Standardized NPSS Propulsion Model integration into Simulink process using FMI

MATLAB EXPO 2023

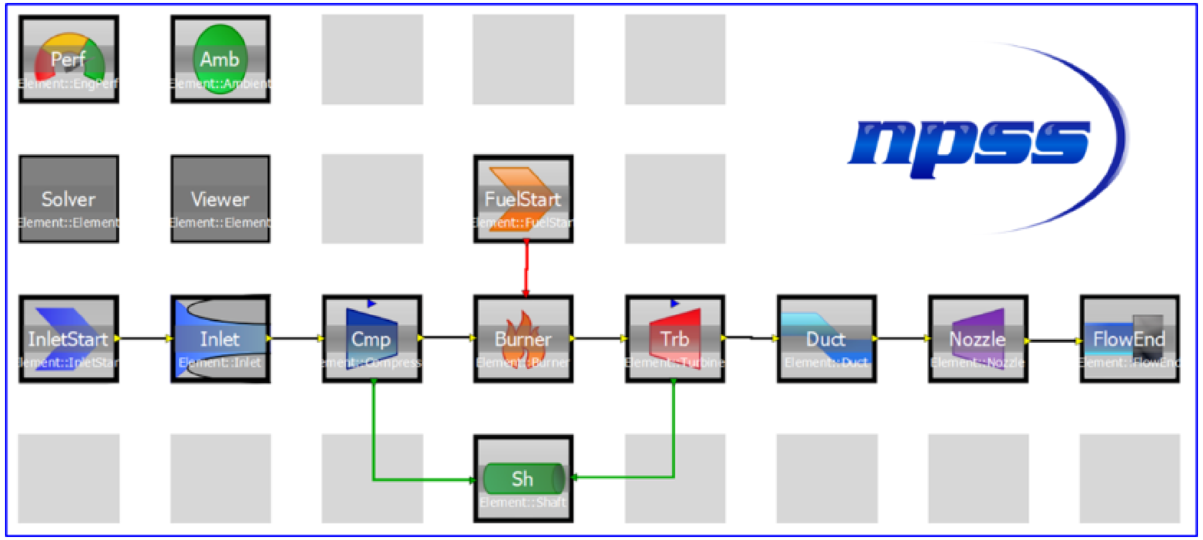
LOCKHEED MARTIN

Keith Pattison, Associate Fellow Propulsion Modeling and Simulation
Lockheed Martin Aeronautics – Fort Worth Tx

Public Information Release Authorization (PIRA) AER2023020038

Lockheed Martin Aeronautics:
Supporting Legacy Aircraft and ADP: Advanced Development Programs

