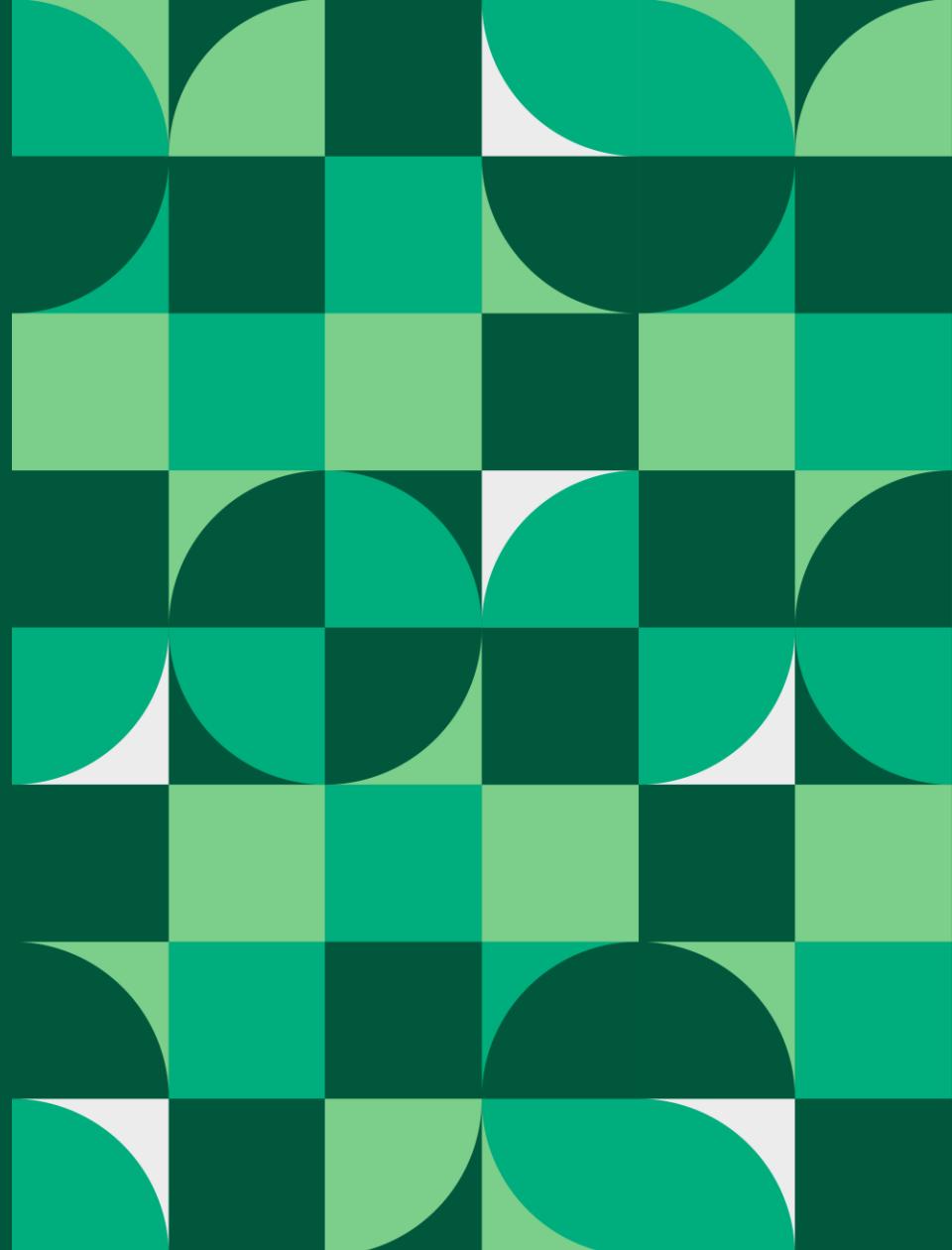


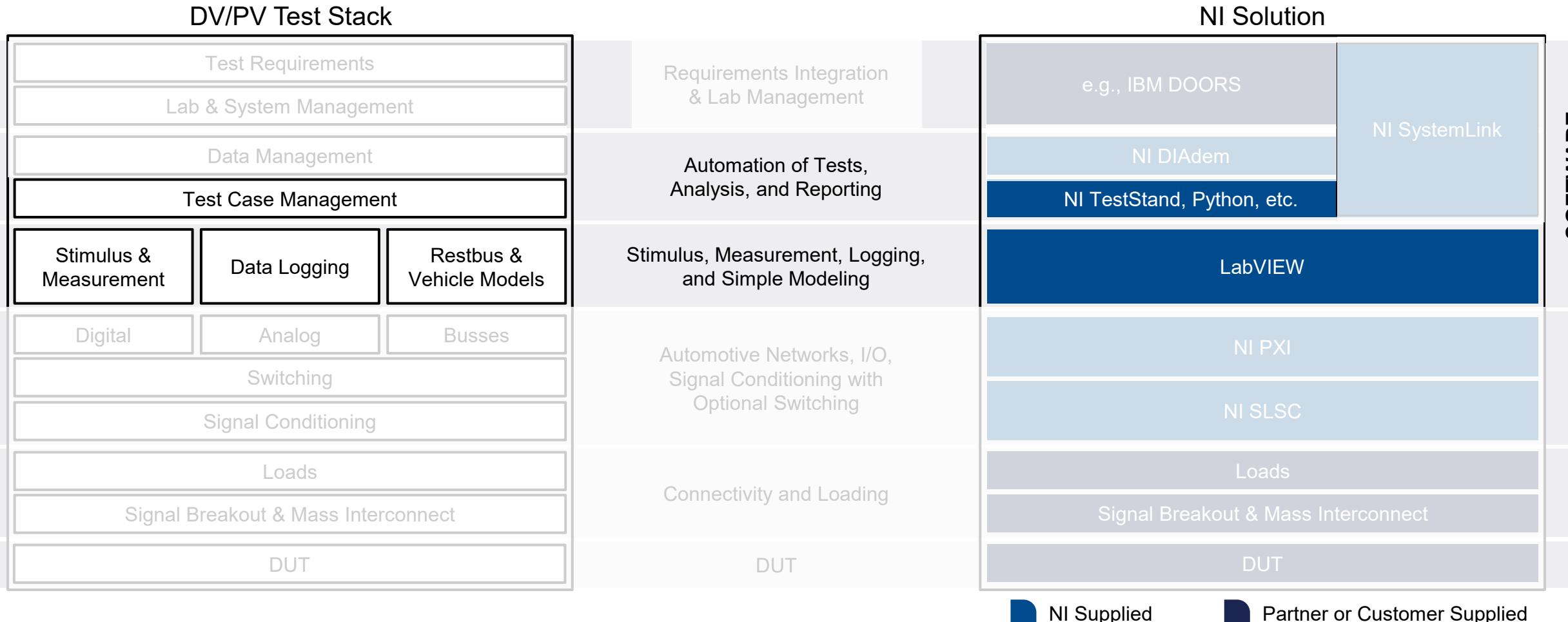
Design for reuse

Daniel Eaton

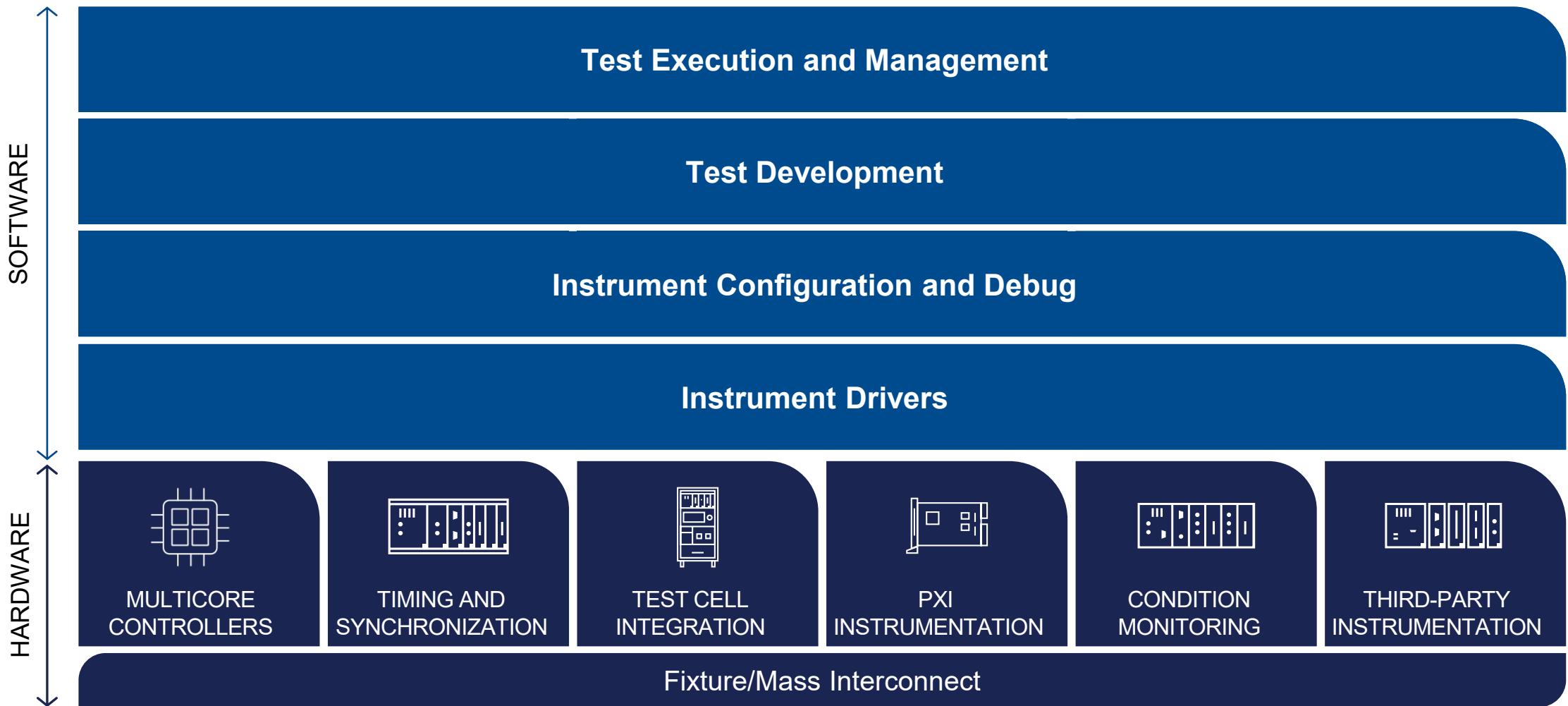
Chief Field Applications Engineer



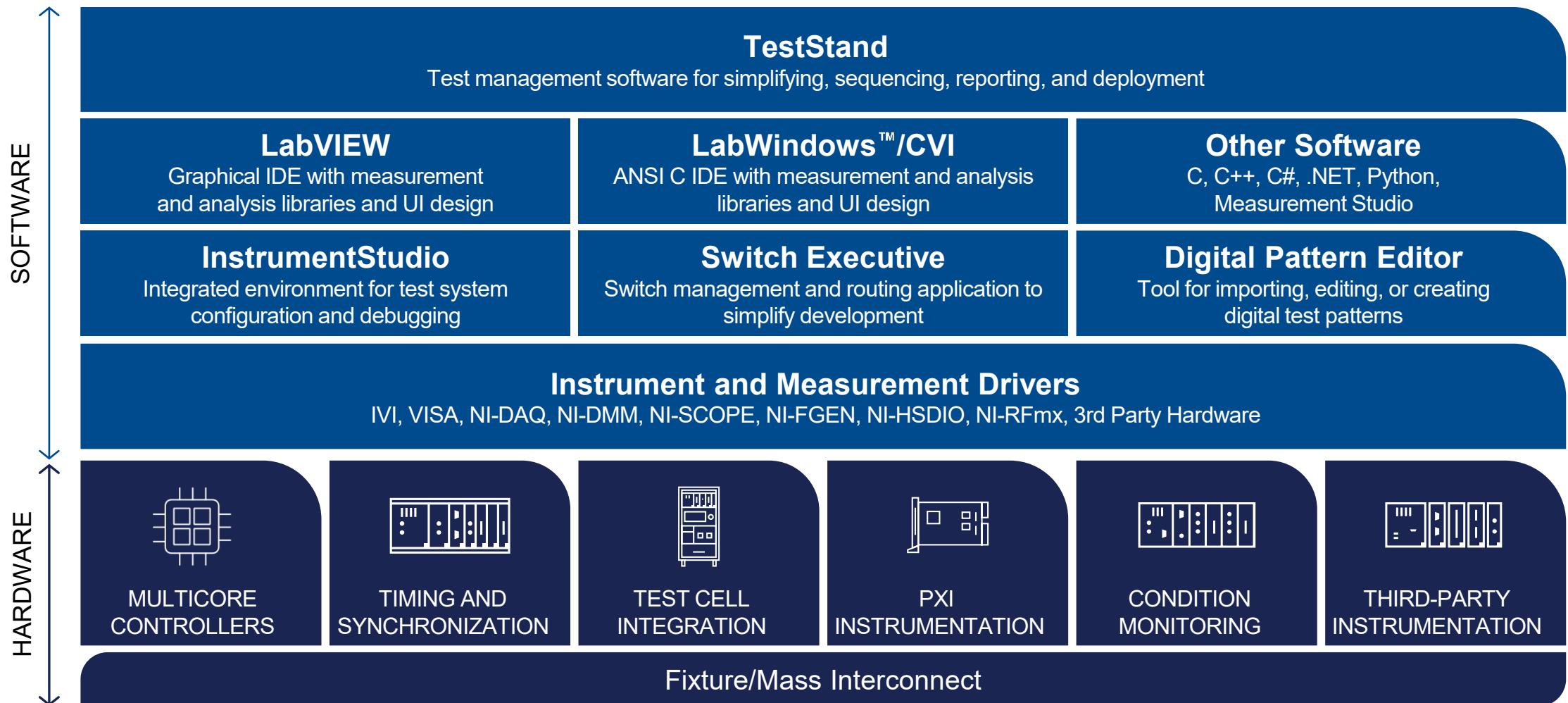
Hardware Validation Solution Overview



Architecture of an Automated Test System



Architecture of an Automated Test System





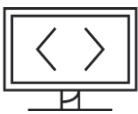
TestStand

A test executive software that accelerates system development and deployment for engineers in validation and production.



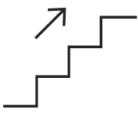
Automate Your System

Create, execute, and debug test sequences using an interactive environment



Leverage Existing Code

Use code from LabVIEW, Python, C/C++, or .NET



Test More, Faster

Use native parallel execution to reduce test time and functionality for advanced tasks such as sweeping and looping



Keep Track of Results

Track units, automatically generate reports, and store results to local or network databases



Deploy To Your Testers

Scale your operations by deploying sequences to numerous test stations with custom or pre-built operator interfaces

The screenshot shows the NI TestStand Sequence Editor interface. The main window displays a tree structure of test steps under the 'Steps' tab. The steps include:

- Setup (2):
 - Allow user to select which components to fail
 - .IN Simulation Dialog
 - .IN Initialize Test Fixture
 - <End Group>
- Main (13):
 - C Power On Test:
 - If the Power Test fails, do not test any additional components
 - ** If
 - C Battery Voltage Test
 - L LCD Video Test
 - E RF Test
 - M Microphone Input Test
 - P Speaker Output Test
 - For Each
 - C Button Test
 - End
 - Else
 - E Power Diagnostics
 - End
 - NOT FAIL Power On Test
 - (3.725), Numeric Limit Test., 3.65 <= x <= 3.8, volt, BatteryVoltageTest (MobileDeviceTest.dll)
 - Multiple Numeric Limit Test., LCD Display Test.vi, Number of Measurements: 2
 - Call RF Test in <Current File>
 - (0.983424142076), Numeric Limit Test., x > 0.95, Microphone Input Test.vi
 - (0.9879324535519), Numeric Limit Test., x > 0.95, Speaker Output Test.vi
 - Locals.currentButton in Locals.ButtonNames
 - Pass/Fail Test, ButtonTest (MobileDeviceTest.dll)
 - Call Power Diagnostics in <Current File>
- Cleanup (1):
 - .IN Disconnect Test Fixture
 - <End Group>





LabVIEW

A graphical programming environment engineers use to develop automated research, validation, and production test systems.



