Introduction to DQMH

The World's Most Popular 3rd-Party Framework for LabVIEW

Darren Nattinger

Chief TSE, CLA

NI

Before we get started

All of my presentations (including this one) are available at:

dnatt.org

(slides, demos, and links to video recordings)

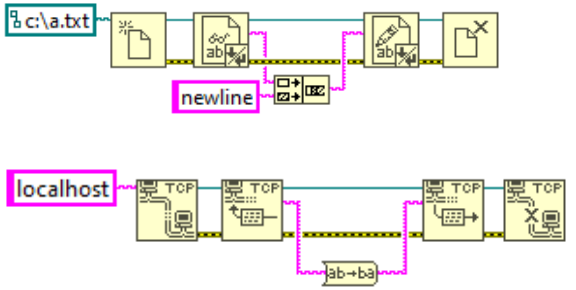
This presentation's link: http://bit.ly/dnattdqmhintro

Glossary

(As defined by me)

General Glossary

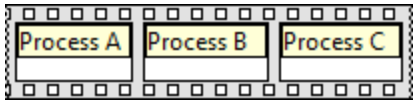
- **Process** – continuously-running code
  - Almost always a VI with one or more **while loops**
- **Reentrancy** – the ability for multiple instances of the same VI to run simultaneously
- **API** – application programming interface
  - Group of related functions organized in a logical manner



- **Business Logic** – application-specific code
  - Code that is *not* part of the framework being used
  - Written in pre-defined, documented places in the overall code

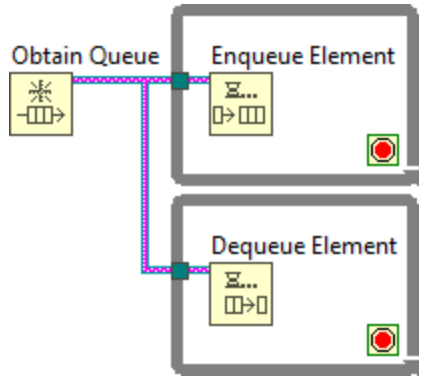
General Glossary

- **Synchronous Process** – ordered operation, dependent on completion of another process

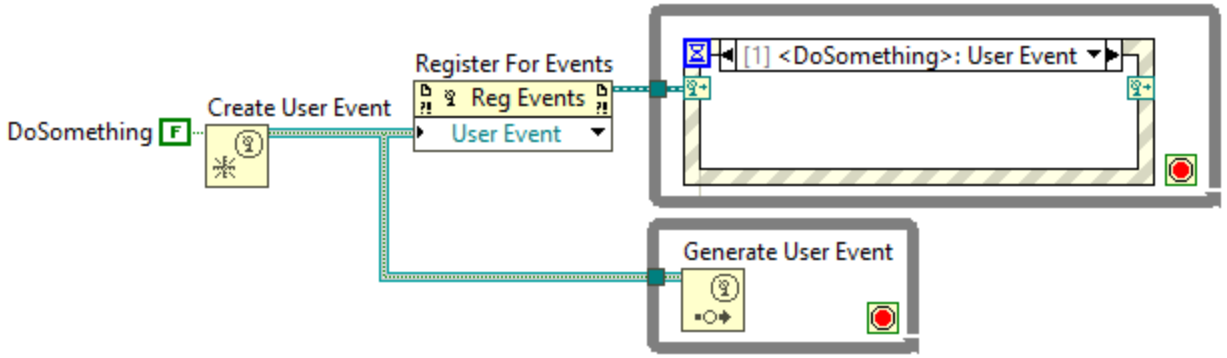


- **Asynchronous Process** – independent operation, runs in parallel with other processes

- **Queue** – LabVIEW API for passing data within or between processes



- **Event** – LabVIEW API for passing data within or between processes



General Glossary

- **Design Pattern -** Theoretical mechanism to execute synchronous or asynchronous code.
  - Examples: state machine, queued state machine, producer/consumer, **queued message handler**

- **Architecture -** Real-world implementation of one or more design patterns that facilitates execution of asynchronous code.
  - Templatized approach to implementing business logic
  - Examples: JKI State Machine, TLB', Messenger