Introduction

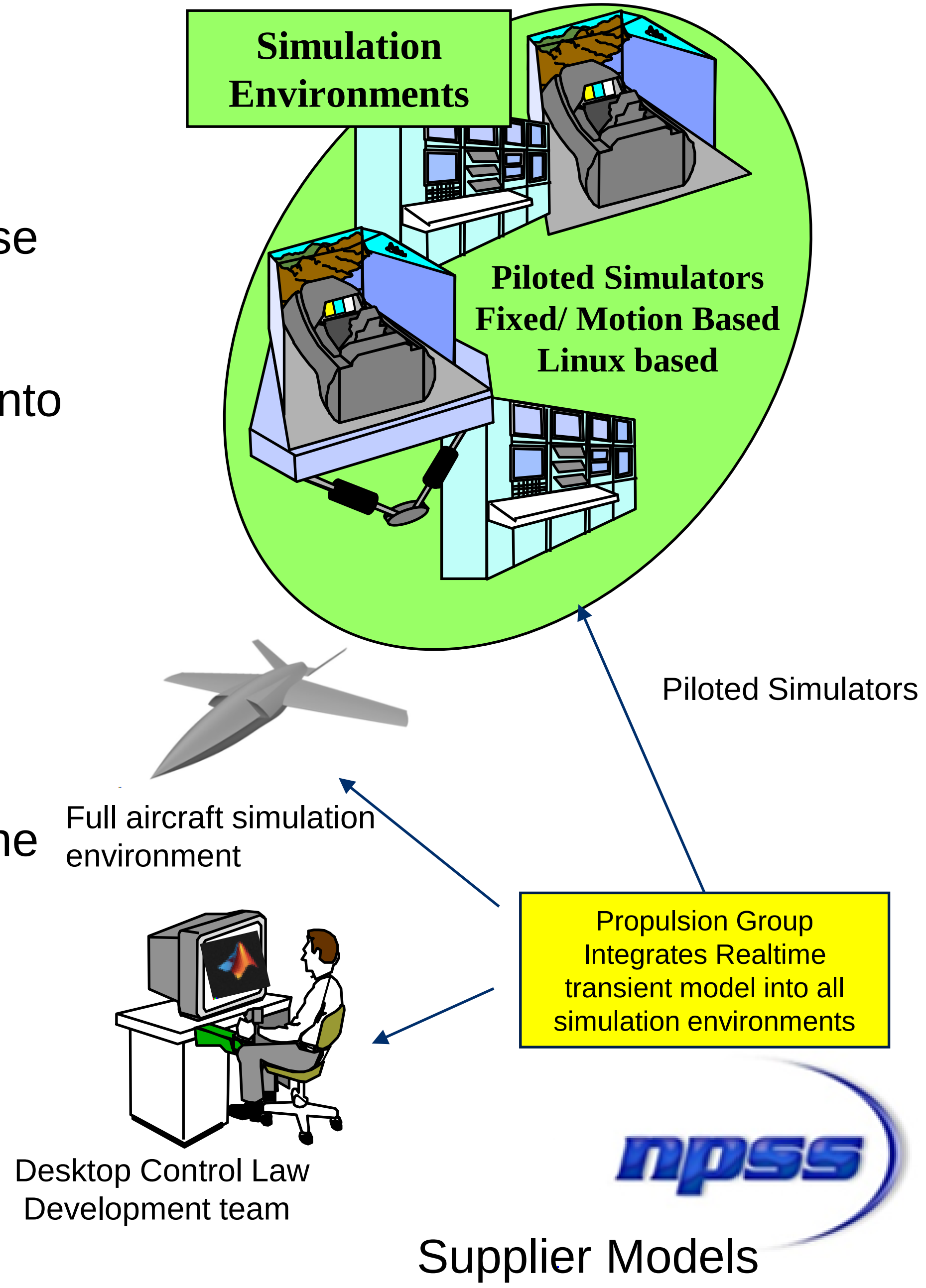


- Integrated Flight and Propulsion Controls has become much more important as aircraft systems become more complex.
- NPSS is a NASA/Industry developed tool that provides an object orientated simulation environment for propulsion performance modeling.
- Utilization of High-Fidelity plant and engine control models, from our suppliers, packaged in NPSS environment is standard practice on almost all Lockheed Martin programs.
- NPSS Engine models are full physics plant model of the engine system, coupled with a representative Engine Control model FADEC(Full Authority Digital Engine Control) to provide installed engine flight performance through out the flight envelope.
- Engine Simulations are utilized at all stages program development from Conceptual design to end of life. They are frequently updated as the program matures, or the control logic is updated.

Engine Models are Constantly being utilized and updated

Background

- Legacy Engine model integration used to be done on a case by case basis with custom interface code developed for each program.
- Usually needed custom interfacing software (Middleware) to hook into Commercial tools like MATLAB/Simulink
- Over the last 10 years utilized Simulink with legacy interface with numerous issues. Required a number of tools to be aligned, NPSS version, Compilers, MATLAB versions. Many stability issues in the interface resulting in numerous hours of trouble shooting.
- FMI / FMU approach removes the middleware layer support from the model interface and places it on the host tool to accommodate.



FMI / FMU approach saves integration time and provides more system stability and common interface to many tools

NPSS FMU Export Development & implementation challenges

- In 2022 Lockheed Martin worked with the NPSS Consortium development team to test and refine the FMI version 2 FMU export capability. Lockheed Martin also worked with MathWorks SMEs to evaluate the Simulink interface to the new NPSS FMUs to develop several features of the NPSS FMUs

- NPSS will have FMI 2 (CS & ME) import/export capabilities in upcoming version 3.3 being released later this year.
  - Released as EMI version 15.0 as part of the 3.3 commercial release
  - Most features shown in this presentation available.
- For NPSS model Export Currently using a Python3 script to package an NPSS model into an FMU, it uses standard python packages with no dependencies. Desire was to have a simple wrapping process without any other tool or package dependencies.