Create Professional User Interfaces

View data and control your test system via an interactive UI built from drag-and-drop UI elements



Integrate All Your Instruments

Acquire data from and control any instrument with 1000s of device drivers and industry-standard protocols



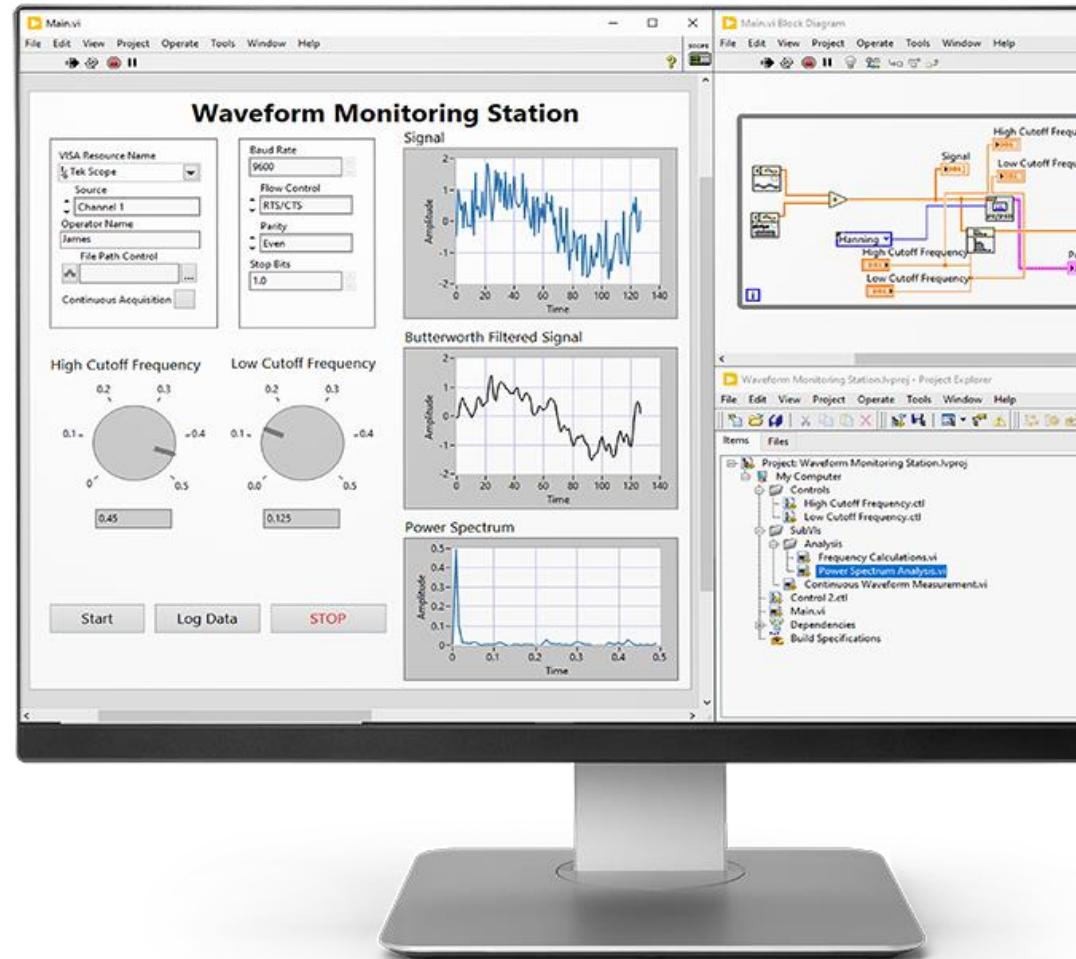
Program Like You Think

Save development time by creating and visualizing applications using data flow programming.



Use Other Code

Leverage other and existing code written in Python, C/C++, MATLAB®, and .NET



North Star: Save That Money...

- By maximizing test steps / sequences /software re-use for:
 - Any Rack
 - Any DUT
 - Any number of DUTs

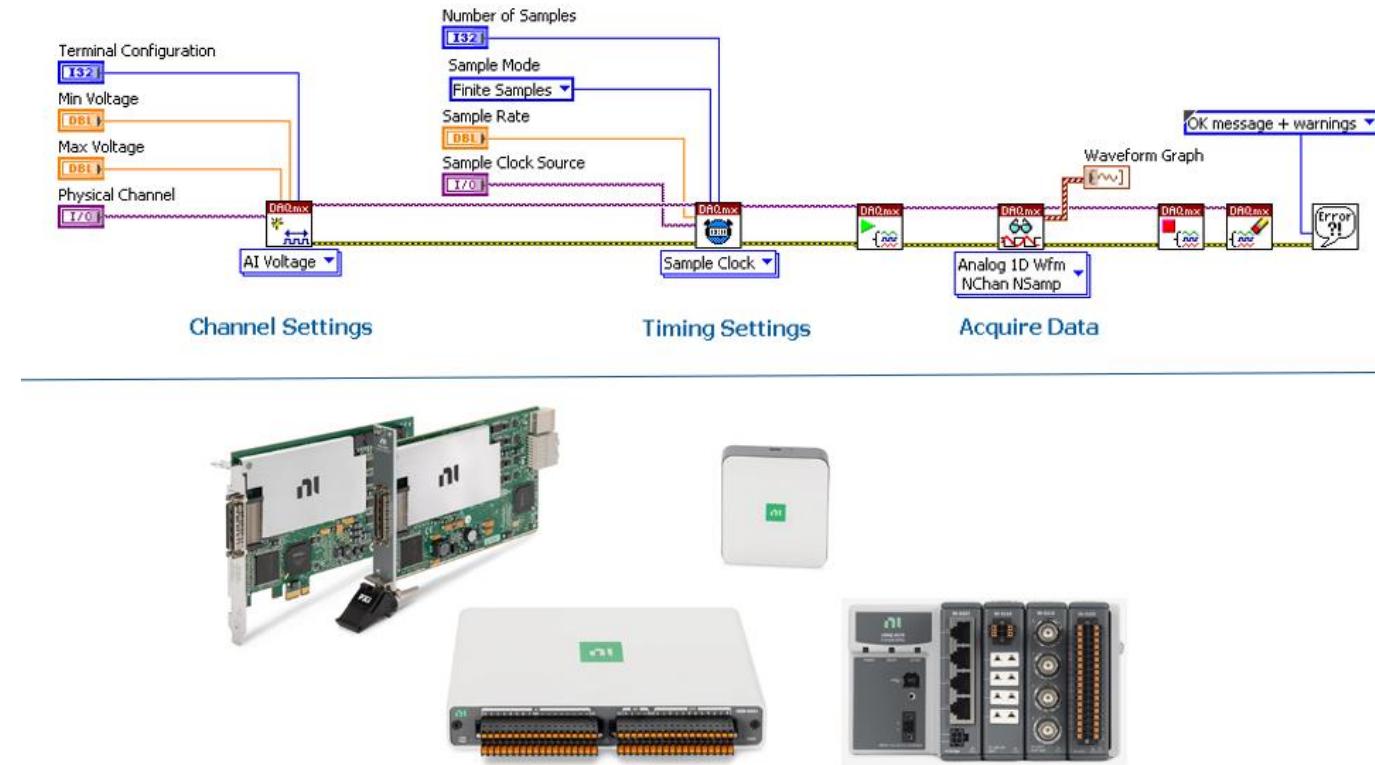
How: Any Rack

Use software to do

- Hardware Abstraction
- Pin Abstraction

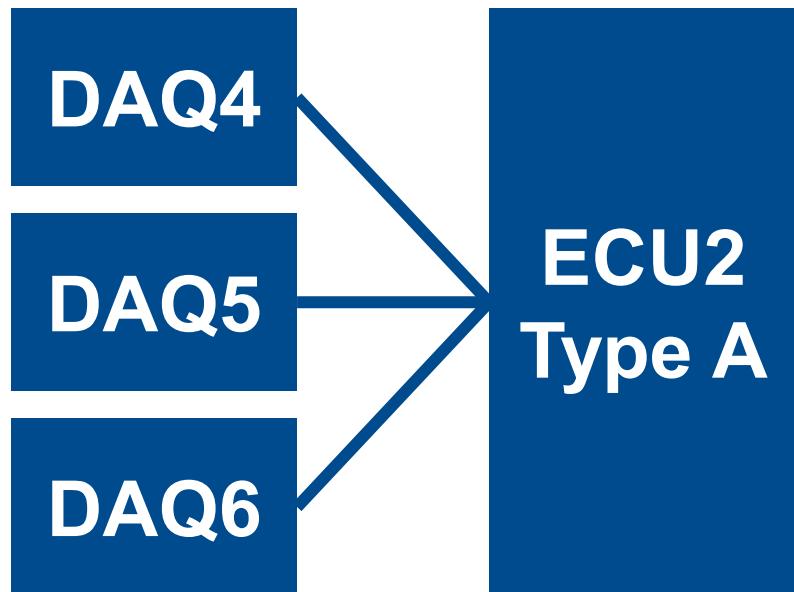
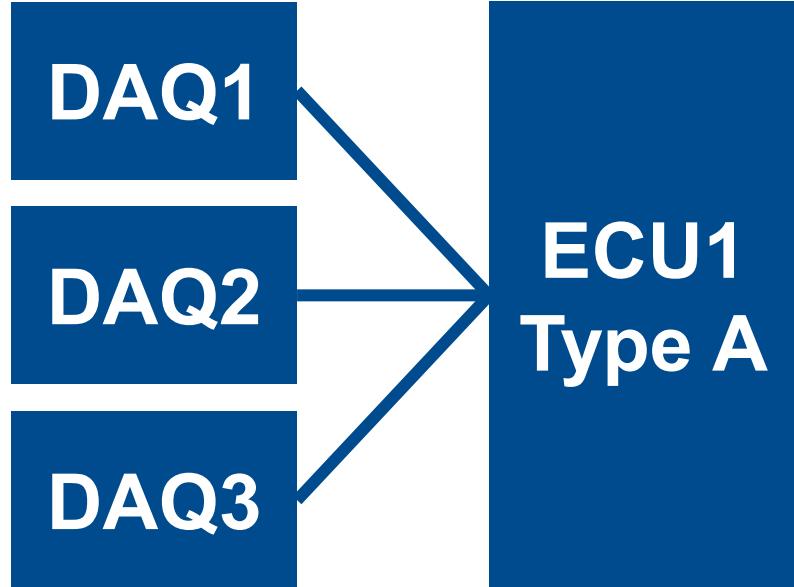
Hardware Abstraction

- Multiple Devices → One API (Software Interface)
- Pick rack specific capabilities → One API (Software Interface)
- NI APIs are HAL for NI devices
 - NI DAQmx: Multi-purpose DAQ
 - NI XNET: Automotive Buses
 - NI Switch: Switches
 - Etc
- Third-party hardware requires a higher-level, user defined HAL



Pin Abstraction

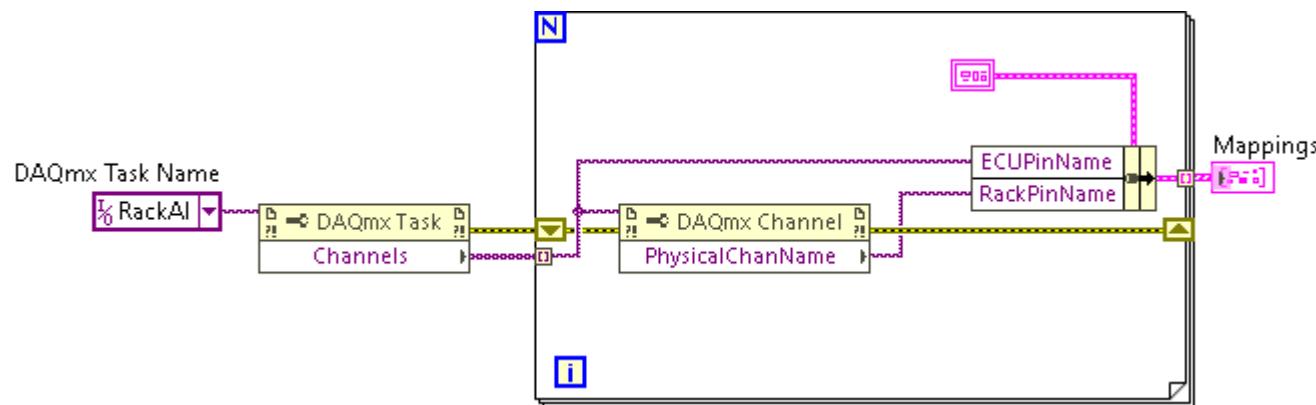
- Abstract rack hardware from sequences/steps
- Scenario: 3 Devices → 1 DUT
- Which option is better re-use?
 - DAQ1/AI1 or ECU.Pin1
 - DAQ2/DO1 or ECU.Pin2
 - DAQ3/AO1 or ECU.Pin3
- Should test sequences care about devices or your ECU?
- Scenario: 6 Devices → 2 DUTs
- Approach: Map ECU Pin → Rack specific pin



Pin Abstraction: Mapping Examples

- DAQmx Global Channels + Naming Convention

Configuration			
Triggering			
Advanced Timing			
Logging			
Channel Settings			
Hide Details <>			
Order			
DUT0_Pin0_SignalName	0	PXI15slot2/ai0	PXIe-6375
DUT0_Pin1_SignalName	1	PXI15slot2/ai1	PXIe-6375
DUT1_Pin0_SignalName	2	PXI15slot2/ai2	PXIe-6375
DUT1_Pin1_SignalName	3	PXI15slot2/ai3	PXIe-6375



Pin Abstraction: DAQmx Global Channels

Pros

- Simple
- GUI Based Configuration
- Naturally include channel properties (voltage levels, etc)
- Exportable
- Allows for naming scheme
- Task required regardless

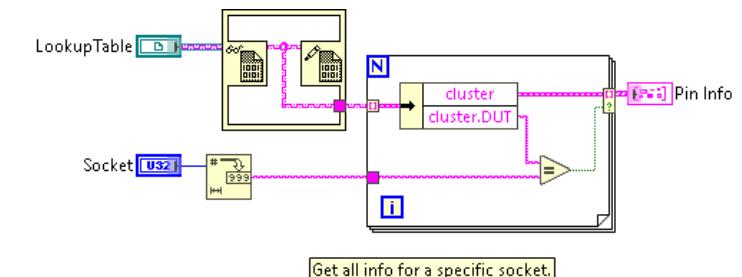
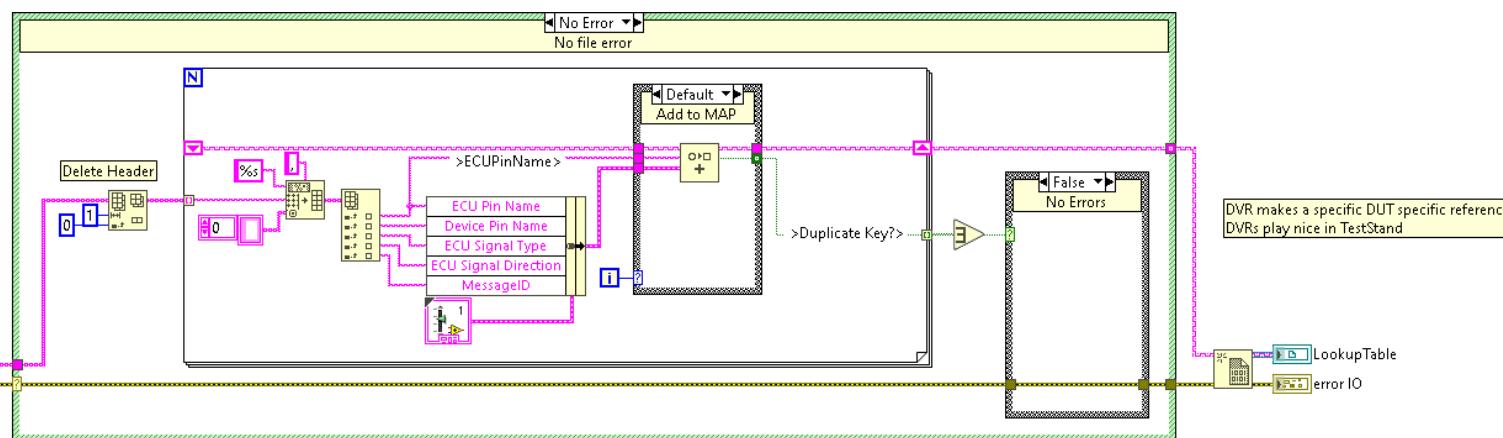
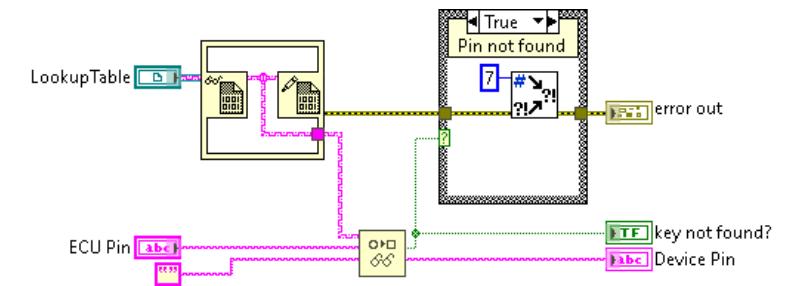
Cons