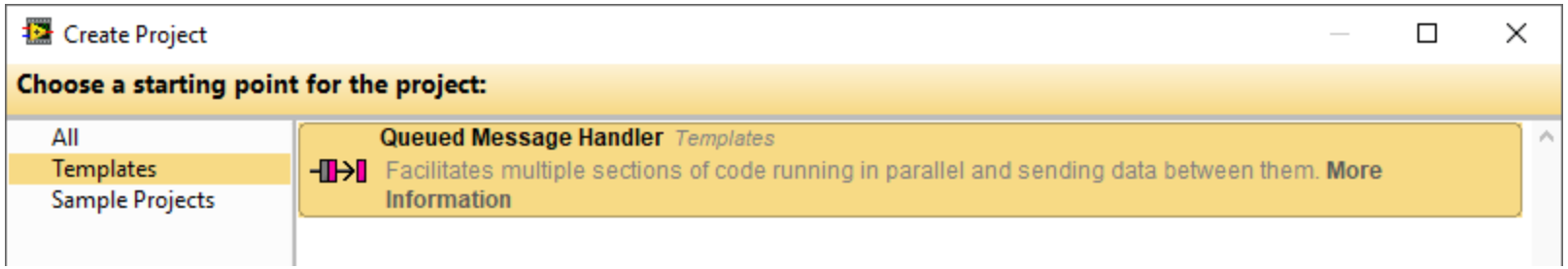
- **Framework**: consumer-grade architecture, with **documentation** and **tooling** to improve developer experience.
  - Examples: DQMH, Actor Framework, JKI State Machine Objects