- Take advantage of FMU Multi OS Capability with Windows/Linux

- Eliminate Software compilation as part of the wrapping process.

- Build Interface to be real time fast, no I/O slowdowns.

FMUs will feed real time simulation environments and need to be as fast as possible.

NPSS FMU Capability implementation

Sim Host Calling FMU

- NPSS FMU packaging script maps variables to 2 files. This is setup in a .config file as shown below:

Handles Naming to the Calling Host per FMI standard

```
# Fields  FMI name, NPSS name (optional), Data type, causality, variability , units, Description
#  These are the inputs
variable: time,,real, independent,  continuous, sec,   Simulation time
variable: SimMode,,real,input,continuous,none,Steady State = 1 Transient = 3
variable: DataRec,,real,input,continuous,none, 0=off 1=on
#  AtmoIn Input Bus
variable: AtmoIn%Amb.alt_in,,real,input,continuous,ft,Pressure Altitude
variable: AtmoIn%Amb.MN_in,,real,input,continuous,none,Mach Number
variable: AtmoIn%Amb.Ps_in,,real,input,continuous,psia,Ambient pressure input
variable: AtmoIn%Amb.Ts_in,,real,input,continuous,R,Ambient temperature
variable: AtmoIn%Amb.switchDay.intValue,,real,input,continuous,none,AS210 Day Type 0:Std 1:Trop -1

variable: EI.PLA_in,,real,input,continuous,deg,Power 10=GI 15=FI 100=Mil 150=MaxAB
variable: WBI_in,,real,input,continuous,lbm/sec,Mid Compressor Bleed input
variable: WBD_in,,real,input,continuous,lbm/sec,Comp Discharge Bleed input
variable: HPX_in,,real,input,continuous,hp,HP Extraction input

# Basic Global outputs
variable: FG,,real,output,continuous,lbf,Gross Thrust
variable: TMDD,,real,output,continuous,lbf,Net Propulsion Force
variable: WFT,,real,output,continuous,lbm/sec,Fuel Flow pound mass per sec
variable: PLA_FDBK,,real,output,continuous,deg,PLA FeedBack
variable: CbldECS.W,,real,output,continuous,lbm/sec,Bleed Flow Output
variable: CbldECS.Pt,,real,output,continuous,lbf,Bleed Pressure
variable: CbldECS.Tt,,real,output,continuous,R,Bleed Temperature
variable: ShH.HPX,,real,output,continuous,hp,High Compressor Shaft Extraction
```

ModelDescription.XML

```
<ModelVariables>
    <ScalarVariable name="time" valueReference="1" description="Simulation time" variability="continuous" causality="independent">
        <Real unit="sec"/>
    </ScalarVariable>
    <ScalarVariable name="SimMode" valueReference="2" description="Steady State = 1 Transient = 3" variability="continuous" causality="inp
        <Real unit="none" start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="DataRec" valueReference="3" description="0=off 1=on" variability="continuous" causality="input">
        <Real unit="none" start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="AtmoIn.Amb_alt_in" valueReference="4" description="Pressure Altitude" variability="continuous" causality="input"
        <Real unit="ft" start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="AtmoIn.Amb_MN_in" valueReference="5" description="Mach Number" variability="continuous" causality="input">
        <Real unit="none" start="0.0"/>
    </ScalarVariable>
```

npss_fmi.json

Handles Naming to the NPSS model thru parallel I/O arrays

```
"input": {
    "index": [
        2,
        3,
        4,
        5,
        6,
        7,
        8,
        9,
        10,
        11,
        12
    ],
    "name": [
        "SimMode",
        "DataRec",
        "Amb.alt_in",
        "Amb.MN_in",
        "Amb.Ps_in",
        "Amb.Ts_in",
        "Amb.switchDay.intValue",
        "EI.PLA_in",
        "WBI_in",
        "WBD_in",
        "HPX_in"
    ],
```

