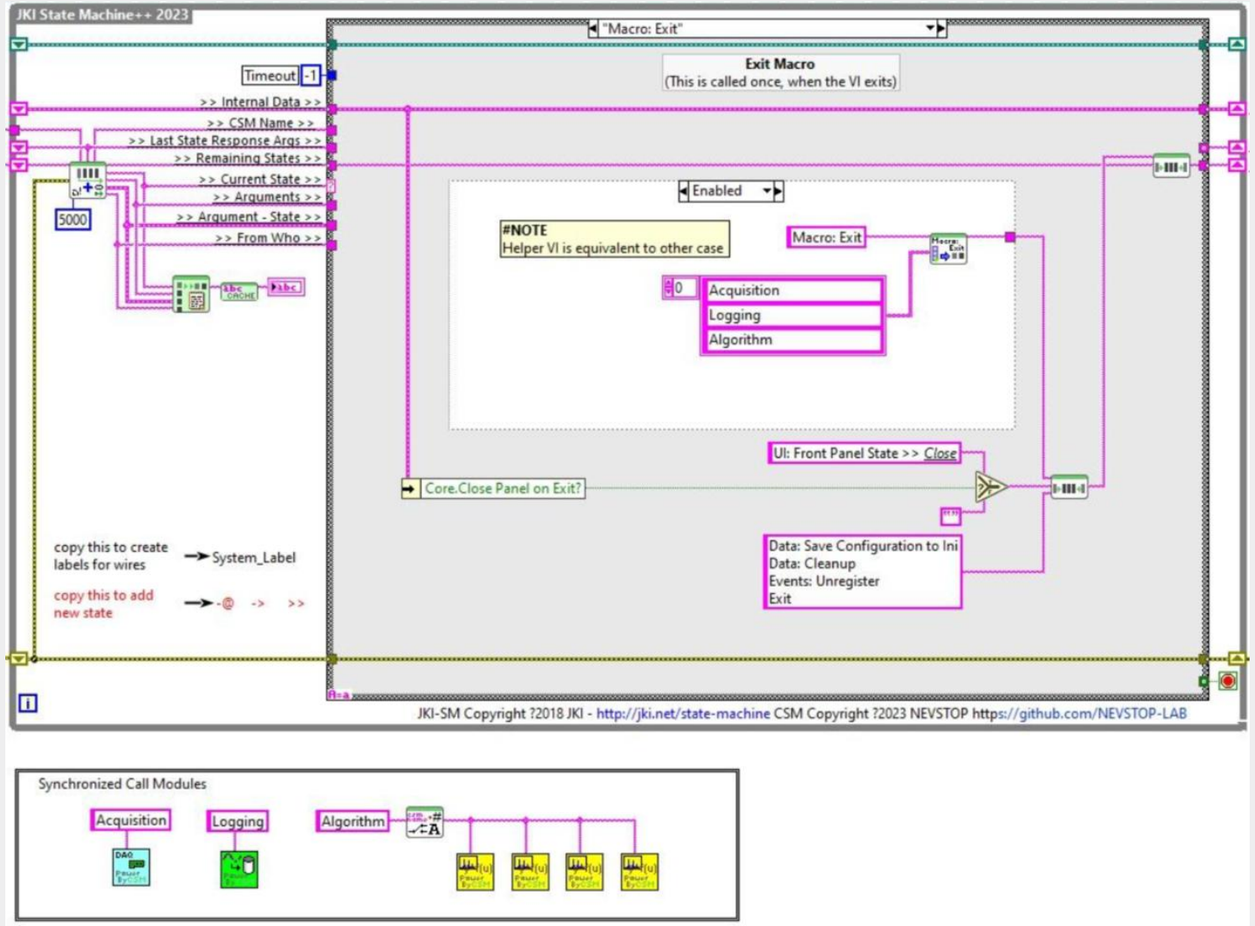
Messages as Plain Text

```
DAQ: Initialize   >> PXIe6363                    -@ DAQDev1
DAQ: Configure >> Clock:SampleClk;Rate:1M        -@ DAQDev1
DAQ: Start                                       -@ DAQDev1
DAQ: Read         >> 1000                        -@ DAQDev1
DAQ: Stop                                        -@ DAQDev1
DAQ: Close                                       -@ DAQDev1
```