- GUI based configuration tedious for larger channel groups
- Specific to NI hardware
- Does not allow “custom parameters”
- Starts to break down in multi-up, multi-device systems

Pin Abstraction: Mapping Examples

- Mapping File & Lookup

A	B	C	D	E	F	
1	DUT	ECU Pin Name	Device Pin Name	ECU Signal Type	ECU Signal Direction	MessageID
2	0	ECU.Connector0.Pin0	Device1/AO0	Voltage	Input	0
3	0	ECU.Connector0.Pin1	Device1/AO1	Voltage	Input	1
4	0	ECU.Connector0.Pin2	Device1/AO2	Voltage	Input	2
5	0	ECU.Connector0.Pin3	Device1/AO3	Voltage	Input	3
6	0	ECU.Connector0.Pin4	Device1/AO4	Voltage	Input	4
7	0	ECU.Connector0.Pin5	Device1/AO5	Voltage	Input	5



Pin Abstraction: Mapping File & Lookup

Pros

- Better for larger channel groups
- Not specific to NI hardware
- Allows for custom parameters
- Holds up under multi-up, multi-device systems
- Allows multiple lookup options (e.g. All pins by DUT, All pins by type, etc)
- Exportable
- Allows for naming scheme

Cons

- Requires front end development investment
- Slight duplication around tasks
- More work to validate config

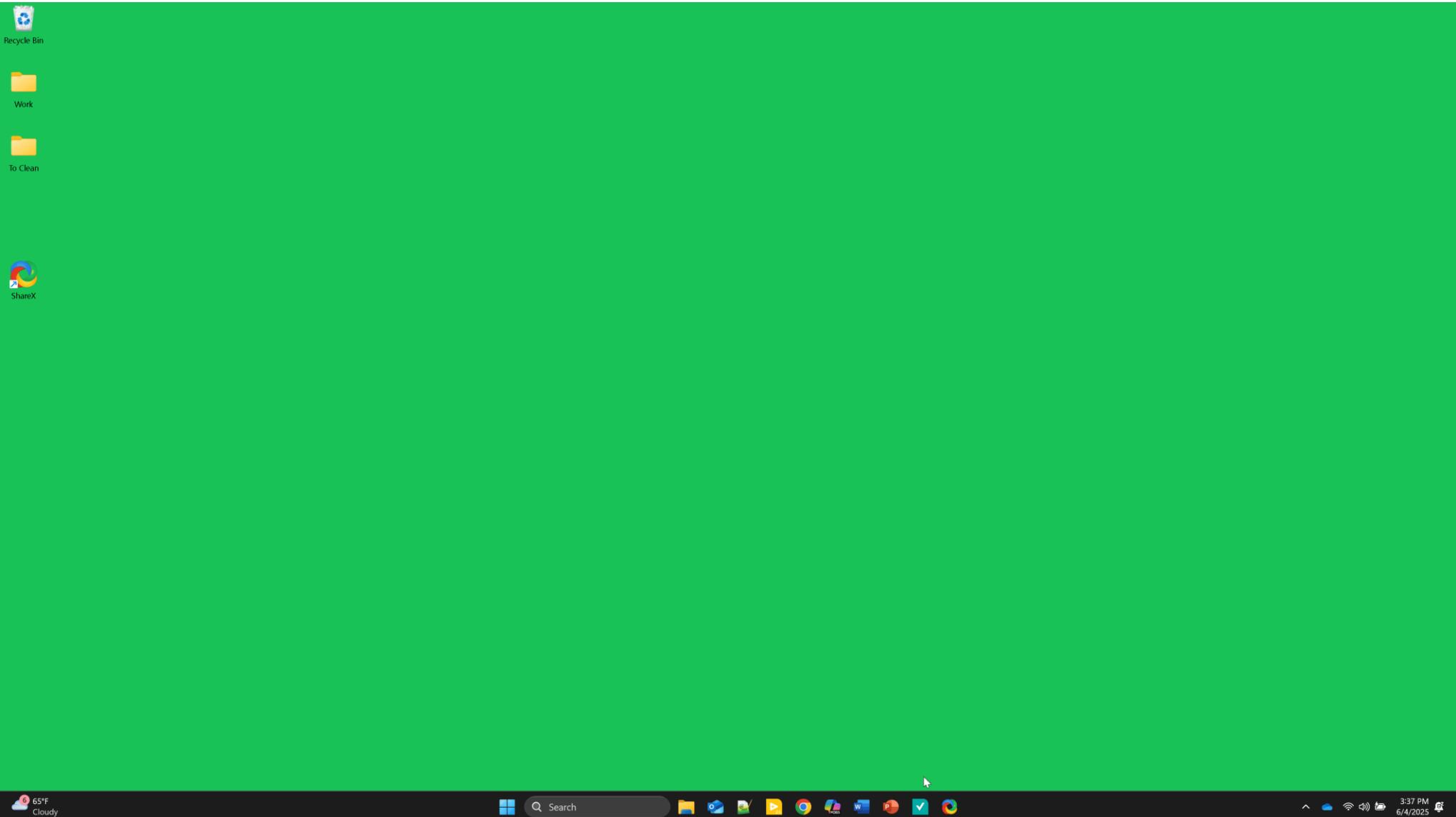
Pin Abstraction: Buses Apply Too

- CAN
- Automotive Ethernet
- Etc

	A	B	C	D	E	F
1	NAME	ECU Port	Tester Port	Baud Rate	FD Baud Rate	Termination
2	DUT0.ECU.RADAR.CAN_FD	0	CAN4	2.17433E+11	3.60453E+16	0
3	DUT0.ECU.HUD_CAN	1	CAN2	500000	0	0
4	DUT0.ECU.CAN3_SPARE	2	CAN3	500000	0	0
5	DUT0.ECU.HS_CAN_FD	3	CAN1	2.17433E+11	3.60453E+16	1

	A	B	C	D	E	F
1	NAME	IP Address	Location	Network	Rate	MAC
2	DUT0.ECU.ITALIA	10.10.10.101	ENET1	SW1	100	00:80:2F:30:64:11
3	DUT0.ECU.LIDAR	10.10.10.102	ENET2	SW1	1000	00:80:2F:30:64:12
4	DUT0.ECU.RADAR3	10.10.10.103	ENET3	SW1	100	00:80:2F:30:64:13
5	DUT0.ECU.RADAR2	10.10.10.104	ENET4	SW1	100	00:80:2F:30:64:14
6	DUT0.ECU.RADAR1	10.10.10.105	ENET5	SW1	100	00:80:2F:30:83:E4
7	DUT0.ECU.ECG	10.10.10.108	ENET8	SW1	100	00:80:2F:30:83:E7
8	DUT0.ECU.RADAR5	10.10.10.109	ENET9	SW1	100	00:80:2F:30:85:40
9	DUT0.ECU.RADAR4	10.10.10.110	ENET10	SW1	100	00:80:2F:30:85:41
10	DUT0.ECU.FFLIDAR	10.10.10.112	ENET12	SW1	1000	00:80:2F:30:85:43

Demo: Creating a Pin Map



Summary: Creating a Pin Map

- Used a config file
- Used a logic naming scheme that enables multi-up
- Had custom parameters
- Had multi-up, multi-device systems
- Allowed multiple lookup options (e.g. All pins by DUT, All pins by type, etc)
- Pin Abstraction: Converts rack IO → ECU Pin Measurements
- Sequences/Steps don't care about rack configuration; makes it re-usable

Summary: Hardware & Pin Abstraction

- Hardware Abstraction: Provides one API for varying rack functionality
 - Use NI Drivers or write your own
- Pin Abstraction: Converts rack IO → ECU Pin Measurements
 - Use a mapping config file and look up table
- Use a logic naming scheme that enables multi-up
 - E.g. DUT#.ECU.Connector.Pin.MeasurementType
- Sequences/Steps reference ECU Pins; nice for the user
- Sequences/Steps don't care about rack configuration; makes it re-usable

Any DUT

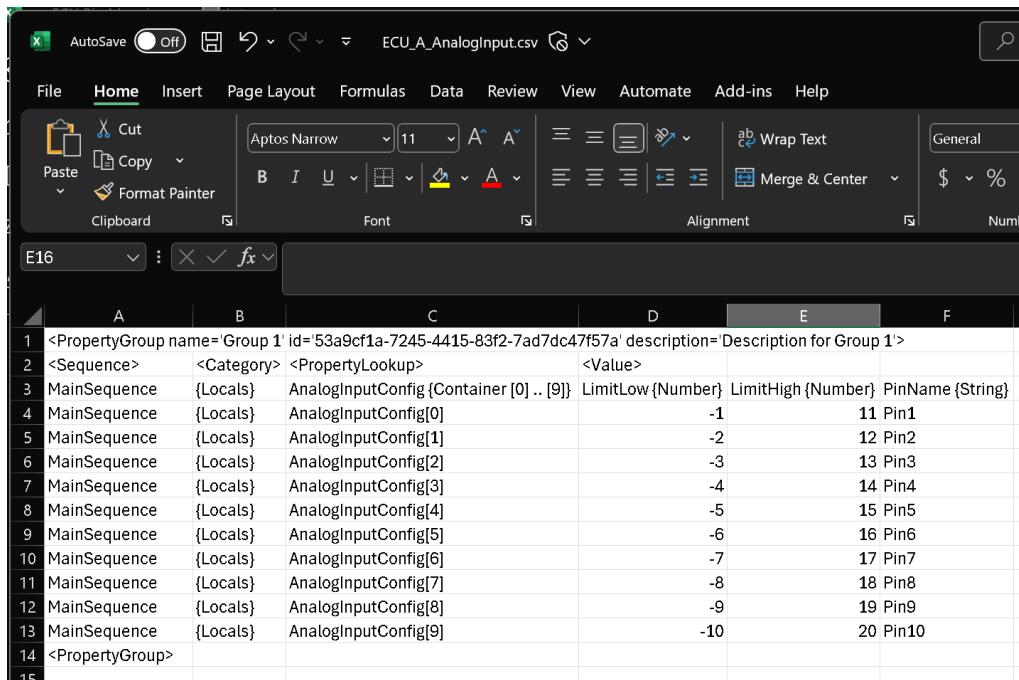
- Goal: One test step → Any ECU
- Built on-top of abstraction
- Configuration files
- Step parameterization
- For Loops
- At the step level
- Assumes common ECU communication approach

Abstraction

- See previous...
- Required to remove test rack specifics from sequences

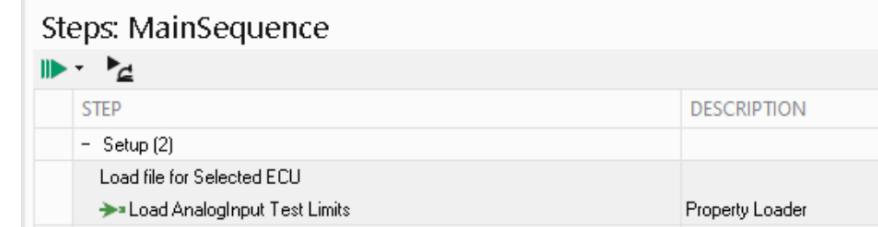
Configuration Files

- ECU Limits
- DUT Control Messages / IDs
- Option 1: Use NI TestStand property loader
- Option 2: Use your own file format and parsing functions



The screenshot shows a Microsoft Excel spreadsheet titled "ECU_A_AnalogInput.csv". The data is organized into columns A through F. Column A contains XML-style configuration code, while columns B through F contain specific parameters like AnalogInputConfig and PinName.

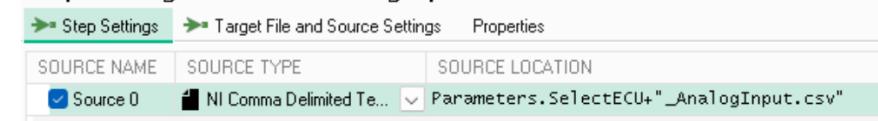
A	B	C	D	E	F
1	<PropertyGroup name='Group 1' id='53a9cf1a-7245-4415-83f2-7ad7dc47f57a' description='Description for Group 1'>				
2	<Sequence>	<Category>	<PropertyLookup>	<Value>	
3	MainSequence	{Locals}	AnalogInputConfig {Container [0] .. [9]}	LimitLow {Number}	LimitHigh {Number} PinName {String}
4	MainSequence	{Locals}	AnalogInputConfig[0]	-1	11 Pin1
5	MainSequence	{Locals}	AnalogInputConfig[1]	-2	12 Pin2
6	MainSequence	{Locals}	AnalogInputConfig[2]	-3	13 Pin3
7	MainSequence	{Locals}	AnalogInputConfig[3]	-4	14 Pin4
8	MainSequence	{Locals}	AnalogInputConfig[4]	-5	15 Pin5
9	MainSequence	{Locals}	AnalogInputConfig[5]	-6	16 Pin6
10	MainSequence	{Locals}	AnalogInputConfig[6]	-7	17 Pin7
11	MainSequence	{Locals}	AnalogInputConfig[7]	-8	18 Pin8
12	MainSequence	{Locals}	AnalogInputConfig[8]	-9	19 Pin9
13	MainSequence	{Locals}	AnalogInputConfig[9]	-10	20 Pin10
14	<PropertyGroup>				
15					



The screenshot shows the "Steps: MainSequence" table in NI TestStand. It lists two steps: "Setup (2)" and "Load AnalogInput Test Limits". The "Load AnalogInput Test Limits" step is associated with the "Property Loader".

STEP	DESCRIPTION
- Setup (2)	
Load file for Selected ECU Load AnalogInput Test Limits	Property Loader

Step Settings for Load AnalogInput Test Limits



The screenshot shows the "Step Settings" dialog for the "Load AnalogInput Test Limits" step. The "SOURCE NAME" is set to "Source 0", "SOURCE TYPE" is "NI Comma Delimited Te...", and "SOURCE LOCATION" is "Parameters.SelectECU+_AnalogInput.csv".