Using 4868 Datalist I/O to NPSS model for Speed

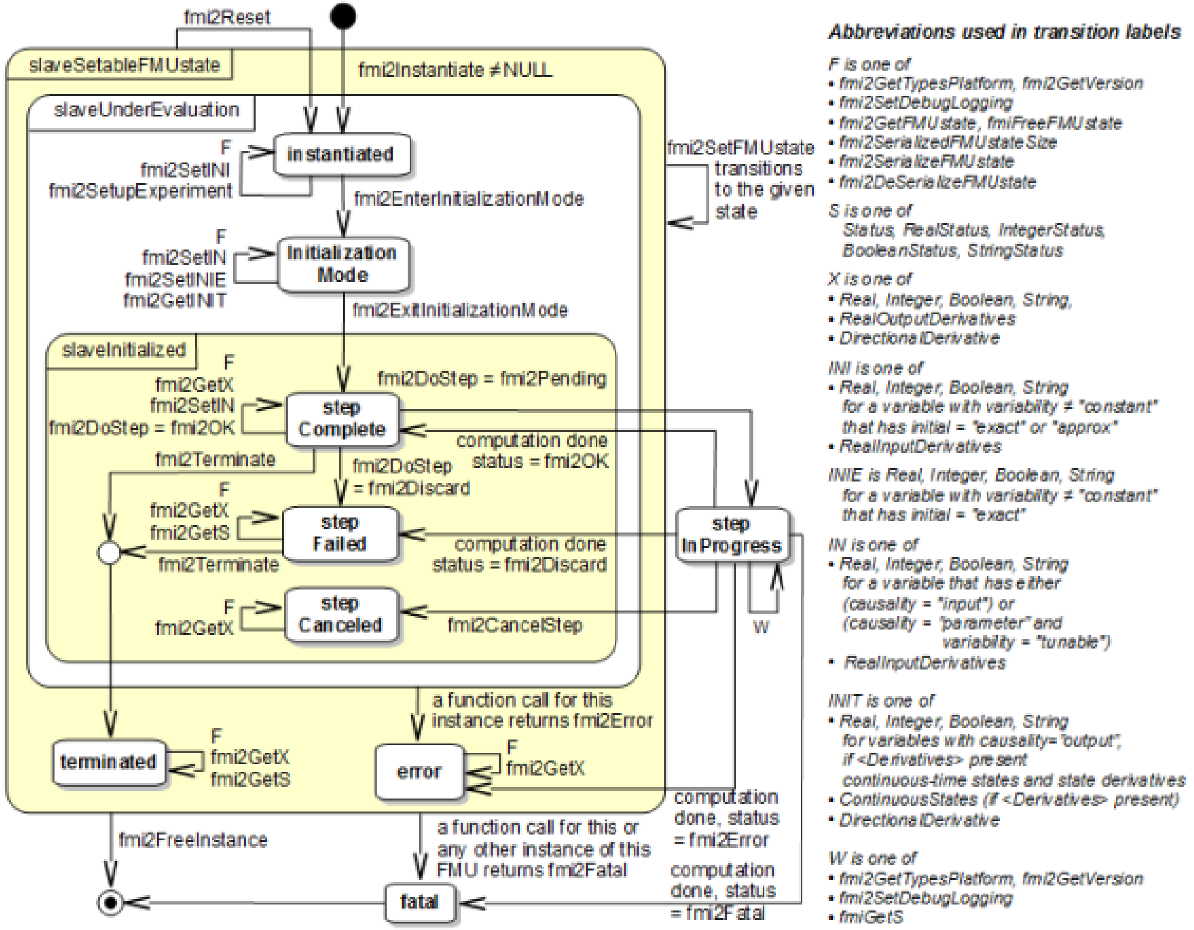
ARP 4868 C API

Variable naming can be adapted to meet requirements of both the FMI standard and NPSS

LOCKHEED MARTIN

NPSS FMU Capability implementation

- Initialization and loading of the NPSS model needed to be scheduled and accommodated several ways due to the way various host programs set initial values and run at time 0.
- NPSS model needs to be loaded (c4868init) in initialization mode prior to setReal() being called.
- Prefer setting input start values at runtime, not using FMU variable StartValue input.
- FMU logging and ARP 4868 Debug dump switchable on/off in FMU config file. Provides debugging capabilities for the FMU execution, but slows down runtime.