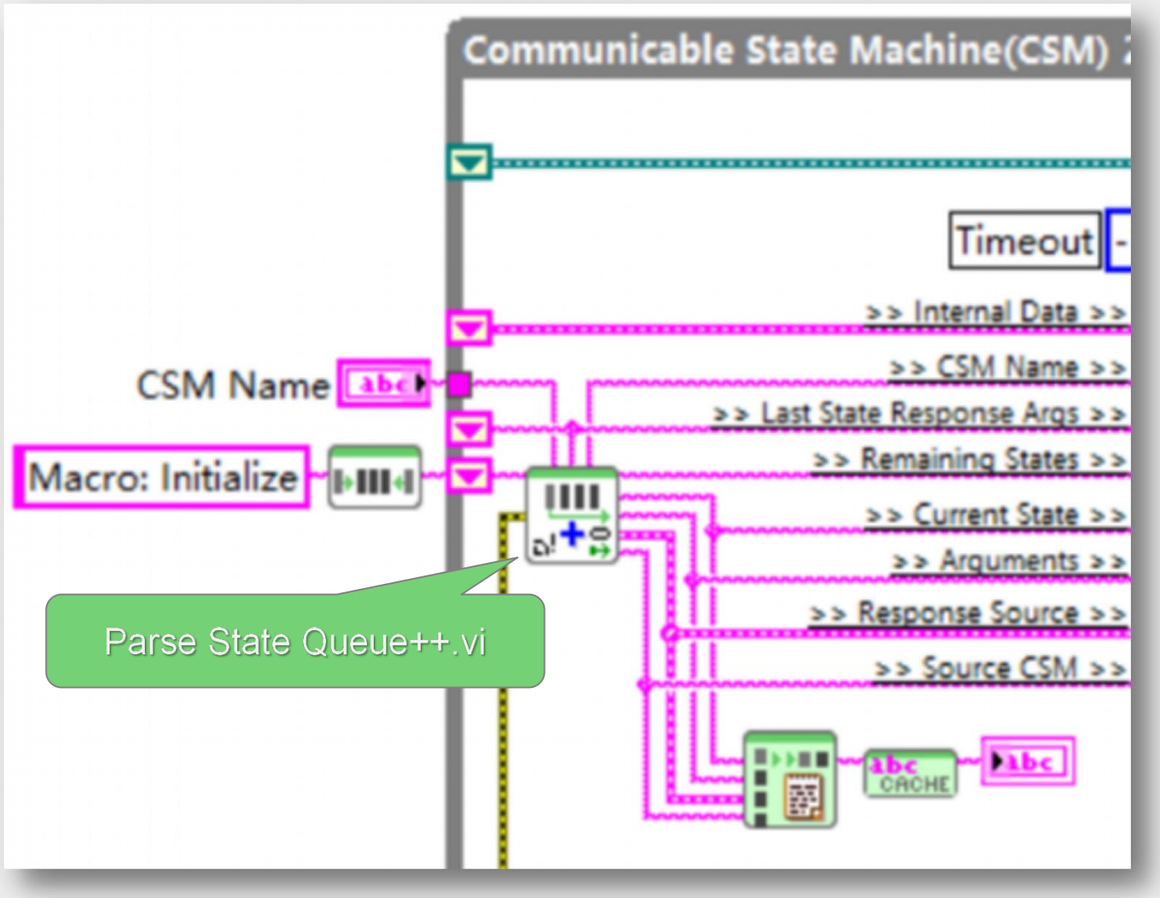
隐形的“通讯总线”

- 容易实现 1:1, 1:N, M:N 的逻辑关系
- 无需显示调用 LabVIEW 队列、事件
- 模块不显示依赖，通过“挂载”的方式加入系统
- 参数/数据 通过“编码” ->传输-> “解码”的方式传递