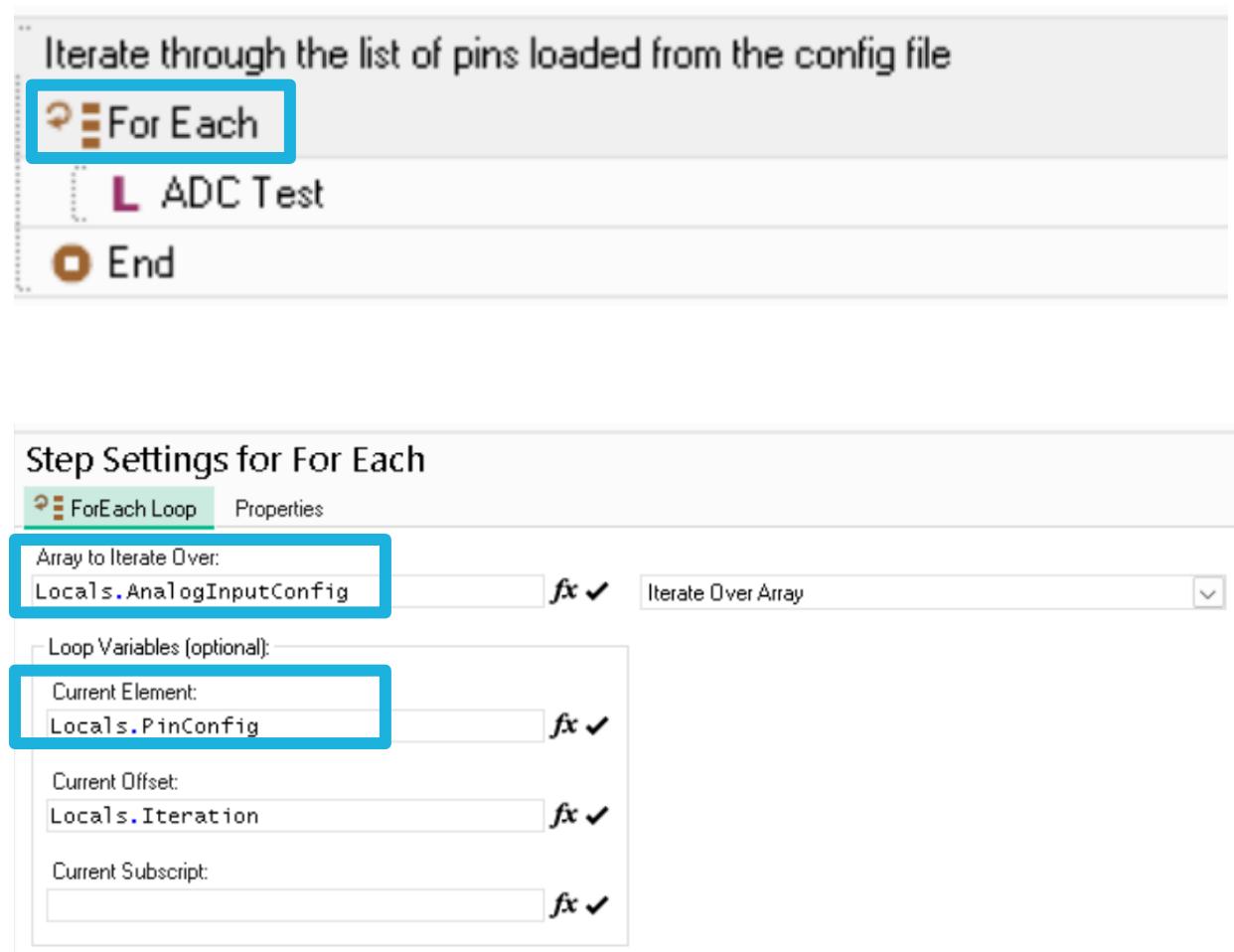
SOURCE NAME	SOURCE TYPE	SOURCE LOCATION
Source 0	NI Comma Delimited Te...	Parameters.SelectECU+_AnalogInput.csv

Step Parameterization

- Build step for common functions
 - Send message & measure pin
 - Generate pin & read message
 - Bus bandwidth
 - Internal chipset check
 - Typical chipsets
 - etc
- Create parameters for test specific settings
 - Pins
 - Messages
 - Limits

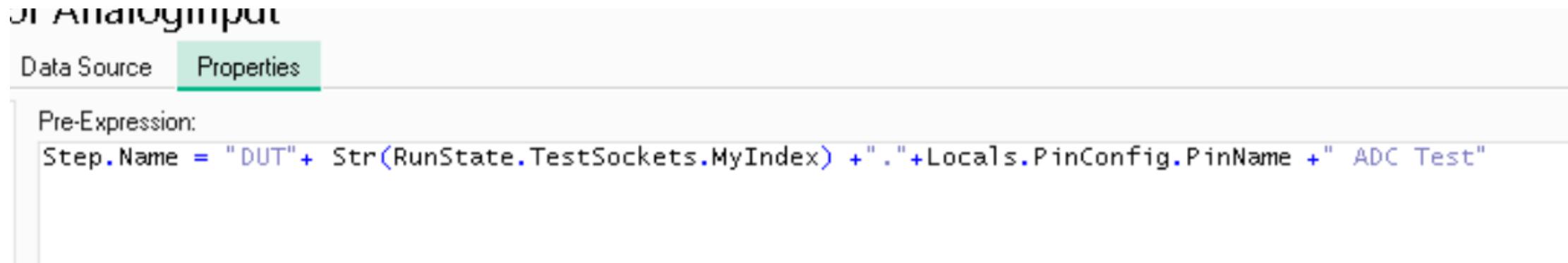
For Loops

- Abstracts the number of iterations
 - Number of pins
 - Number of tests
- Arrays are your friends
- Pass array element to local variable
- Capture iteration



Pro Tips

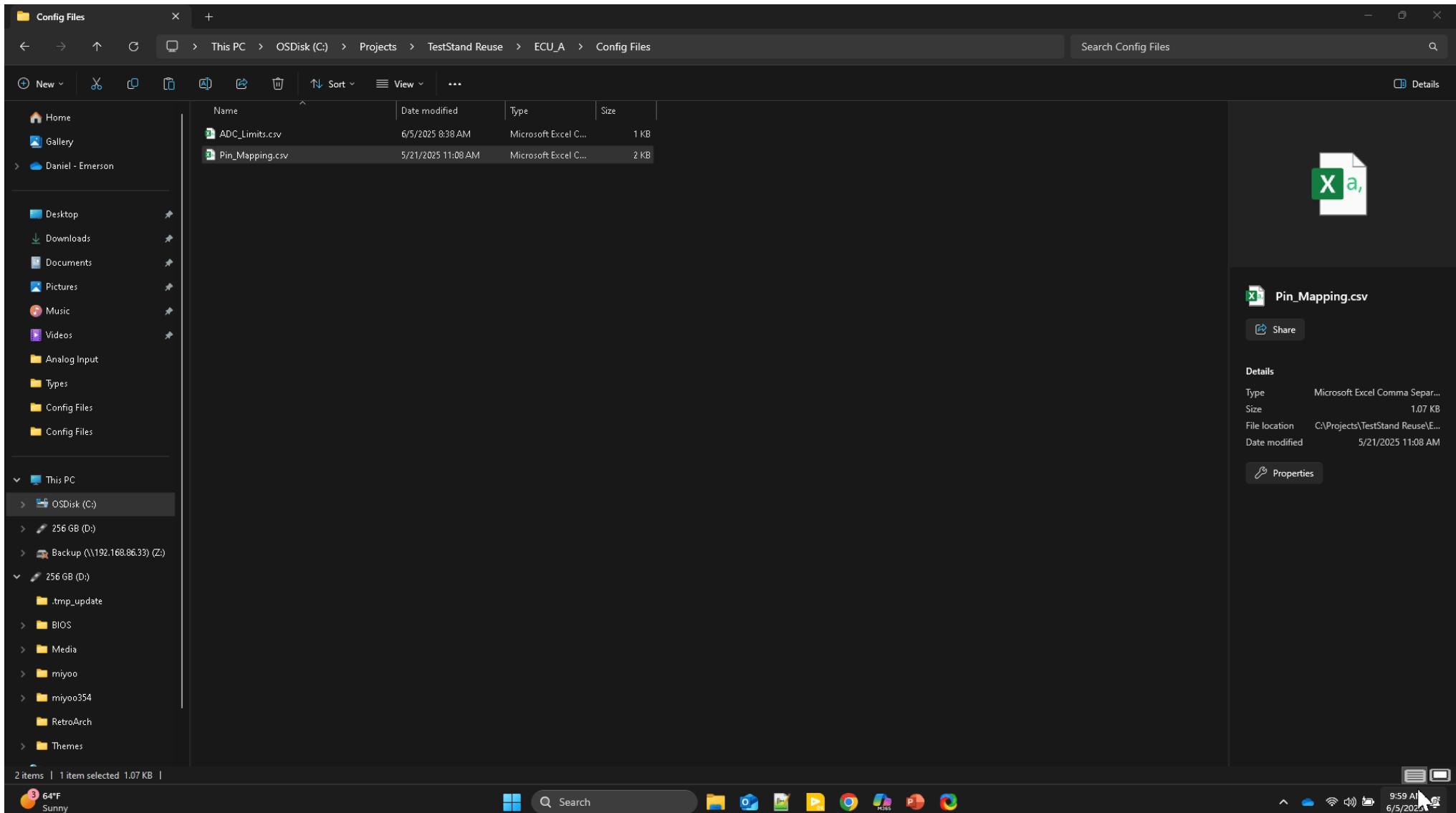
- Change Step Name at Runtime:



The screenshot shows the 'Properties' tab of a LabVIEW VI. The 'Pre-Expression' field contains the following code:

```
Step.Name = "DUT"+ Str(RunState.TestSockets.MyIndex) +"." +Locals.PinConfig.PinName +" ADC Test"
```

DEMO – Different Config & Limits Files



DEMO – Same TestStand Sequence

The screenshot shows the NI TestStand Sequence Editor interface. The main window displays a sequence named "ECU ADC Test.seq" under the "Steps: MainSequence" tab. The sequence consists of several steps, including "Setup (7)", "If", "Initialize Lookup Table", "Cache pin settings for traceability", "List Socket Properties", "Update limits path based on ECU Type", "Set Limits File", "End", "Load limits file for selected ECU", "Load AnalogInput Test Limits", "Initialize connection to ECU based on socket", "Initialize HW for AI test", "Main (5)", "If", "Iterate through the list of pins loaded from the config file", and "For Each". The "Initialize Lookup Table" step is currently selected, showing its properties in the "Step Settings for Initialize Lookup Table" dialog. This dialog includes tabs for "Module" and "Properties". The "Properties" tab is active, displaying a table with columns for NAME, VALUE TO LOG, TYPE, and CONDITION. It lists parameters such as "Mapping file path [In]" (Path), "LookupTable [Out]" (Number U32), and "error out [Out]" (Container). The "CONDITION" column contains logic expressions like "fx > fx & fx < fx". To the right of the main window, there is an "Variables" palette showing a list of variables with their names, values, and types. The "NAME" column includes "Locals ('MainSequence')", "123 ConnectionRef", "PinLimit", "ADCLimits", "Iteration", "ResultList", "Parameters ('MainSequence')", "123 LookupTableRef", "LimitsFile", "Called", "ECUType", and "FileGlobals ('ECU ADC Test.seq')". The "TYPE" column indicates the variable type for each entry.

Summary: Different Configurations; Same Sequence

- Used pin mapping config file to abstract rack specifics
- Used limits config file to abstract away ECU specifics
- Used For Loop to abstract number of pins / tests
- Used ECU name to parameterize path, step names, etc
- Used pin name to parameterize step names
- Used pin name as cross reference between files
- Result: Different ECU & rack configurations → Same TestStand Sequence

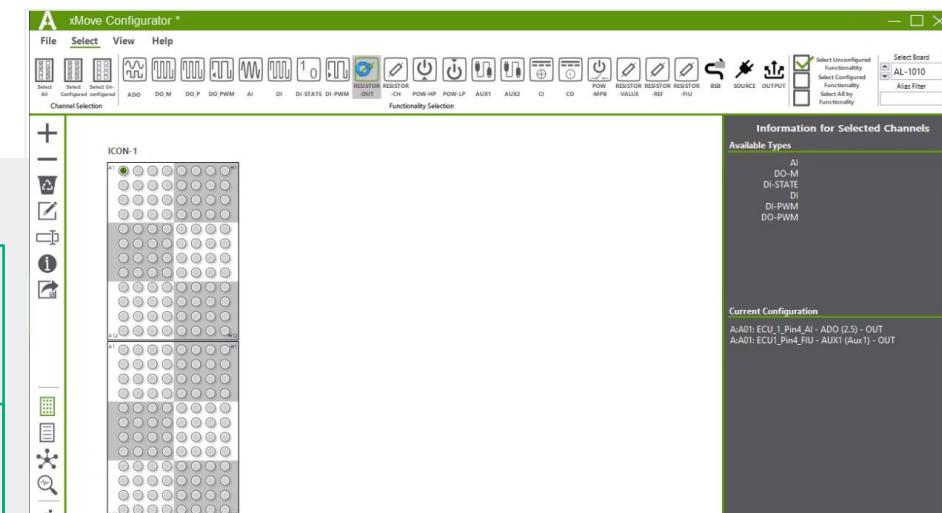
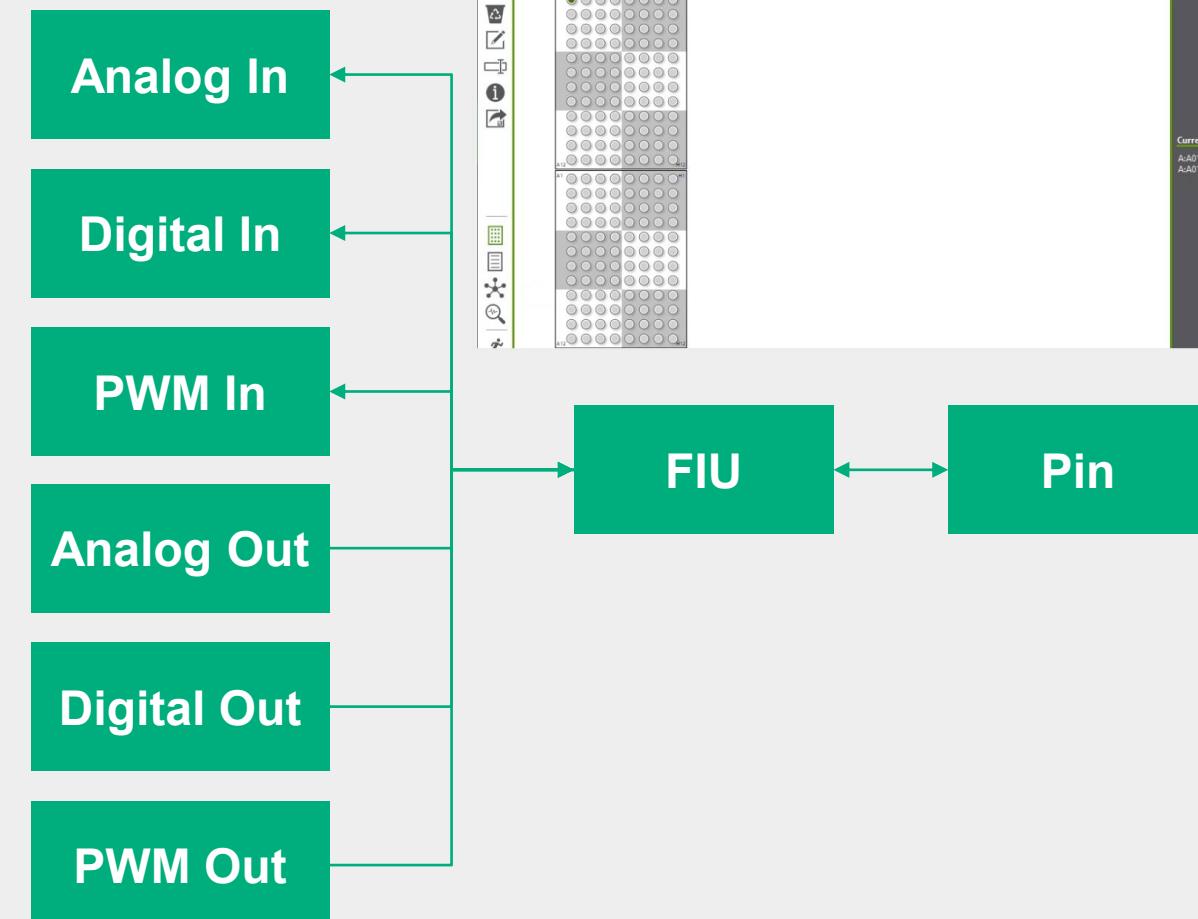
Hardware Racks for Any DUT

- Typical: Rack designed per DUT
- Abstraction allows software tests to be re-used
- Hardware can be pulled out and re-used program to program (if allowed)
 - Typically requires additional hardware based on IO count/type
- Aliaro created software-defined hardware that enables direct program to program reuse

NI / Aliaro Approach

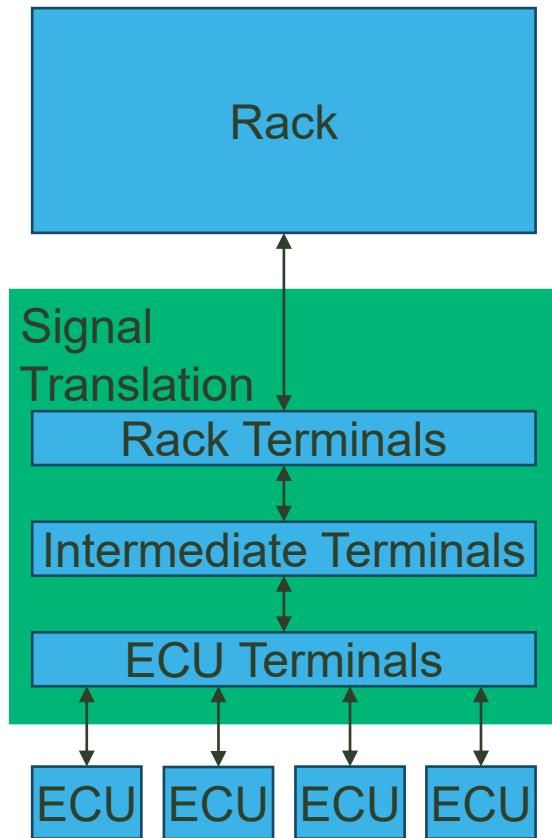
Truly Flexible Pins (Oversimplified)

- Software Selectable w/ appropriate protections
- Can be input, output or both (allows loop back)
 - Closest competition is in or out, not both
- FIU on every pin
- 12 & 32 channel cards based on requirements
 - Closest competition is 10 channels per card