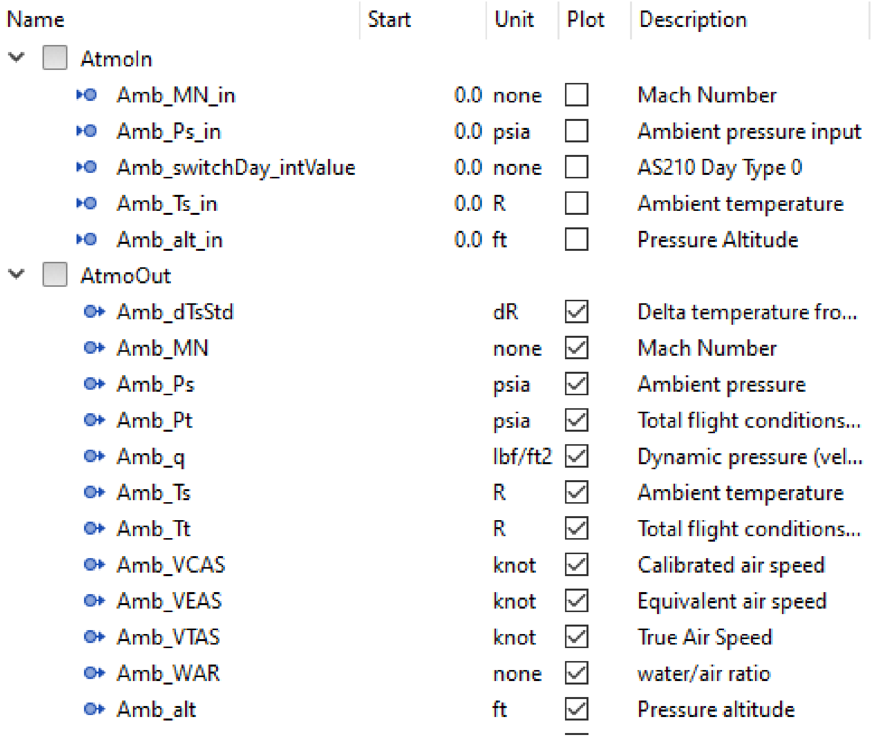
FMU Execution process aligned to allow proper NPSS model execution and initialization.

LOCKHEED MARTIN

NPSS FMU Capability implementation

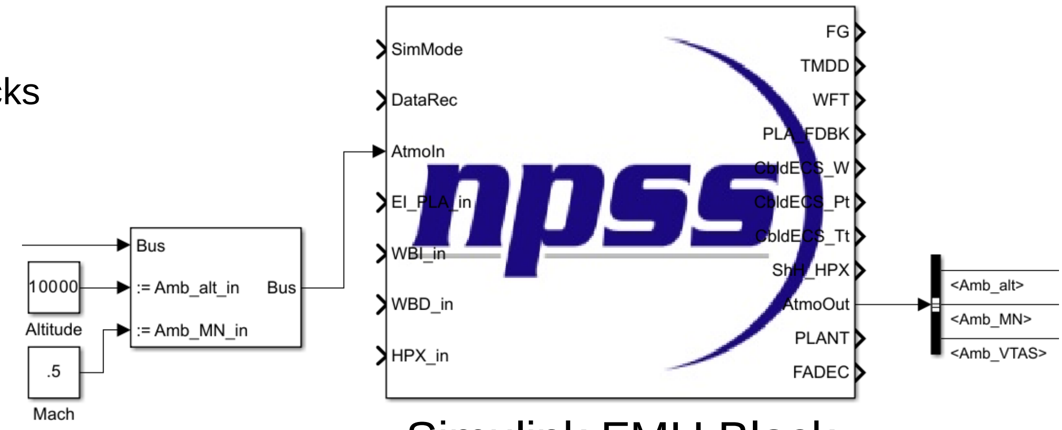
- Variable naming and issues with NPSS and FMI were first issue
  - NPSS is object oriented names can have C++ style object scope: Amb.alt_in , EI.PLA_in, TrbH.S_map.effDes
  - NPSS objects Contain both Inputs / Output variables
- Various host programs treat the . Differently for FMU I/O variables
  - FMPy treats . Blocks as an array
  - Simulink treats as a Bus Object with the dot separating multi level bus signals
  - Bus feature is good however need to map Inputs to one bus and outputs to another so NPSS for example:
    - Amb.alt_in (Input) Amb.alt(Output) need to be in separate Bus blocks