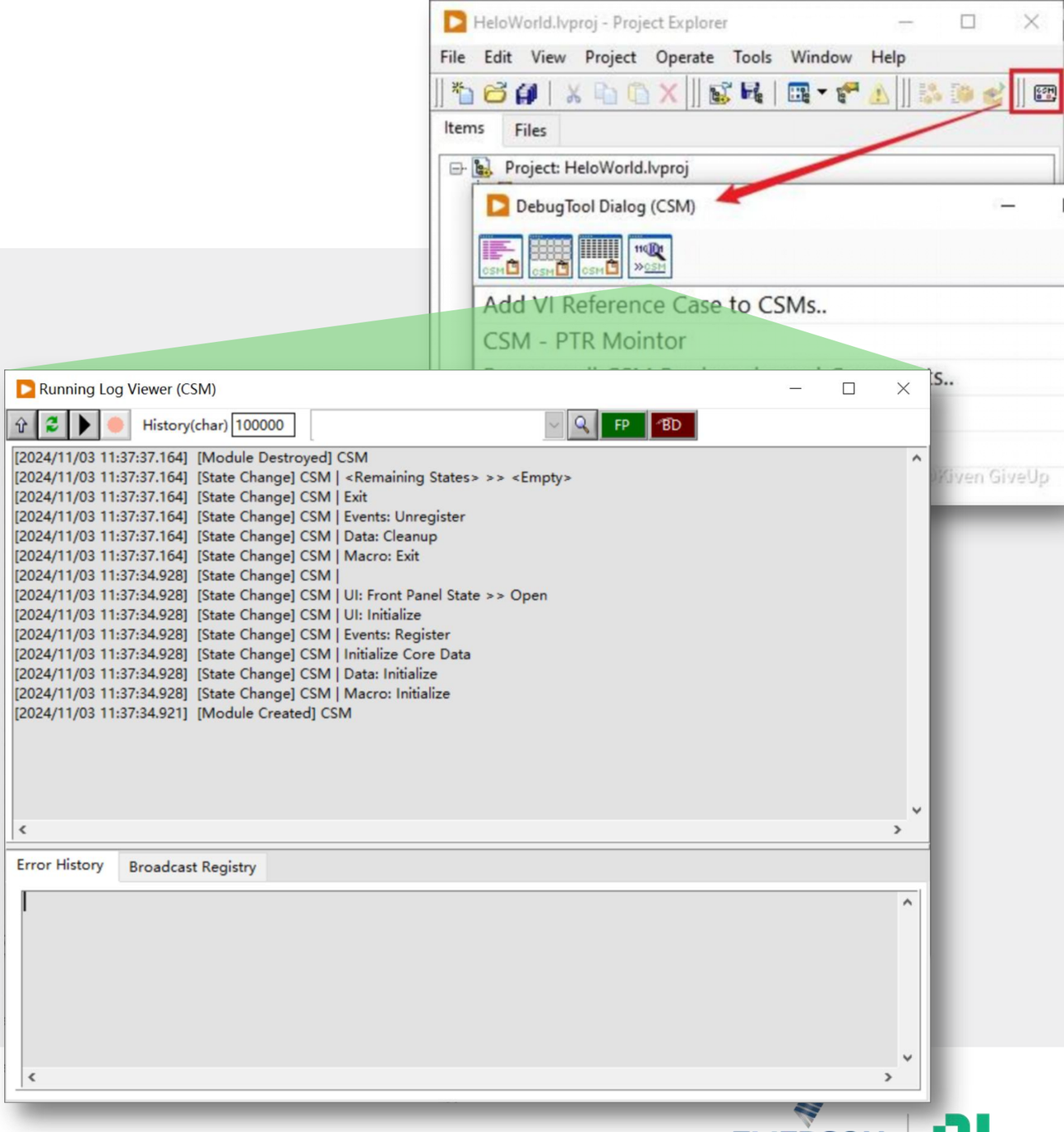
VI 即模块

- VI 就是模块，模块就是VI
  - VI 可重入属性决定 Singleton / Cloneable
  - 易于编译发布
- 代码集中度高
  - 框架代码集中在 *Parse State Queue++.vi* 中
  - 可见代码大部分都是用户代码



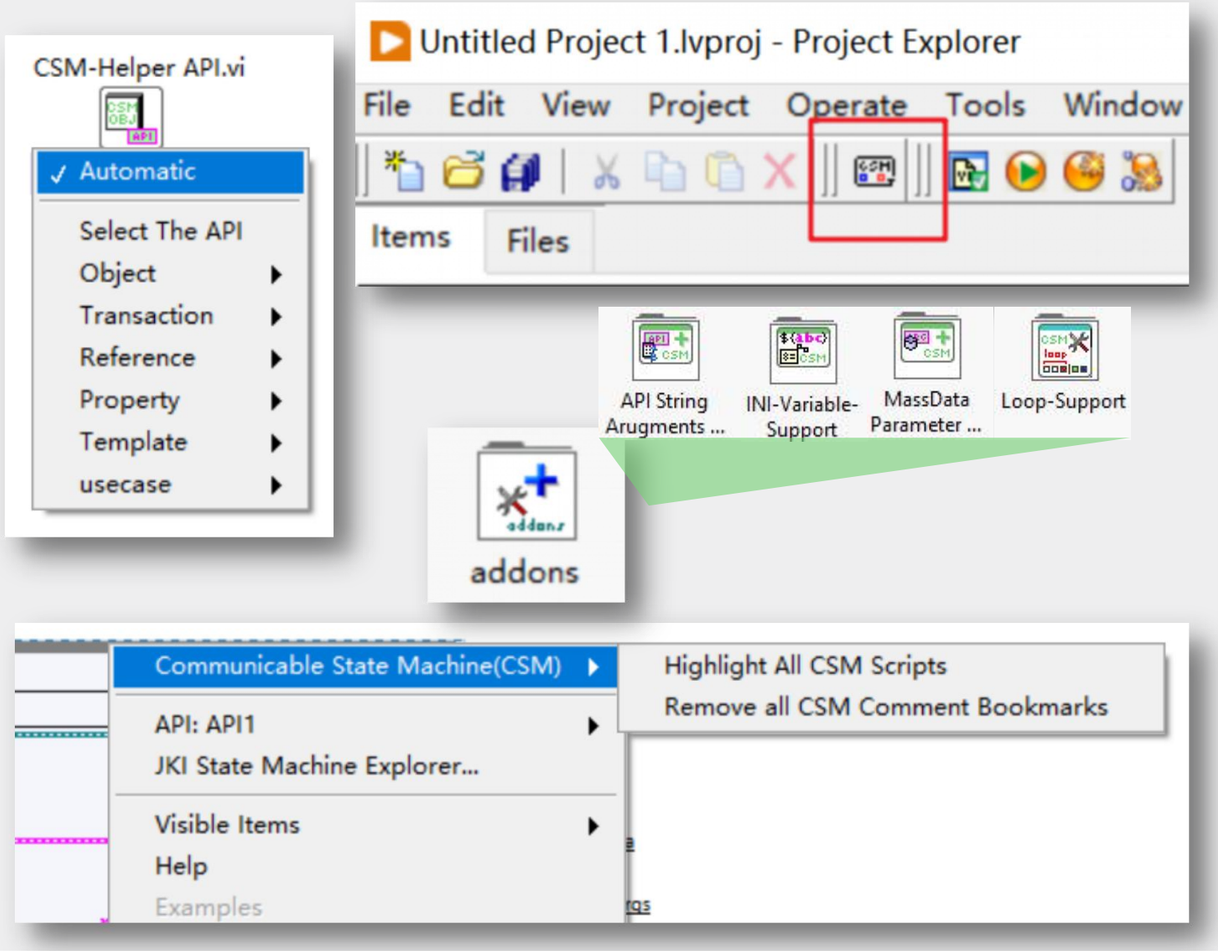
内置超级详细的调试接口

- 内置调试接口，记录详细的过程信息
  - 模块创建、销毁
  - 模块状态 (state) 轮转，包含参数
  - 模块间消息间通讯，包含参数、模块名称、返回值
  - 状态 (status) 发布
  - 状态订阅、取消订阅
  - 模块错误（Error）
  - 用户自定义消息
- 基于调试接口创建各种调试工具



拓展式的设计

- ***CSM Global Log API***, 可拓展调试工具
- ***Add-on/Template*** 函数选板可加载第三方工具包
- ***CSM-Helper API***, 可自定义 CSM 插件等
- 支持 VI Analyzer, 可拓展 CSM 模块静态分析
- 工具栏入口、右键菜单入口可以加载第三方工具
- 兼容 JKISM 工具
- 安装后自动切换 VI 帮助语言