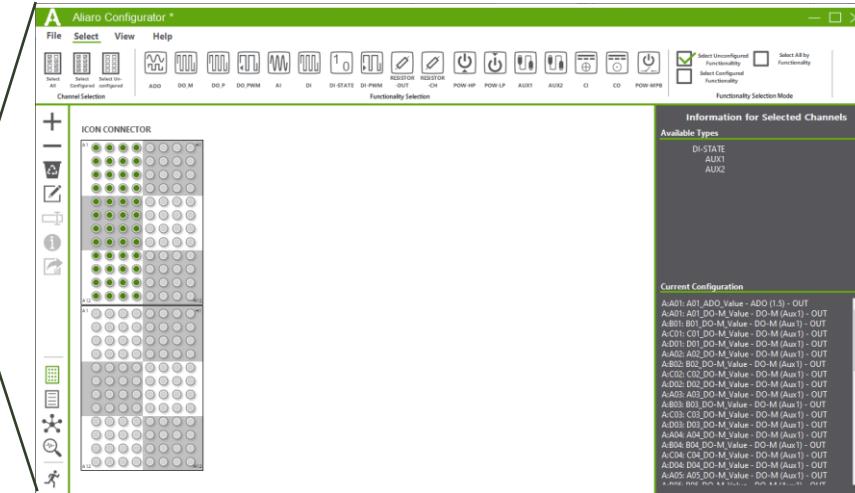
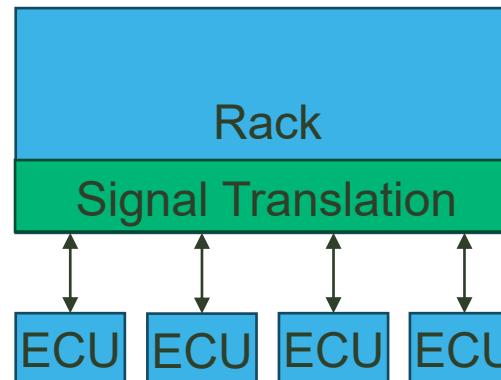


Reduce Commissioning (time to test)

Current State:



Future State:



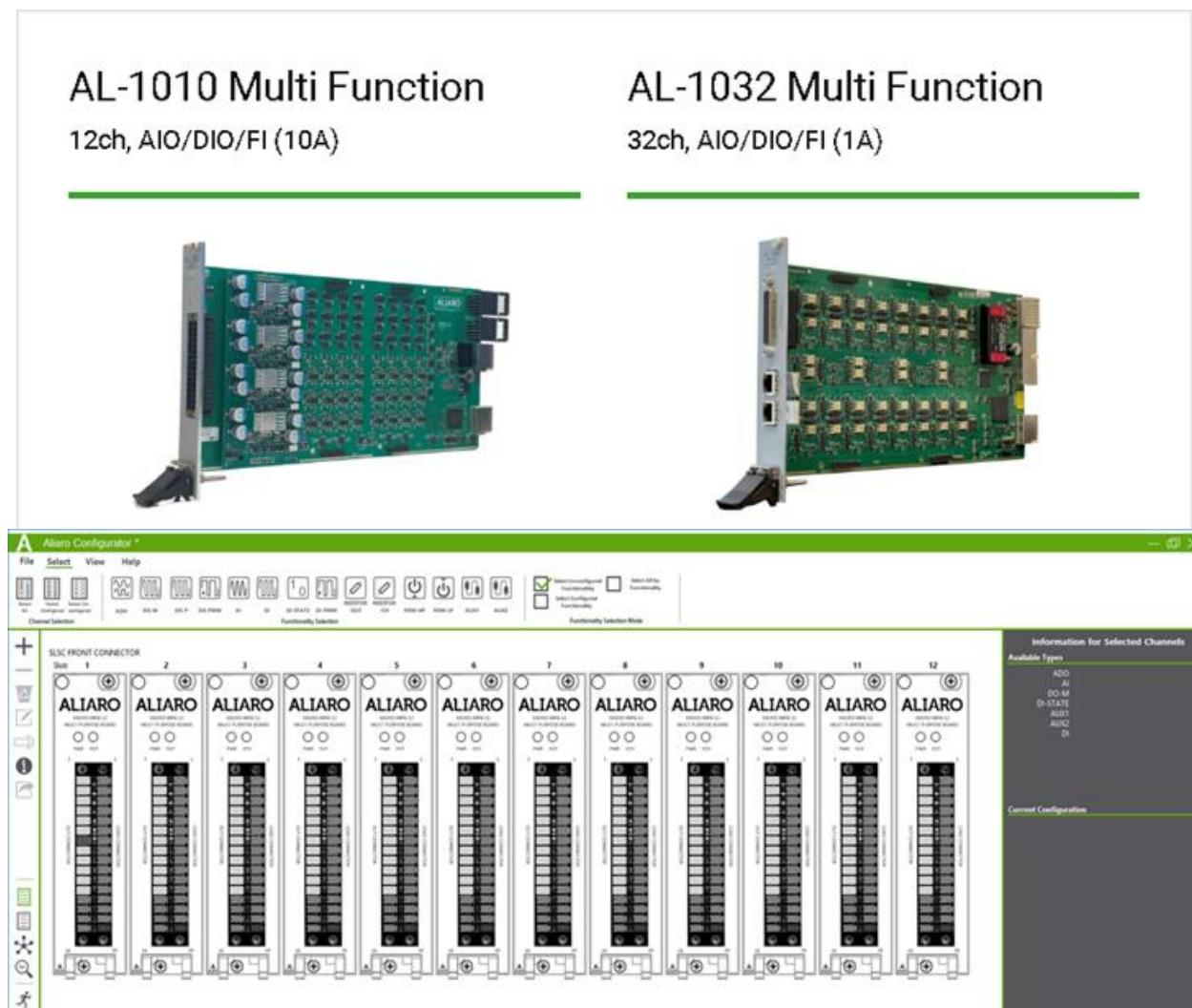
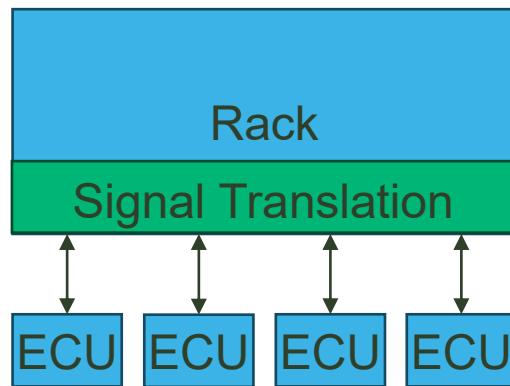
- Software defined wiring
- Reduce wiring 3x to 4x (and everything associated with it)
- Frees weeks in project plan
- Frees resources to write scripts

How: Aliaro Software Configurable Pins

Aliaro 1010 & 1032

- AIO, DIO, PWM, FIU **per pin**
- Software configurable

Connect, Configure, Next



Any Number of DUTs

- Goal: One Sequence → Any number of DUTs
- Hardware Design
- Design code for multiple callers
- TestStand Batch or Parallel Mode

Multi-up: Hardware Design

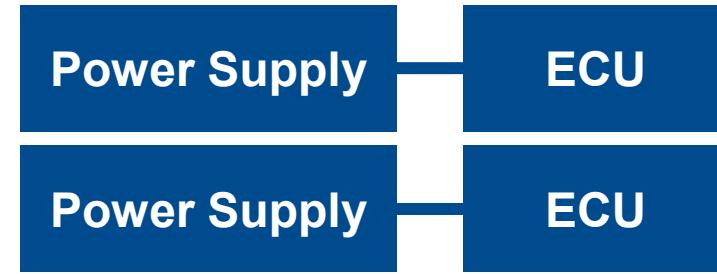
- Ideal: Grouped, Independent Hardware set per DUT
 - Will cost more in hardware
 - Will reduce software development costs
 - Will reduce test times; no shared resources

Example:

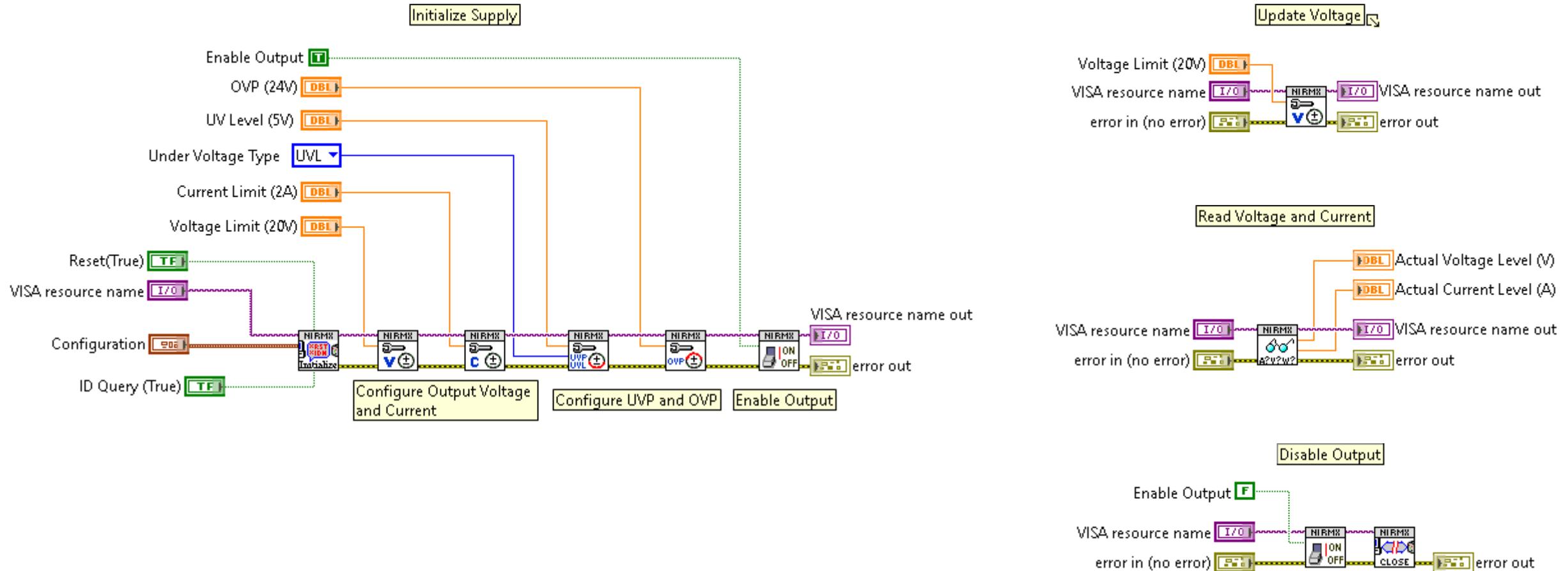
- Power supply per DUT
- Shared power supply for all DUTs

Power Supply Example: Independent (Ideal)

- Power supply per DUT (+ cost)
- Power supply provides per DUT:
 - Enable relay
 - Current Sense
 - Voltage Sense
 - Drop compensation



Power Supply Example: Independent (Ideal)



Power Supply Example: Shared (Not Ideal)

Shared power supply (- cost)

Bigger power supply (+ cost)

Relays (+ cost)

Relay Control (+ cost)

Individual current sensors (+ cost)

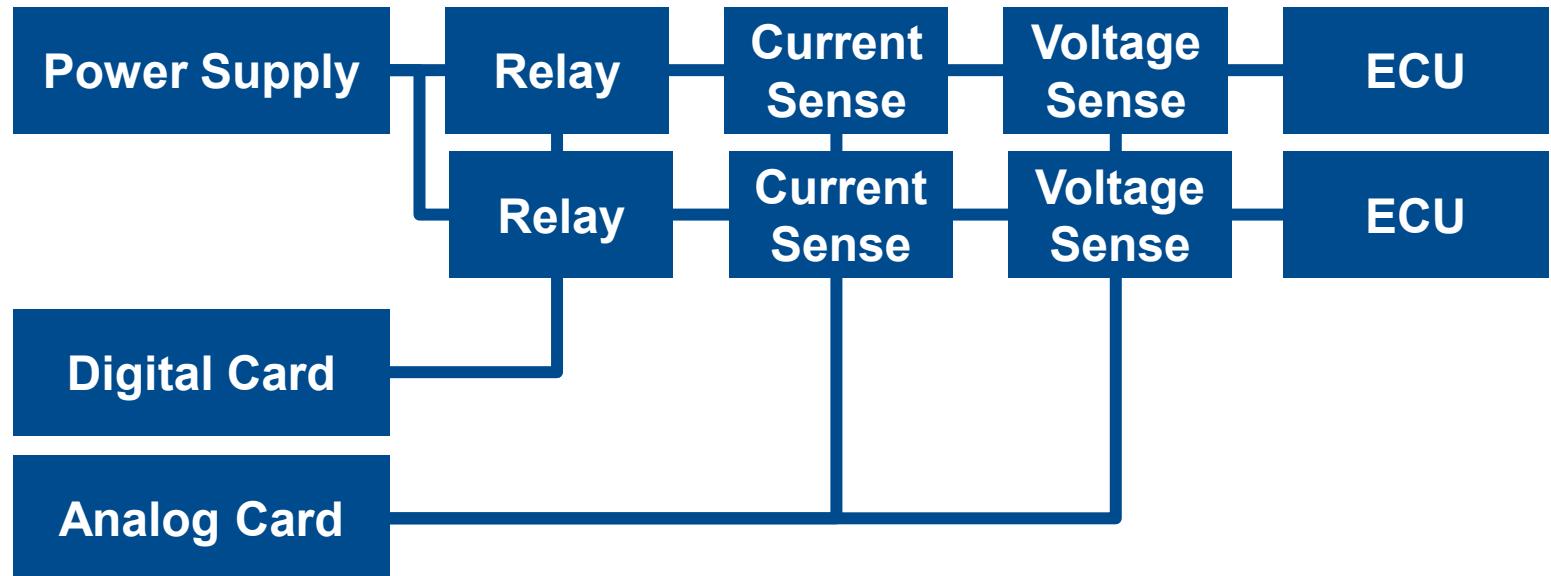
DAQ to measure current/voltage (+ cost)

Additional wiring (+ cost)

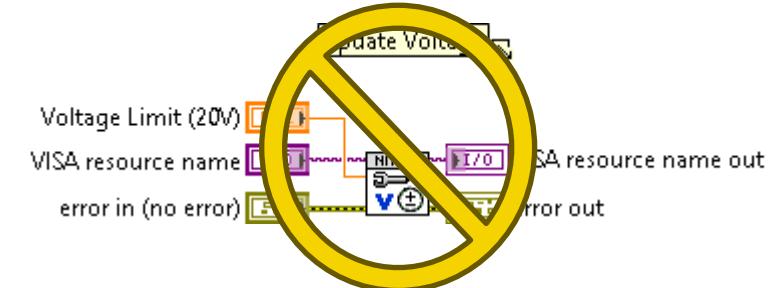
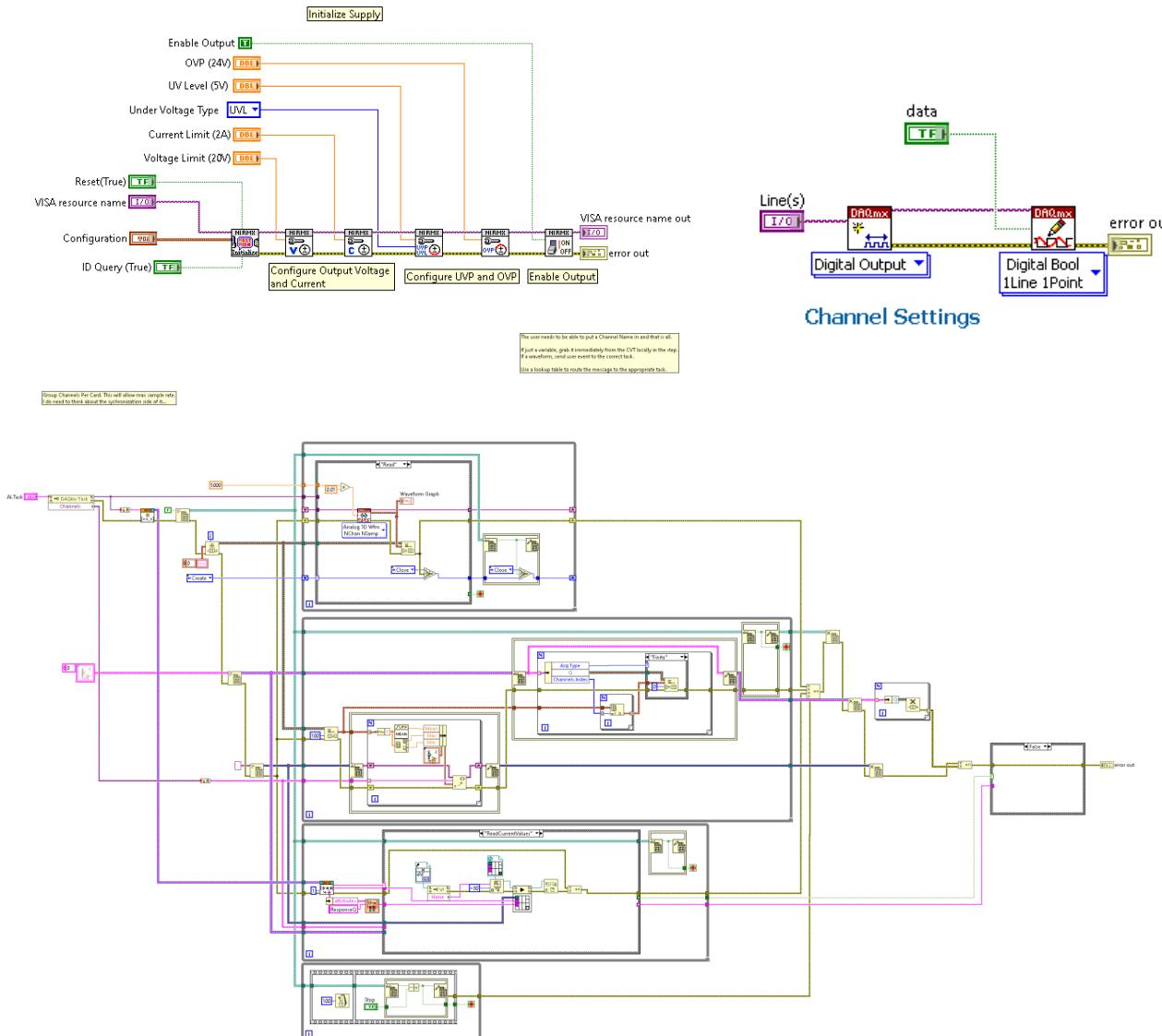
Requires significant software (+ cost)