A Common approach was developed to handle issue in a number of tools allowing the Simulink Bus object to be utilized for large I/O blocks



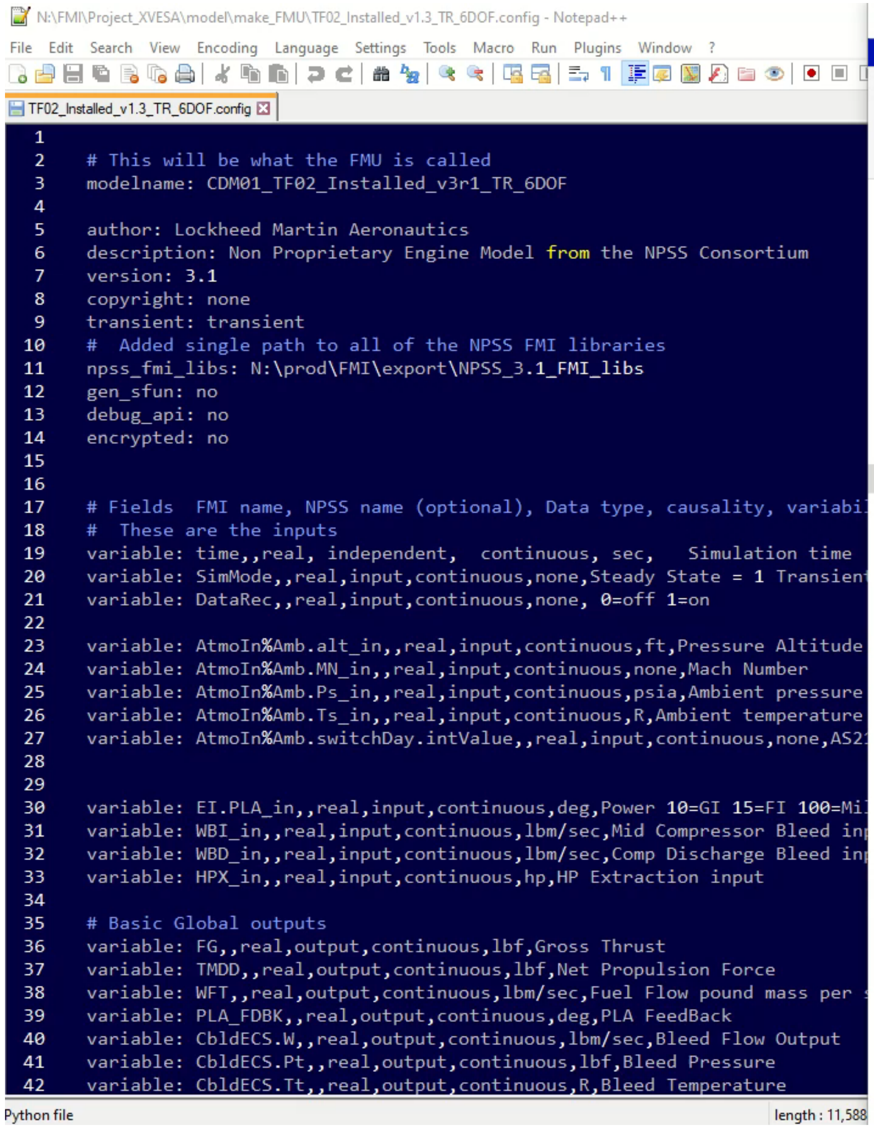
Name	Start	Unit	Plot	Description
AtmoIn				
Amb_MN_in	0.0	none	☐	Mach Number
Amb_Ps_in	0.0	psia	☐	Ambient pressure input
Amb_switchDay_intValue	0.0	none	☐	AS210 Day Type 0
Amb_Ts_in	0.0	R	☐	Ambient temperature
Amb_alt_in	0.0	ft	☐	Pressure Altitude
AtmoOut				
Amb_dTsStd		dR	☑	Delta temperature fro...
Amb_MN		none	☑	Mach Number
Amb_Ps		psia	☑	Ambient pressure
Amb_Pt		psia	☑	Total flight conditions...
Amb_q		lbf/ft2	☑	Dynamic pressure (vel...
Amb_Ts		R	☑	Ambient temperature
Amb_Tt		R	☑	Total flight conditions...
Amb_VCAS		knot	☑	Calibrated air speed
Amb_VEAS		knot	☑	Equivalent air speed
Amb_VTAS		knot	☑	True Air Speed
Amb_WAR		none	☑	water/air ratio
Amb_alt		ft	☑	Pressure altitude