欢迎下载试用，并提供反馈

通过 GitHub/Gitee Issue/PR/Discussions 参与开发，提供反馈。

1 GitHub

NEVSTOP-LAB/Communicable-State-Machine

2 gitee

Communicable-State-Machine: 可通信状态机（CSM）

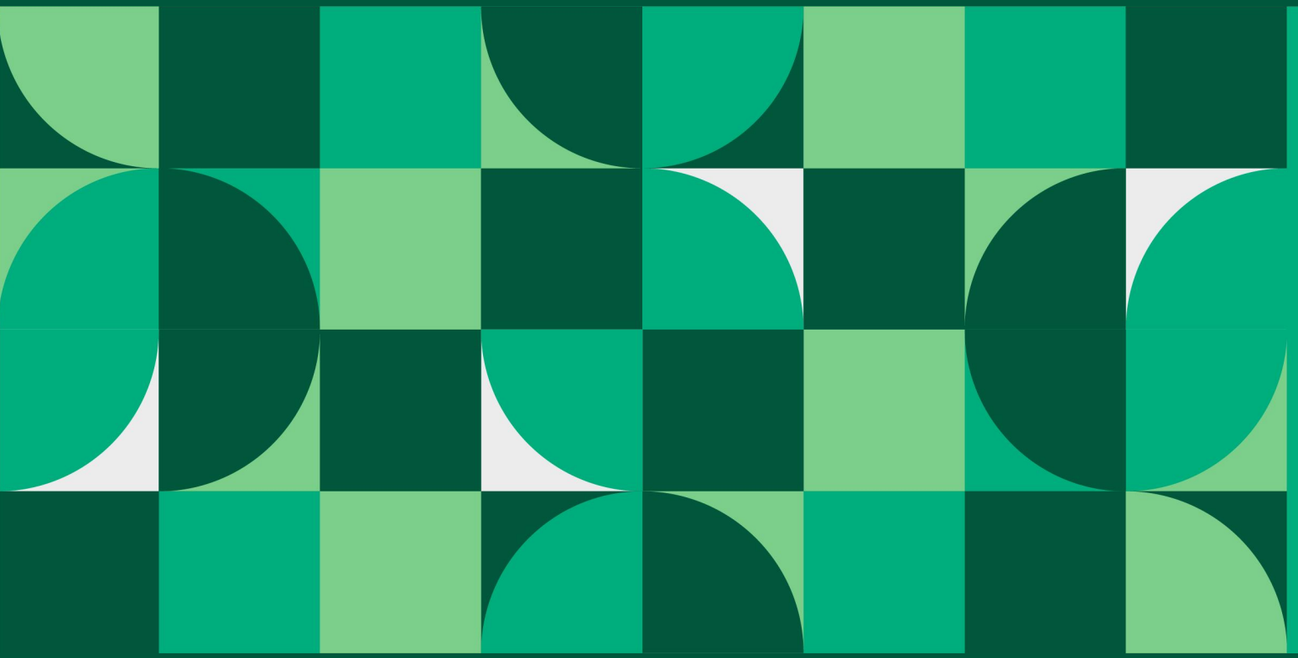
3 VIPM

Communicable State Machine Framework - Package List

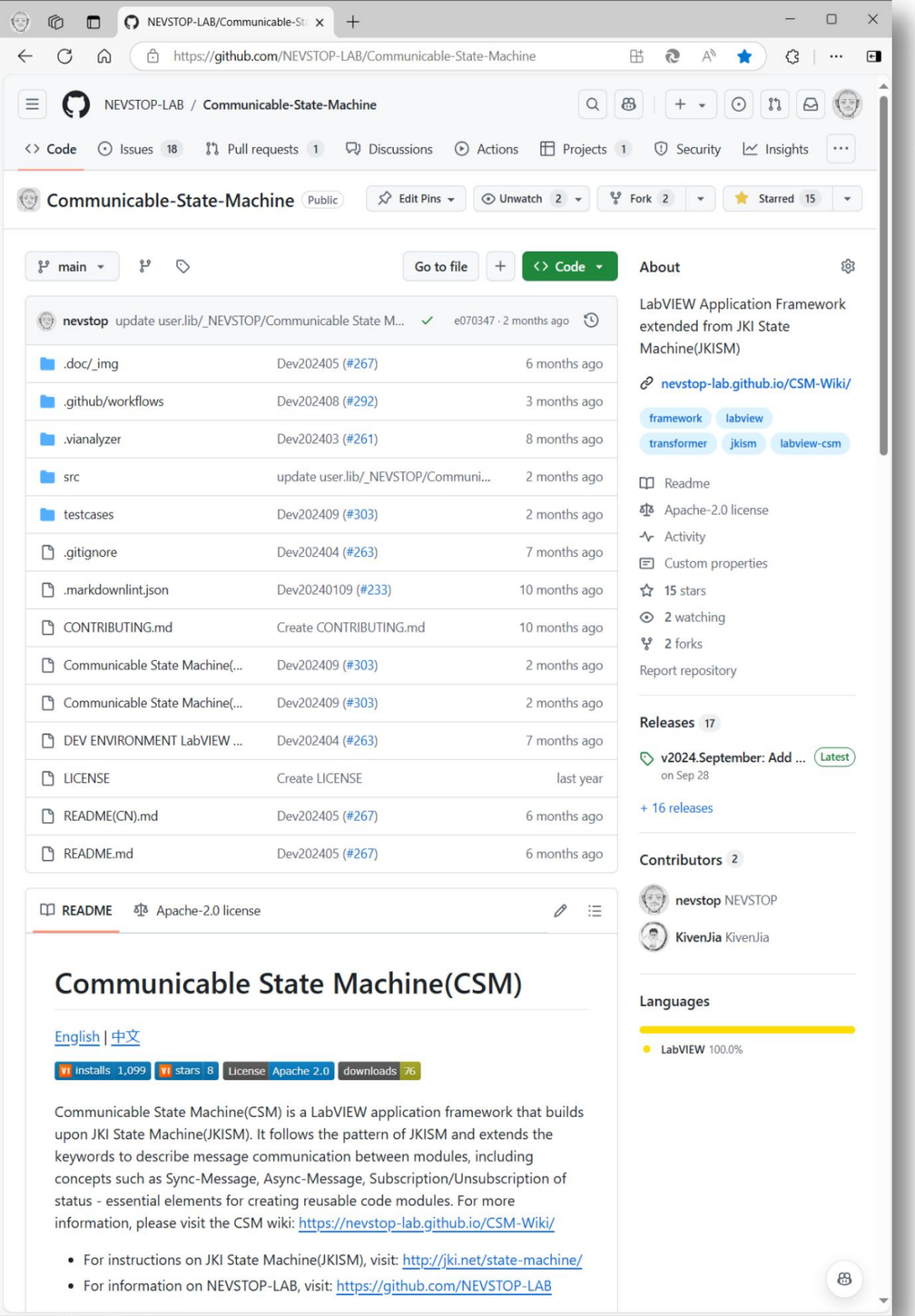
EMERSON

关注NI微信公众号
【恩艾在您身边】

会后获取技术演讲讲义
及更多干货内容



NI is now part of Emerson.



可通讯状态机框架 (Communicable State Machine)

Communicable State Machine(CSM) 是一个基于 JKI State Machine (JKISM) 扩展的开源 LabVIEW 应用开发框架。它遵循 JKISM 的字符串状态描述模式, 扩展了关键词, 用于描述模块之间的消息通信, 包括同步消息、异步消息、状态订阅/取消订阅等概念.

1. 完善的 LabVIEW 程序框架 (vs DQMH/SMO/AF …)
2. 全开源项目 (GitHub | gitee | vipm.io), MIT License
3. 中文技术支持
4. 持续的更新迭代

EMERSON

可通讯状态机框架(Communicable State Machine)