Can't adjust voltage per DUT

No drop compensation

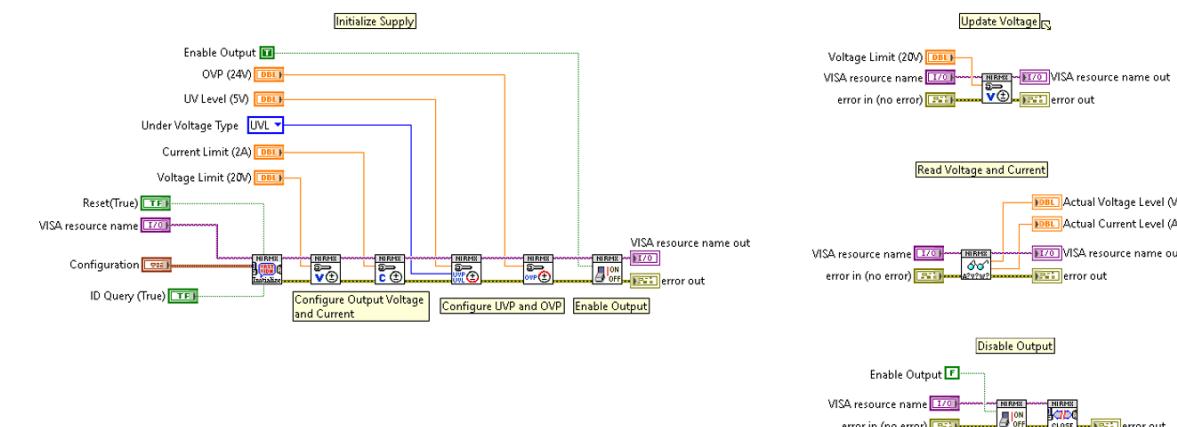
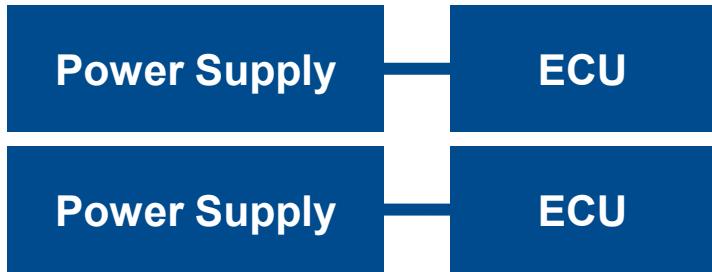


Power Supply Example: Shared (Not Ideal)

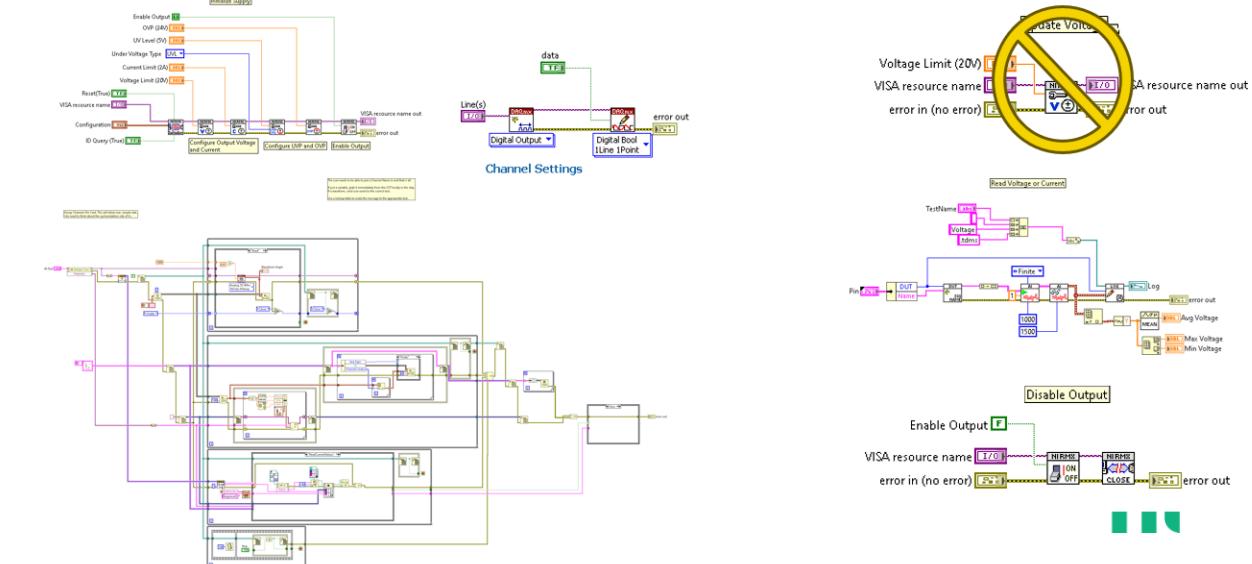
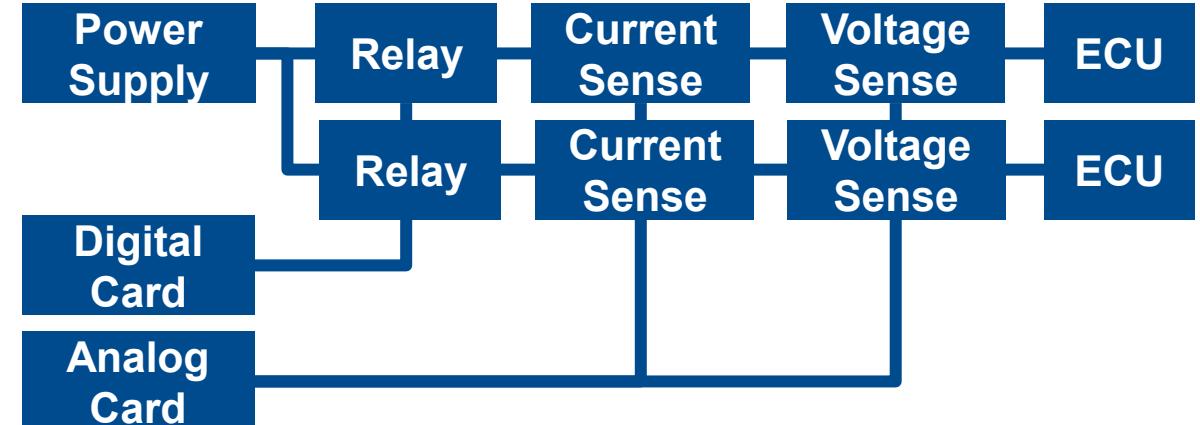


Compared: Cost – Up Front Vs Total System

Upfront: \$\$; System: \$\$\$

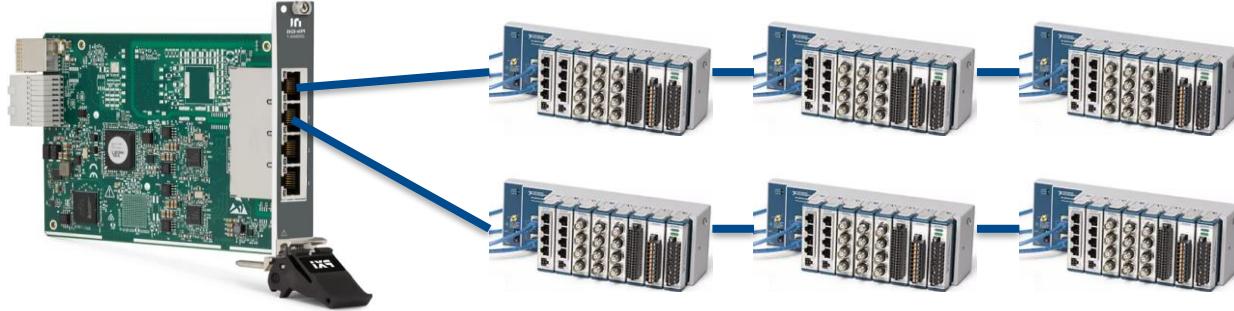


Upfront: \$; System:\$\$\$\$

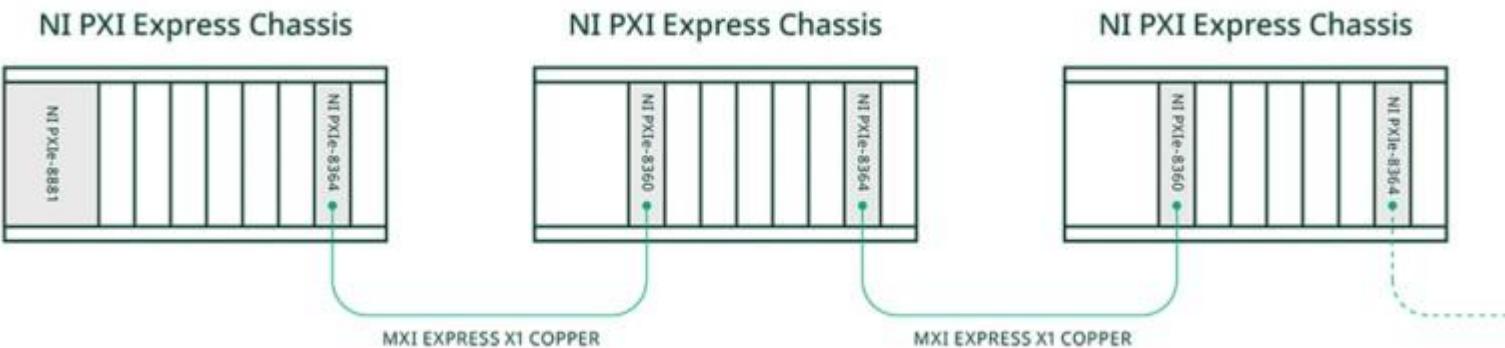


Multi-up: Design Hardware

- cDAQ Ethernet String per DUT

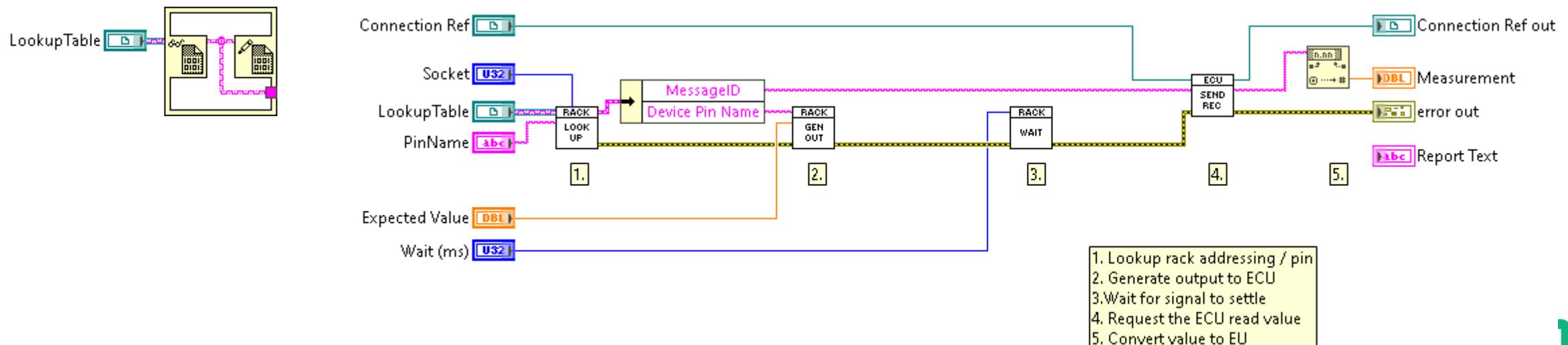
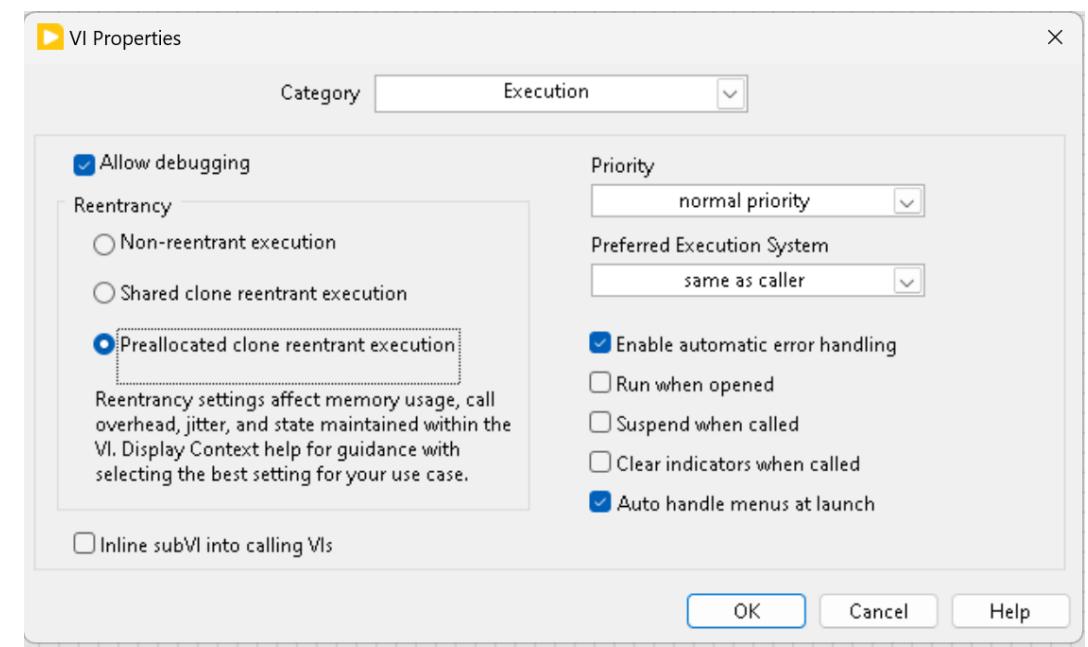