NI QMH: A Brief Discussion

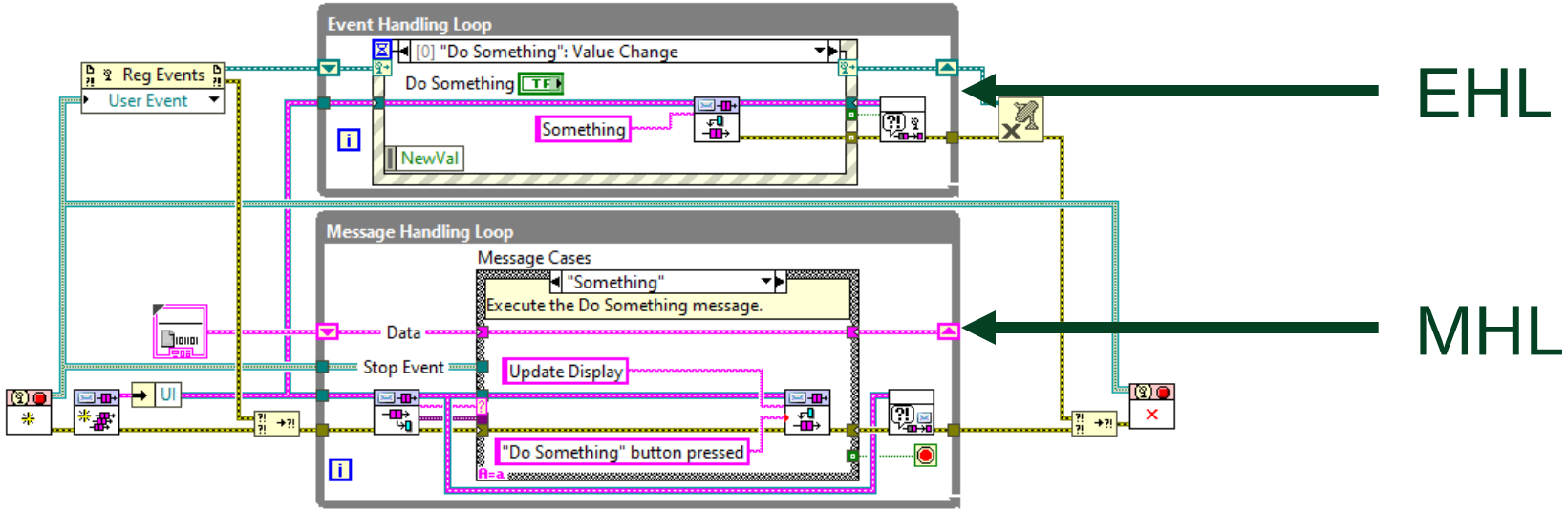
QMH: Queued Message Handler

NI QMH

- Project Template introduced in LabVIEW 2012

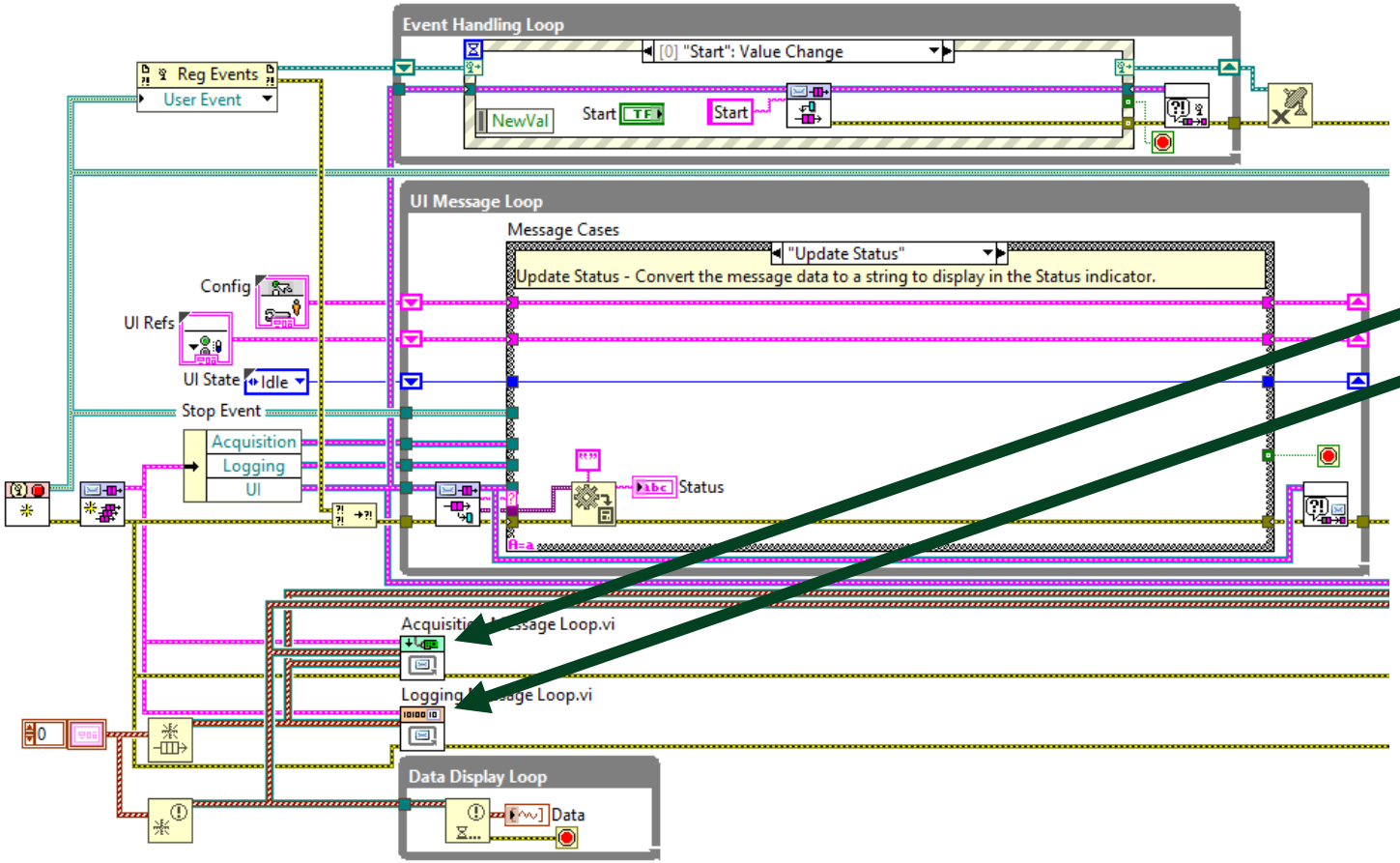


- The official NI template for using the QMH design pattern



NI QMH

- Continuous Measurement and Logging sample project built on NI QMH
  - NI-DAQmx installer includes a version that uses DAQmx API calls



- Same initial structure of NI QMH template
- Additional MHLs
  - Acquisition
  - Logging
- Queue references passed into subVIs that need access to them

Problems with NI QMH

- Communicating between processes is not straightforward
  - Wire more queue references into subVI connector panes?
  - Not scalable for systems with many modules
- Limited reusability of process VIs
  - For example, the Acquisition MHL VI has a reference to the UI Queue on its connector pane
- Difficulty in supporting reentrancy
  - No built-in mechanism to support multiple instances of the QMH VI running in parallel
- “Architecture”-level changes must be made manually
  - Adding frames to EHL and MHL can be tedious and error-prone