在 CSM 框架中调用模块

同步调用
状态流程

-@ 同步调用

异步调用
状态流程

-> 异步调用

Async Message Posted

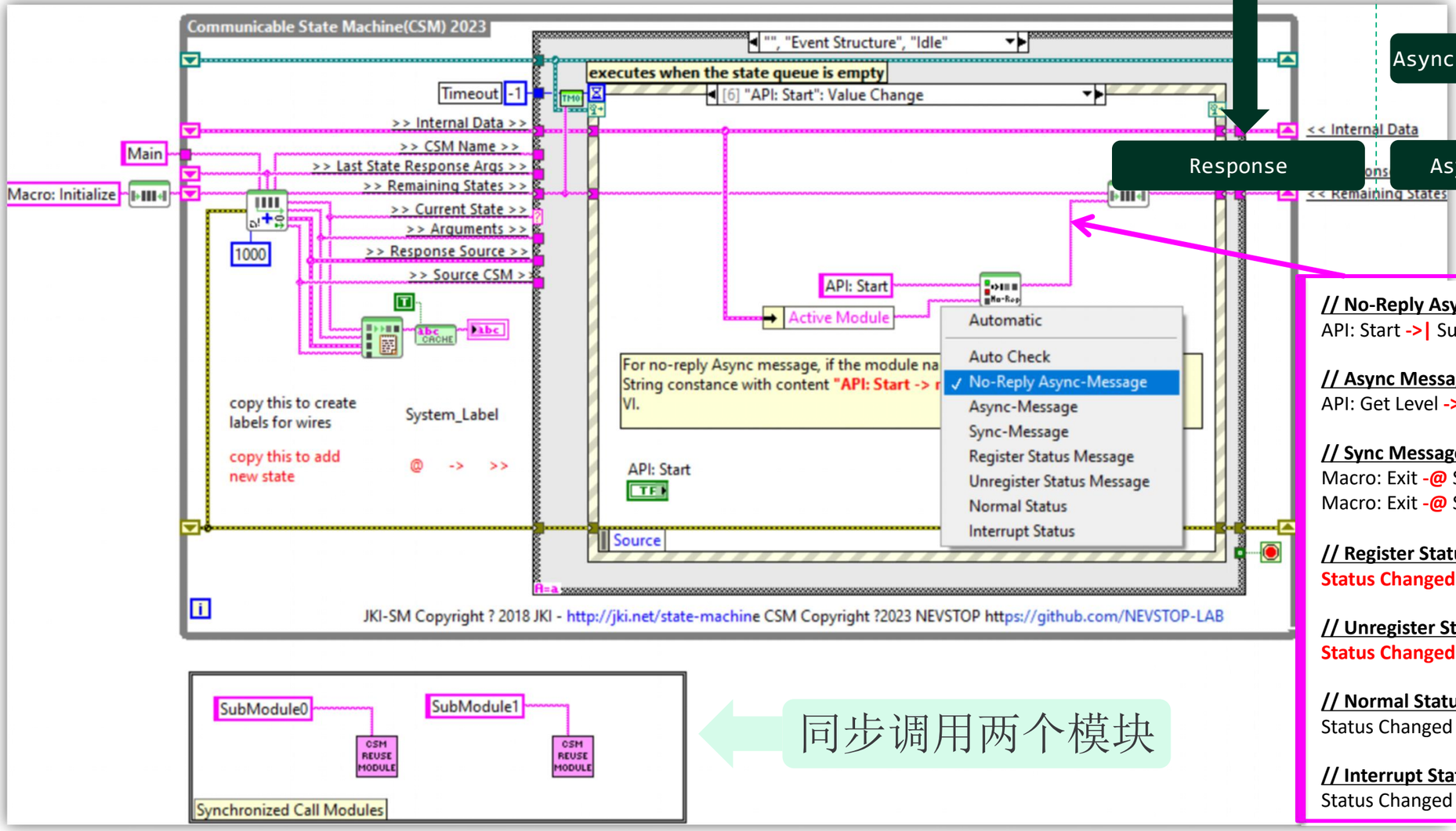
Async Response

异步无返回
状态流程

->| 异步调用

Async Message Posted

Response



同步调用两个模块

// No-Reply Async Message(无返回异步消息)
API: Start ->| SubModule0

// Async Message(异步消息)
API: Get Level -> SubModule0

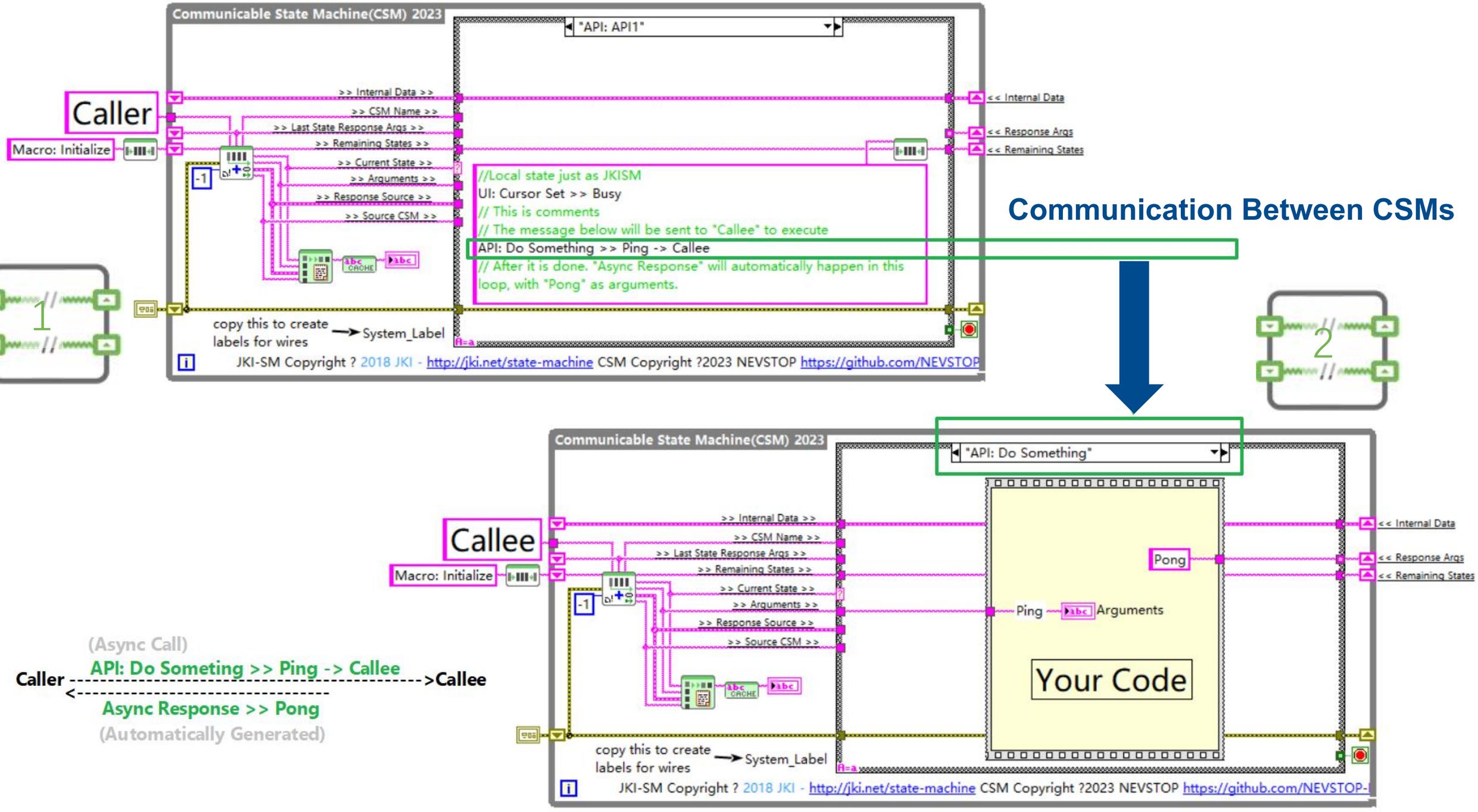
// Sync Message(同步消息)
Macro: Exit -@ SubModule0
Macro: Exit -@ SubModule1

// Register Status Message(状态注册)
Status Changed@* >> Action: Status Change Handler -><register>

// Unregister Status Message(取消状态注册)
Status Changed@* >> Action: Status Change Handler -><unregister>