- PXI Chassis Expansion



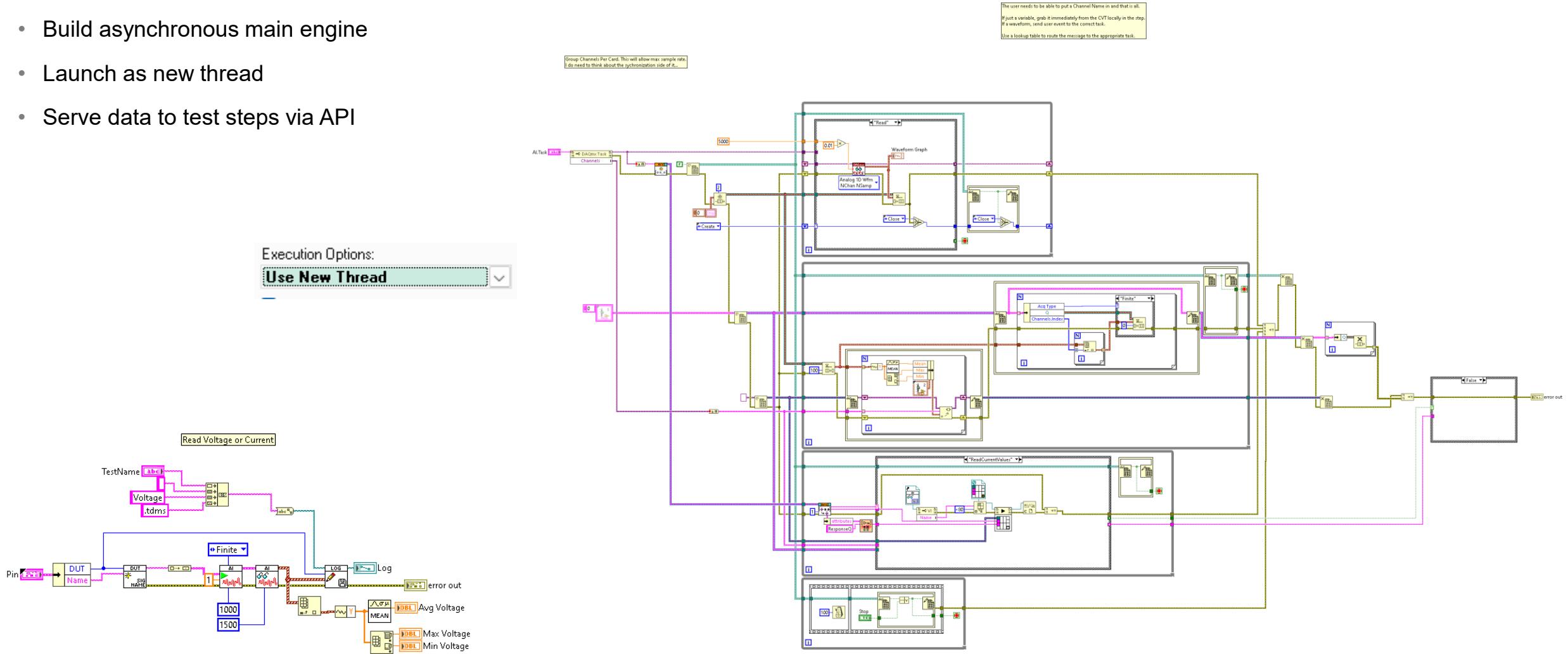
Multi-up: Software Considerations

- Re-entrant (single source file → each call is uniquely instantiated)
- Separate memory per DUT
 - Use DVRs or named queues, not (functional) globals
 - Pass unique references
- Abstract shared resources with async engine and APIs
- Parameterize functions with Socket ID



Multi-up: Shared Resources

- Build asynchronous main engine
 - Launch as new thread
 - Serve data to test steps via API

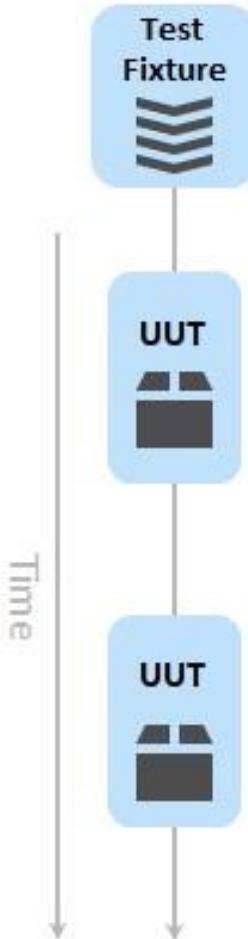


Multi-up: TestStand Process Model

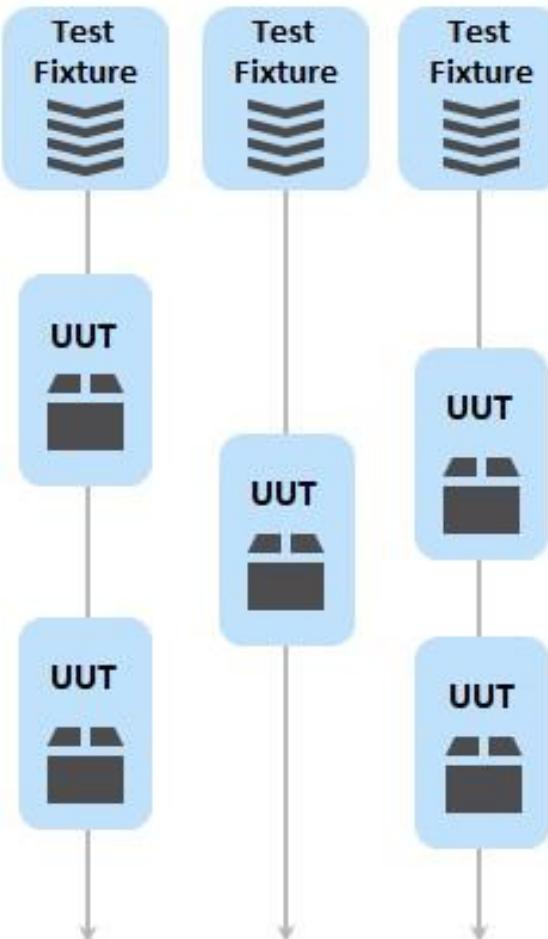
- Set Process Model: Sequential vs Batch vs Parallel Mode
- Set Number of Sockets
- Get Socket ID

TestStand Process Model Options

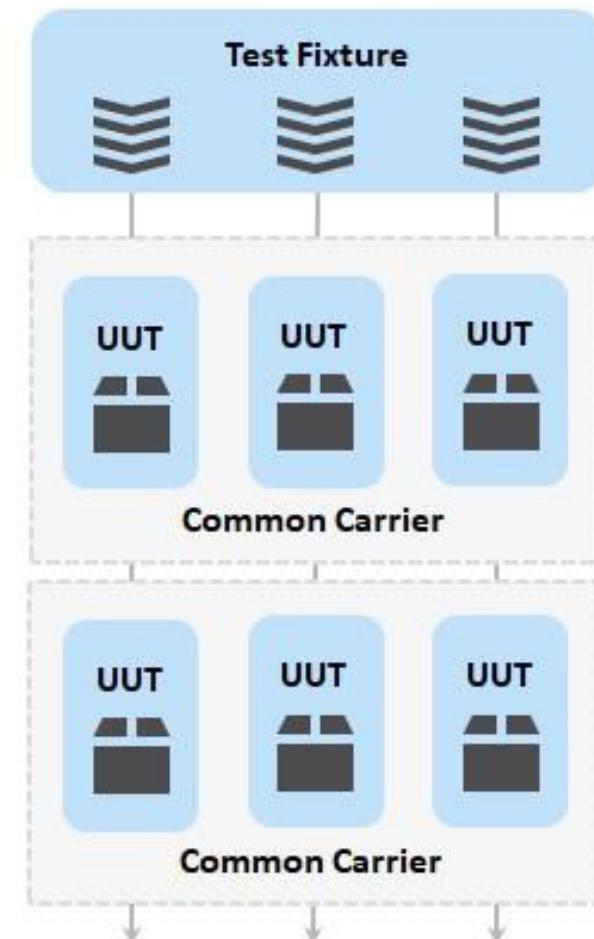
Sequential Model



Parallel Model



Batch Model



TestStand Process Model Options

Sequential

- One DUT and one fixture

Batch

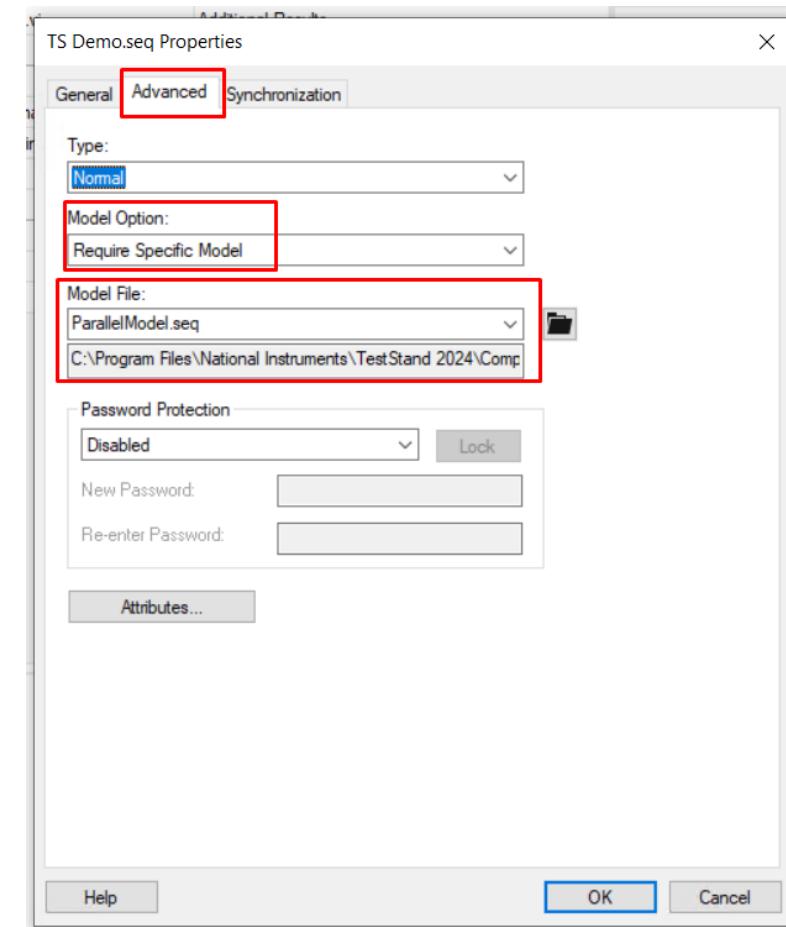
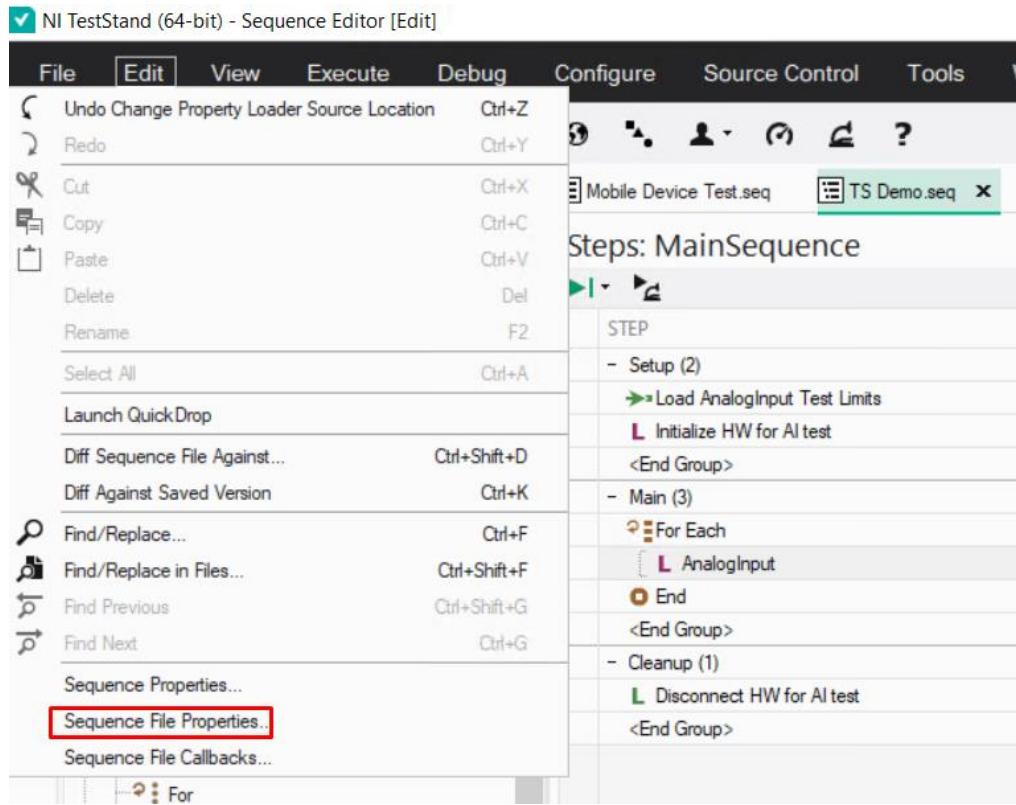
- Multiple DUTs and multiple fixtures
- Acts together
- DUT fails → Abort all
- Does not require hardware independence

Parallel

- Multiple DUTs and multiple fixtures
- Acts independently
- DUT fails → Abort failed DUT
- Requires hardware independence

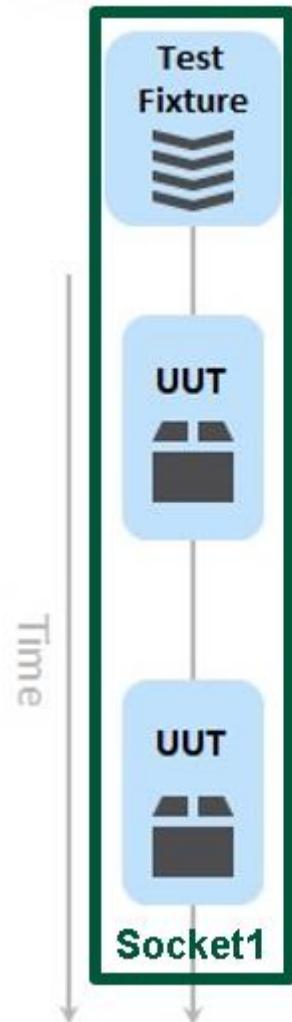
Specify a Process Model for TestStand Sequence File

- <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P7NrSAK&l=en-US>

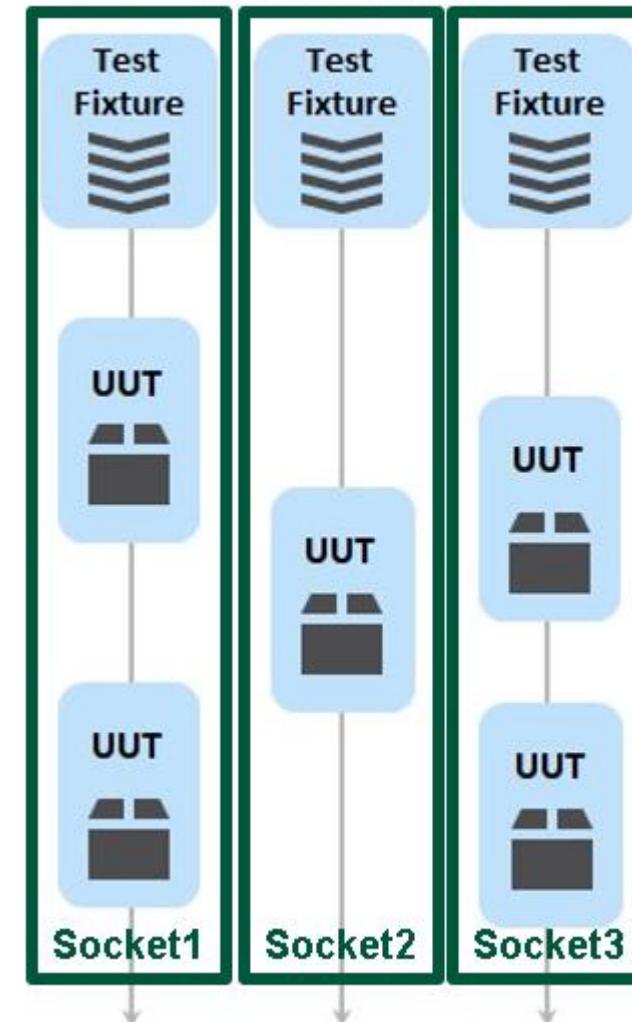


TestStand Sockets

Sequential Model



Parallel Model



Programmatically Set the Number of Test Sockets in TestStand

- <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P7p7SAC&l=en-US>

The screenshot shows the NI TestStand interface with two sequences open: "Mobile Device Test.seq" and "TS Demo.seq".