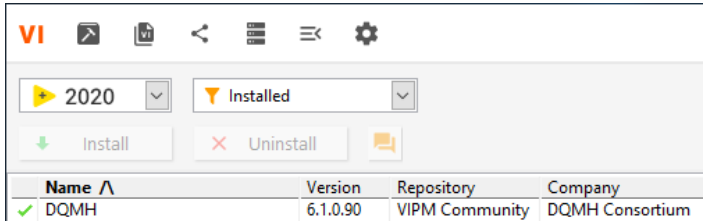
DQMH Basics

DQMH History

- 2015 - First public release
  - Originally developed by **D**elacor, an NI Alliance Partner
    - Chief Architect – Fabiola De la Cueva
- 2016 - LabVIEW Tools Network Product of the Year
- 2018 – DQMH Trusted Advisors Program
- 2020 – DQMH Podcast
- 2021 – Formation of DQMH Consortium
  - http://www.dqmh.org
- 2023 (present day) - Latest product release – DQMH 6.1

DQMH Basics

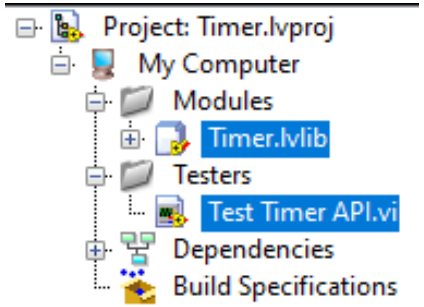
- Free to download and use



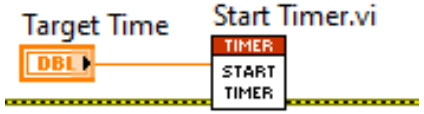
- Framework for LabVIEW to facilitate large application development
- Most popular 3rd-party framework for LabVIEW in the world
- DQMH Framework certification badge available from NI
- Same intra-process communication scheme as NI QMH (queues)
- Adds an inter-process communication scheme (user events)
- Designed to be accessible to **CLAD/CLD-level LabVIEW programmers**
  - Minimal use of LabVIEW classes out of the box

DQMH Glossary

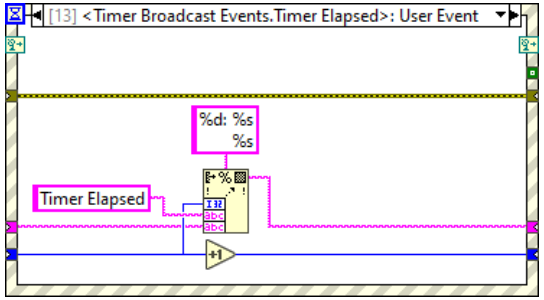
- **DQMH Module** – The basic building block of the DQMH framework
  - Asynchronous process with a well-defined **Public API**
  - A LabVIEW Library (.lvlib)
  - Can be **singleton** (non-reentrant) or **cloneable** (reentrant)



- **Request** – A way for the external world to ask the DQMH Module to do something
  - Communication mechanism is a User Event
  - Implemented as a VI in the module's Public API
  - Can optionally include a reply for whoever makes the request



- **Broadcast** – A way for the DQMH Module to tell something to the external world
  - Communication mechanism is a User Event
  - Zero or more external event structures might be registered for this event
    - (the module doesn't care)



DQMH Glossary

- **Main VI** – The main QMH VI of your application

- **API Tester** – A VI that lets you “test” the Public API of your DQMH Module
  - One of the most useful parts of DQMH

- **Scripting Tools** – What makes DQMH Framework a framework
  - Create/rename/validate module
  - Create/rename/remove/convert request and broadcast events
  - Workflow encourages best practices
    - Enter documentation when creating an event
    - Tester VI diagram shown after scripting, encouraging tester maintenance

Demo

Let’s create a simple DQMH module

Creating a DQMH Module

- Let's create a **Timer** DQMH Module
- Features:
  - Start a timer with a specified time duration
    - This will be a **request** event
  - Stop the timer
    - This will be a **request** event
  - Be notified when the timer elapses
    - This will be a **broadcast** event