// Normal Status(普通优先级消息)
Status Changed >> 0.981826 -> <status>

// Interrupt Status(高优先级消息)
Status Changed >> 0.981826 -> <Interrupt>

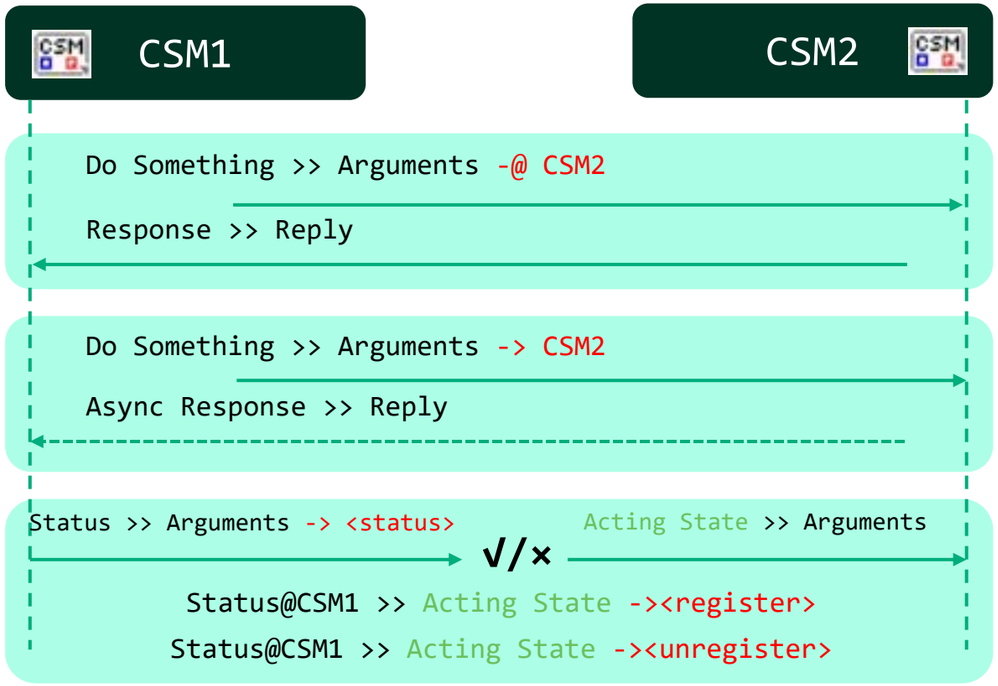

Communicable State Machine(CSM) 2023
"API: API1"
Caller
Macro: Initialize
>> Internal Data >>
>> CSM Name >>
>> Last State Response Args >>
>> Remaining States >>
>> Current State >>
>> Arguments >>
>> Response Source >>
>> Source CSM >>
<< Internal Data
<< Response Args
<< Remaining States
//Local state just as JKISM
UI: Cursor Set >> Busy
// This is comments
// The message below will be sent to "Callee" to execute
API: Do Something >> Ping -> Callee
// After it is done. "Async Response" will automatically happen in this loop, with "Pong" as arguments.
copy this to create labels for wires
System_Label
JKI-SM Copyright ? 2018 JKI - http://jki.net/state-machine CSM Copyright ?2023 NEVSTOP https://github.com/NEVSTOP
1
2
Communication Between CSMs
Communicable State Machine(CSM) 2023
"API: Do Something"
Callee
Macro: Initialize
Pong
Ping
Arguments
Your Code
(Async Call)
API: Do Someting >> Ping -> Callee
Caller
Callee
Async Response >> Pong
(Automatically Generated)

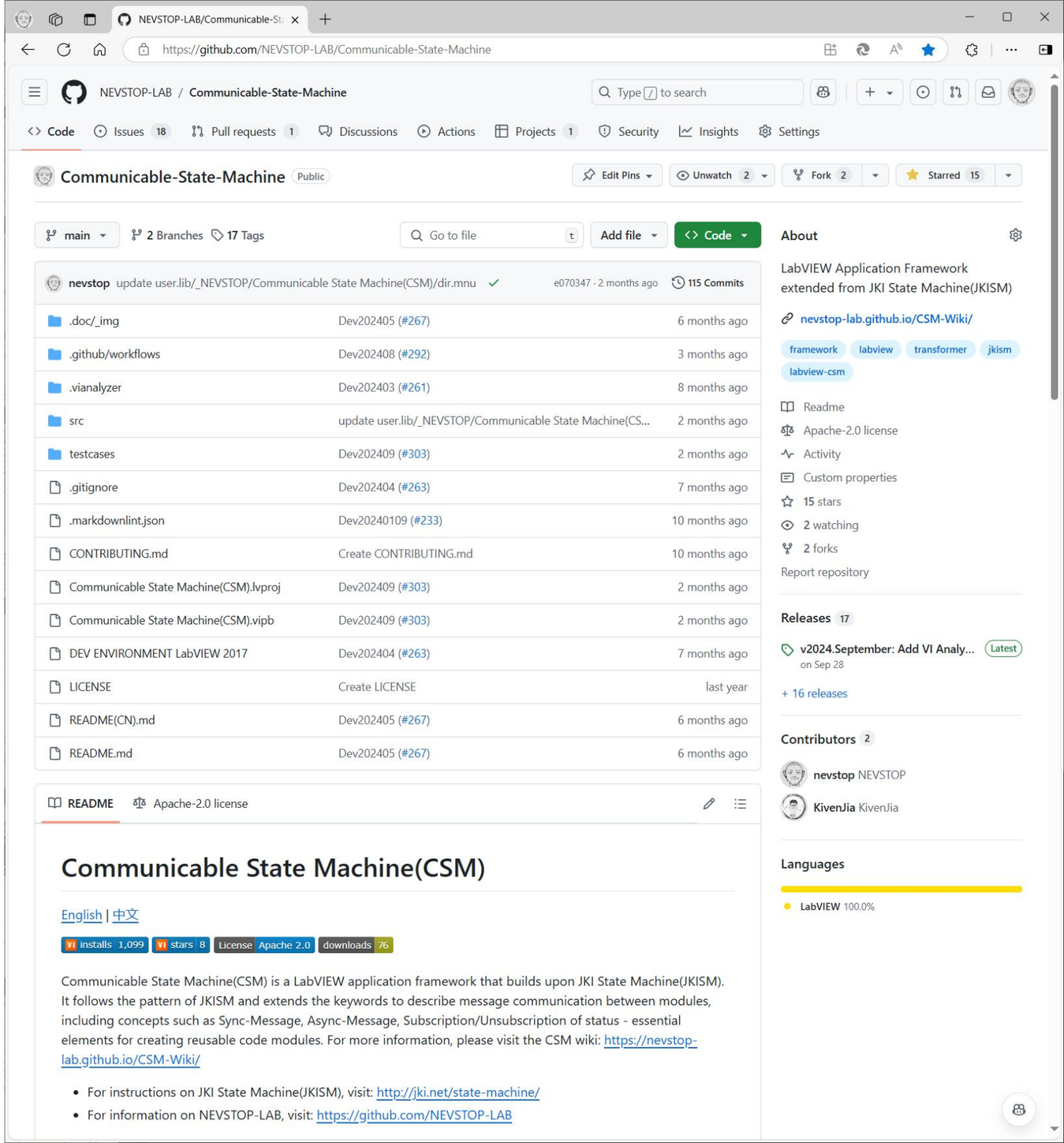
可通讯状态机框架(Communicable State Machine)