Mobile Device Test.seq:

- Steps: ModelOptions**: A sequence of steps under the "ModelOptions" category.
- Setup (9) Steps:**
 - "Enter Number of DUTs": Sets the variable `Locals.NumberSockets` to the user input.
 - "Select": Sets the variable `Locals.NumberSockets` to a value from the list {1,2,3,4,5,6}.
 - "Case": Sets the variable `Parameters.ModelOptions.NumTestSockets` to the selected value.
 - "End"
 - "Case": Sets the variable `Parameters.ModelOptions.NumTestSockets` to 1 if no valid entry was made.
 - "End"
 - "End"
 - "End Group"
- Main (1) Step:** Sets the variable `Parameters.ModelOptions.NumTestSockets` to 1.
- Cleanup (0) Steps:** None.

TS Demo.seq:

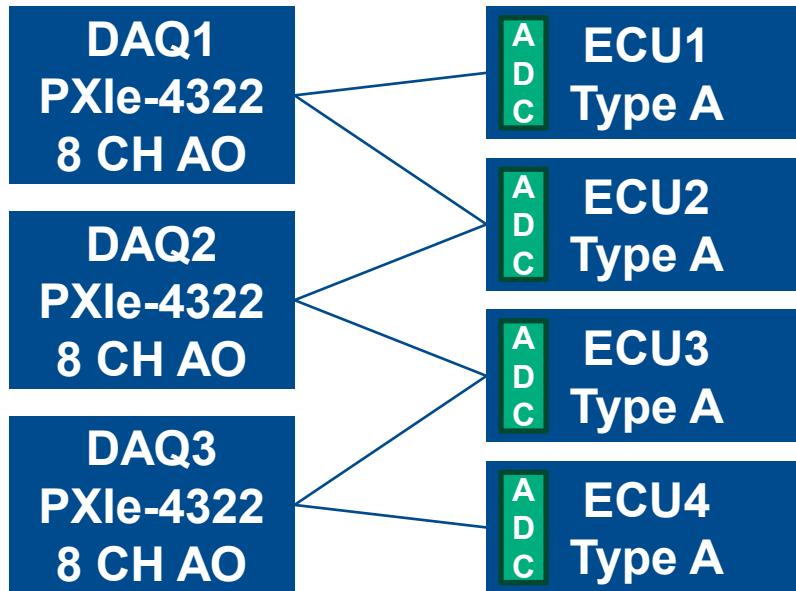
- Sequences Tab:** Shows the "Sequences" table with two entries:
 - MainSequence**: Commented as "GetModelOptions c..."
 - ModelOptions**: Commented as "GetModelOptions c..."
- Step Settings for Set the number of test sockets:**
 - Expression Tab:** Shows the expression: `Parameters.ModelOptions.NumTestSockets = 1`.

Programmatically Get Socket ID

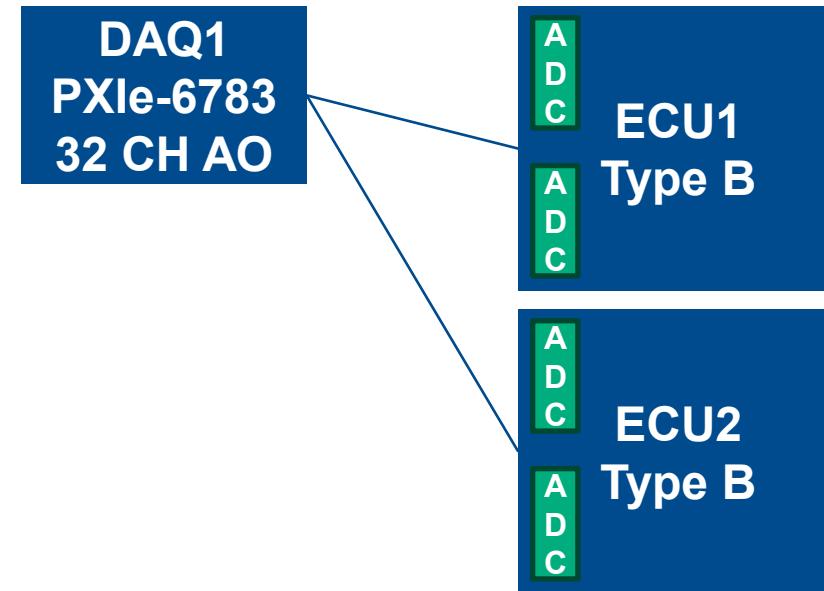
- Use TestStand Variable: RunState.TestSockets.MyIndex

PARAMETER NAME	TYPE	IN/OUT	LOG	DEFAULT	VALUE
Socket	Number (U32)	in	<input type="checkbox"/>	<input type="checkbox"/>	RunState.TestSockets.MyIndex

Demo: Putting it all together

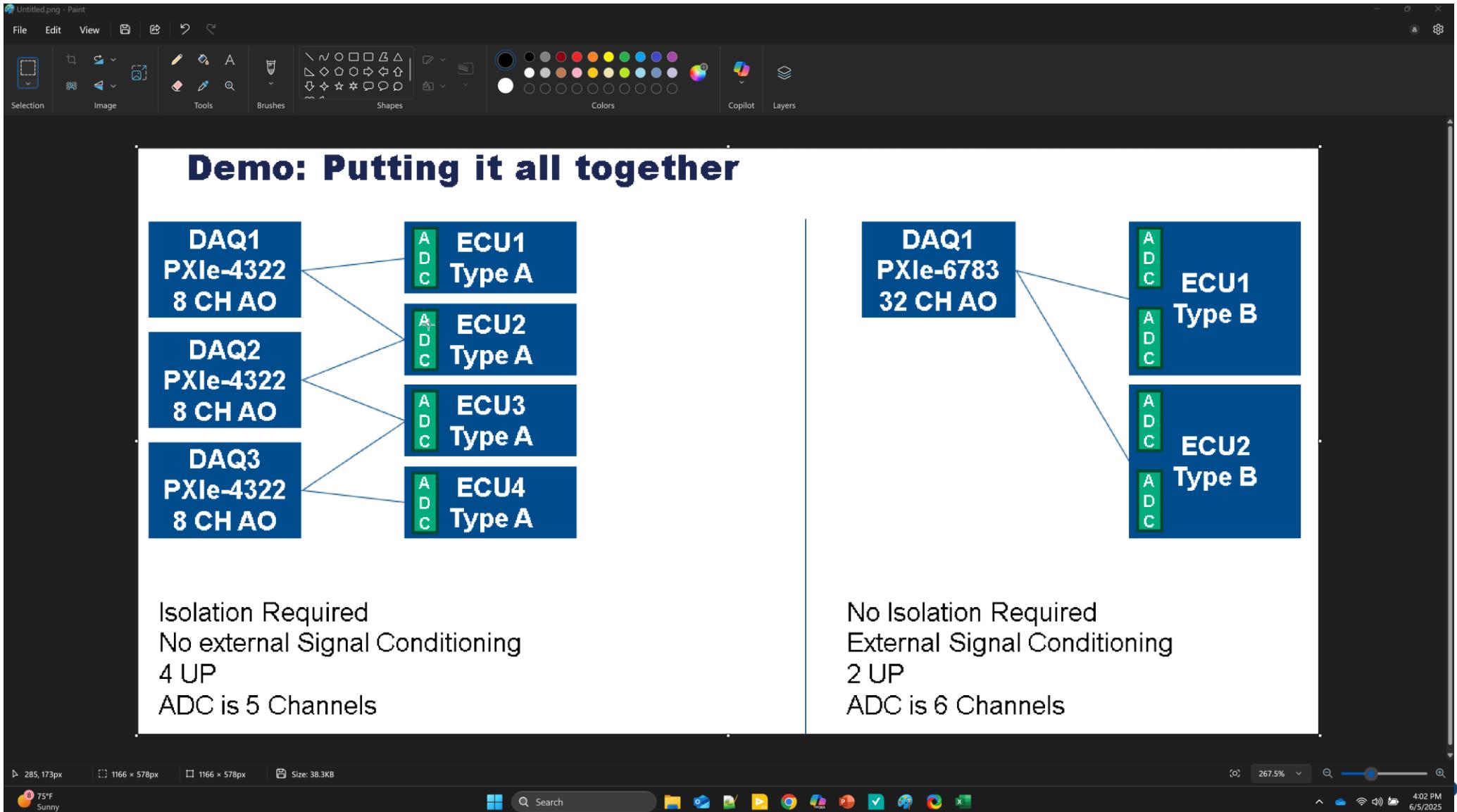


Isolation Required
No external Signal Conditioning
4 UP
ADC is 5 Channels



No Isolation Required
External Signal Conditioning
2 UP
ADC is 6 Channels

Setup: Putting it all together



Demo: ECU A – Putting it together

The screenshot shows the NI TestStand Sequence Editor interface. The main window displays the 'ECU A LTT Sequence.seq' sequence. The 'Steps: MainSequence' table lists the following steps:

STEP	DESCRIPTION	SETTINGS
- Setup [3]	Load lookup table for selected ECU Initialize Lookup Table	Action, Shared Steps.lvproj, CreateMap.vi
- If	Cache pin settings for traceability	Action, Shared Steps.lvproj, List Info by Socket.vi
- Else If	Update limits path based on ECU Type	Additional Results Locals.LimitsFile = "C:\Projects\TestStand Reuse\" + Locals.ECType + ...
- For Each	Set Limits File	
- For	<End Group>	
- Main [1]	Execute ADC Test	
- Check ADC	Call MainSequence in ECU ADC Test.seq	
- <End Group>		
+ Cleanup [0]		

The 'Variables' pane shows the following variable definitions:

NAME	VALUE	TYPE	COMMENT
Locals (MainSequence)			
t23.LookupTableRef	0	Number	
Locals	...	String	
ECUType	"ECU_A"	String	
ResultList		Array of Result[0..empty]	
Parameters (MainSequence)			
FileGlobal (ECU A LTT Sequence se...			

A file explorer window titled 'ECU_A' is open, showing the contents of the folder. It contains two items: 'Config Files' (a folder) and 'ECU A LTT Sequence.seq' (a TestStand Sequence file). The sequence file was created on 6/5/2025 at 4:48 PM.

The bottom status bar shows: User: administrator, Environment: <Global>, Model: BatchModel.seq, No Steps Selected, Number of Steps: 4.

Demo: ECU B – Putting it together

NI TestStand (64-bit) - Sequence Editor [Edit]

File Edit View Execute Debug Configure Source Control Tools Window Help

Insertion Palette: ECU B LTT Sequence.seq

Steps: MainSequence

STEP	DESCRIPTION	SETTINGS
- Setup [4]		
Load lookup table for selected ECU	Action, Shared Steps.lvproj.CreateMap.vi	
Initialize Lookup Table		
Cache pin settings for traceability		
List Socket Properties	Action, Shared Steps.lvproj.List Info by Socket.vi	Additional Results
Update limits path based on ECU Type		
Set ADC 1 Limits File		
Update limits path based on ECU Type	Locals.ADC1_LimitsFile = "C:\\Projects\\TestStand Reuse\\" + Locals.EC...	
Set ADC 2 Limits File		
<End Group>	Locals.ADC2_LimitsFile = "C:\\Projects\\TestStand Reuse\\" + Locals.EC...	
- Main [2]		
Execute ADC Test on ADC 1		
Check ADC 1	Call MainSequence in ECU ADC Test.seq	
Execute same ADC Test on ADC 2		
Check ADC 2	Call MainSequence in ECU ADC Test.seq	
<End Group>		
+ Cleanup [0]		

Variables

NAME	VALUE	TYPE	COMMENT
Locals ('MainSequence')			
123_LookupTableRef	0	Number	
ADC1_LimitsFile	...	String	
ADC2_LimitsFile	...	String	
ECUType	"ECU_B"	String	
ResultList		Array of Result[0..empty]	<Right click to insert Local>
Parameters ('MainSequence')			<Right click to insert Parameter>
FileGlobals ('ECU B LTT Sequence.seq...')			<Right click to insert File Global>
StationGlobals			
ThisContext			Sequence Context
RunState			
Step ('Initialize Lookup Table')		Action (Step)	Load lookup ta...

Step Settings for Initialize Lookup Table

Module Properties

Call Type: VI Call

Project Path: ..\SharedSteps\Shared Steps.lvproj

(Optional) C:\Projects\TestStand Reuse\Shared\Steps\Shared Steps.lvproj

VI Path: My Computer\Shared Steps.lvlib\Shared\CreateMap.vi

C:\Projects\TestStand Reuse\Shared\Steps\CreateMap.vi

PARAMETER NAME	TYPE	IN/OUT	LOG	DEFAULT	VALUE
Mapping file path	Path	in	<input type="checkbox"/>	<input type="checkbox"/>	"C:\\Projects\\TestStand Reuse\\\" + Locals.ECUType + \"\\Config Files\\\\Pin_Mapping.csv"
LookupTable	Number (U32)	out	<input type="checkbox"/>	<input type="checkbox"/>	Locals.LookupTableRef
error out	Container	out	<input type="checkbox"/>	<input type="checkbox"/>	Step.Result.Error

Shared Steps.lvproj
Shared Steps.lvlib:CreateMap.vi

Mapping file path [] → [] LookupTable
[] error out

Load mapping file (CSV), create map, and create DVR reference to the map.

User: administrator Environment: <Global> Model: BatchModel.seq 1 Step Selected [0] Number of Steps: 6

Summary: Putting it all together

- Used DAQmx for HAL
- Used file mapping for pin abstraction & command IDs per ECU Type
- Config file for limits per ECU type
- ECU Type sets number of sockets
- Executing in Batch mode
- Single Sequence for ADC test
 - Worked for both ECUs
 - Worked for one or any number of ADCs
 - ADC test step renamed each iteration to specific pin under test

North Star: Save That Money...

- By maximizing test steps / sequences re-use for:
 - Any Rack → Abstraction
 - Any DUT → Hardware and Test Software Design
 - Any number of DUTs → Test Sequencer Design

Two Paths to Success

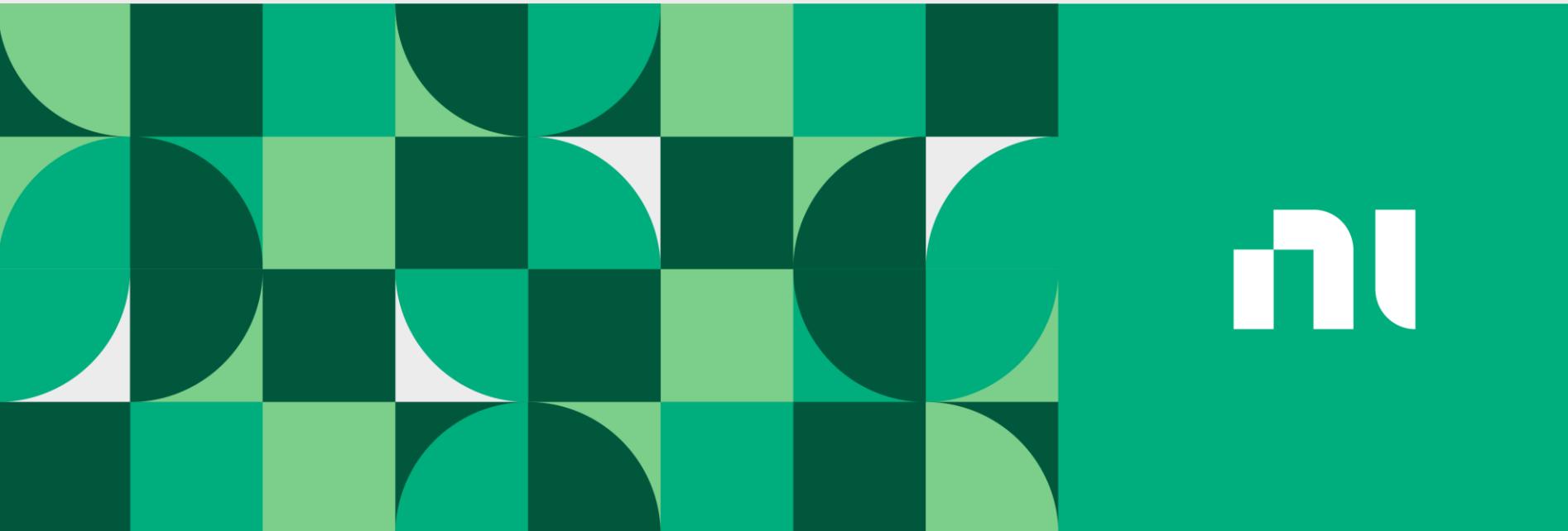
DIY

- This is always a conversation
- Include us...We can help

Want to outsource?

- This is always a conversation
- Include us...We can help
- We can point you to partners who can help more

Thank you!!



NI is now part of Emerson.