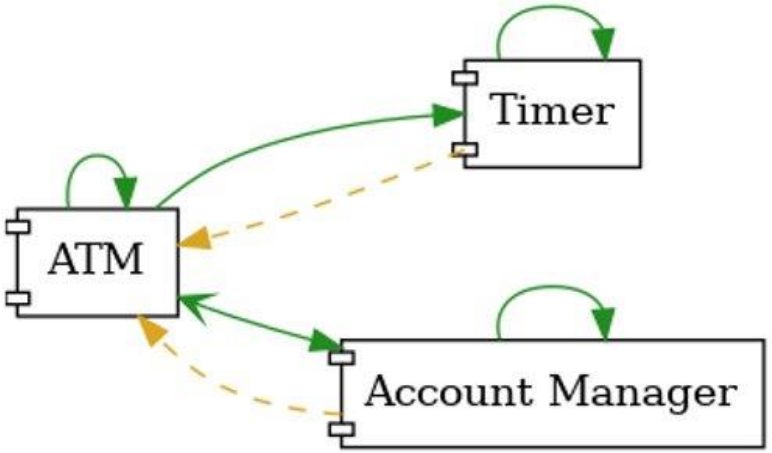
- **DEMO** – Creating the timer module
- **DEMO** – Integrating the timer into a larger application

Developer Roles

How are you interacting with DQMH?

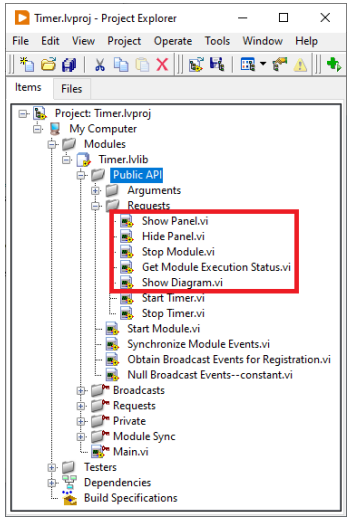
Are you a DQMH Module **developer**?

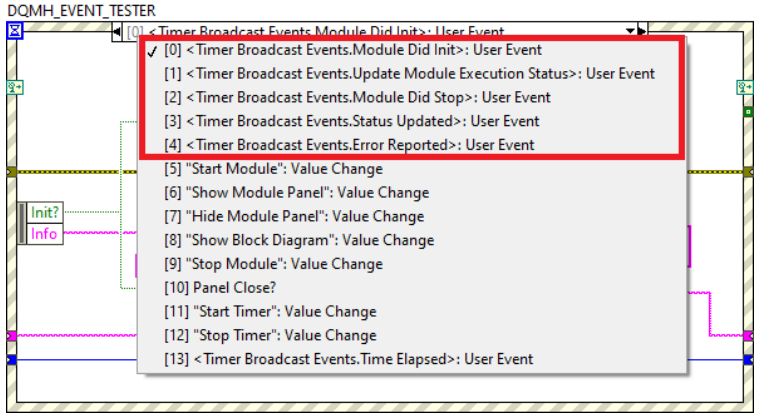
- **Keep your API tester up to date!**
- Follow DQMH Best Practices
- Use the scripting tools for all framework-level changes
  - Don't manually update framework VIs!
- Document **everything**
  - Module description
  - Event descriptions
  - Use Antidoc to auto-generate project documentation
- Create **module templates** to save time



Are you a DQMH Module **user**?

- If the module was installed via VIPM, it should have a palette
- If not, find what you need:
  - Requests: in the **Public API** folder in the module .lvlib
  - Broadcasts: in the event list of the Event Structure
- DQMH modules include several default requests and broadcasts:





Look at the tester VI to see examples of how to use the module's Public API

- ...and to verify the module is running correctly

Benefits of DQMH

Benefits of DQMH

- **Scripting Tools** automate framework-level changes
  - New module, new request, new broadcast, etc.
  - Module Validation: automatic application of bug fixes when upgrading!
  - Continued maintenance of API Tester during development
- QMH is a **familiar pattern** for CLAD/CLD-level LabVIEW programmers
- Handles **fundamental aspects of asynchronous LabVIEW programming** (with no additional effort on the part of the developer):
  - Starts when you tell it to start
  - Stops when you tell it to stop
  - Basic error management
  - Basic panel management
  - Built-in debugging
- Active DQMH **developer community**
- DQMH Consortium responsive to **feature requests**
- Many more benefits listed on the HSE site here

HAMPEL SOFTWARE ENGINEERING

How to Learn DQMH

How to Learn DQMH

- DQMH Documentation: How to Learn DQMH
- Tom's LabVIEW Adventure videos
- DQMH Consortium videos
- Official DQMH Training Course
- DQMH Podcast
- Shipping example
  - *[LabVIEW 20xx]\examples\DQMH Consortium\DQMH Fundamentals - Thermal Chamber*

Thanks for attending!

bit.ly/dnattdqmhintro

dnatt.org