CSM简介

Communicable State Machine(CSM) 是一个基于 JKI State Machine（JKISM）扩展的 LabVIEW 应用框架.

它遵循 JKISM 的字符串状态描述模式，扩展了关键词，用于描述模块之间的消息通信，包括同步消息、异步消息、状态订阅/取消订阅等概念.





Framework Design(General Part)

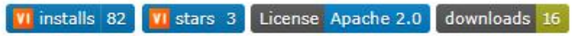
Shared among all customers

- Communicable State Machine(CSM) Framework
- Other Shared Libraries – Arguments..
- Architecture/Tools to support development process
- Low-code development Support – TS Customer Steps

Communicable State Machine

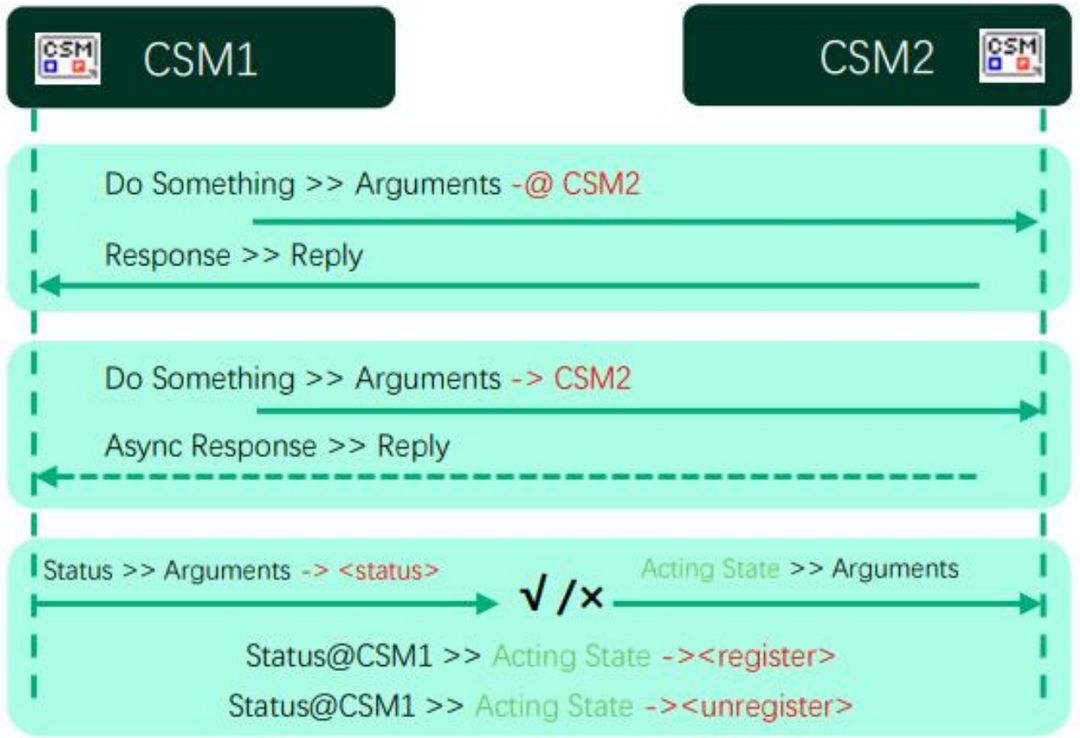
1. **Completed Functionality as DQMH/Workers/AF**
2. **Texted Based Communication(Sync, Async, Event)**
3. **动态的注册机制**
4. **No LabVIEW Queue, No LabVIEW Use Event**
5. **VI is Module. One VI for everything.**
6. **Addon to enhance MassData/API Parameter/Configuration …**

Communicable State Machine(CSM)



Communicable State Machine(CSM) is a LabVIEW application framework that builds upon JKI State Machine(JKISM). It follows the pattern of JKISM and extends the keywords to describe message communication between modules, including concepts such as Sync-Message, Async-Message, Subscription/Unsubscription of status - essential elements for creating reusable code modules. For more information, please visit the CSM wiki: https://github.com/NEVSTOP-LAB/Communicable-State-Machine/wiki

- For instructions on JKI State Machine(JKISM), visit: http://jki.net/state-machine/
- For information on Communicable State Machine(CSM), visit: https://github.com/NEVSTOP-LAB



Communicable-State-Machine: LabVIEW Application Framework extended from JKI State Machine(JKISM) (github.com)