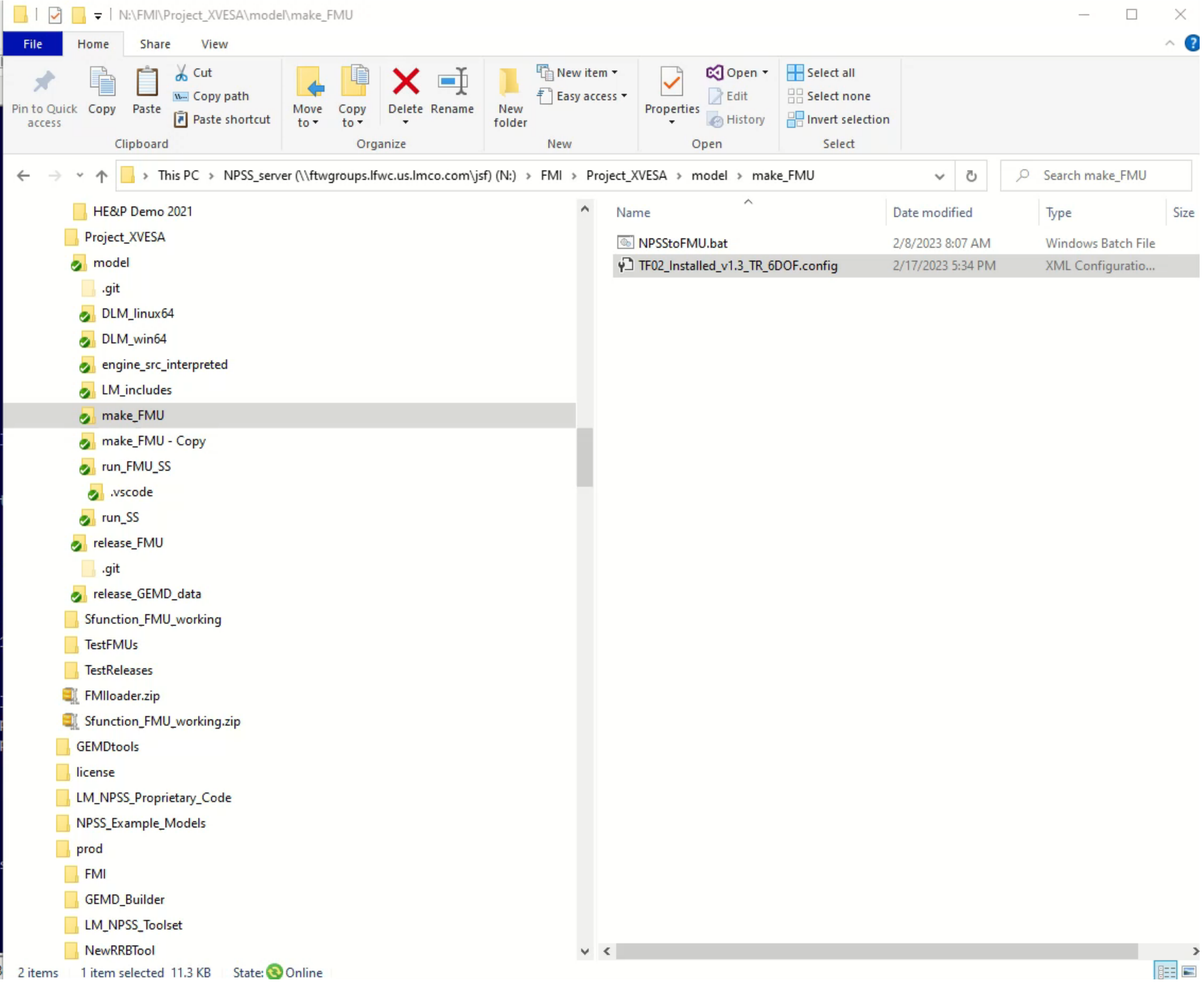
Dassault OSS Python FMPy tool



Simulink FMU Block

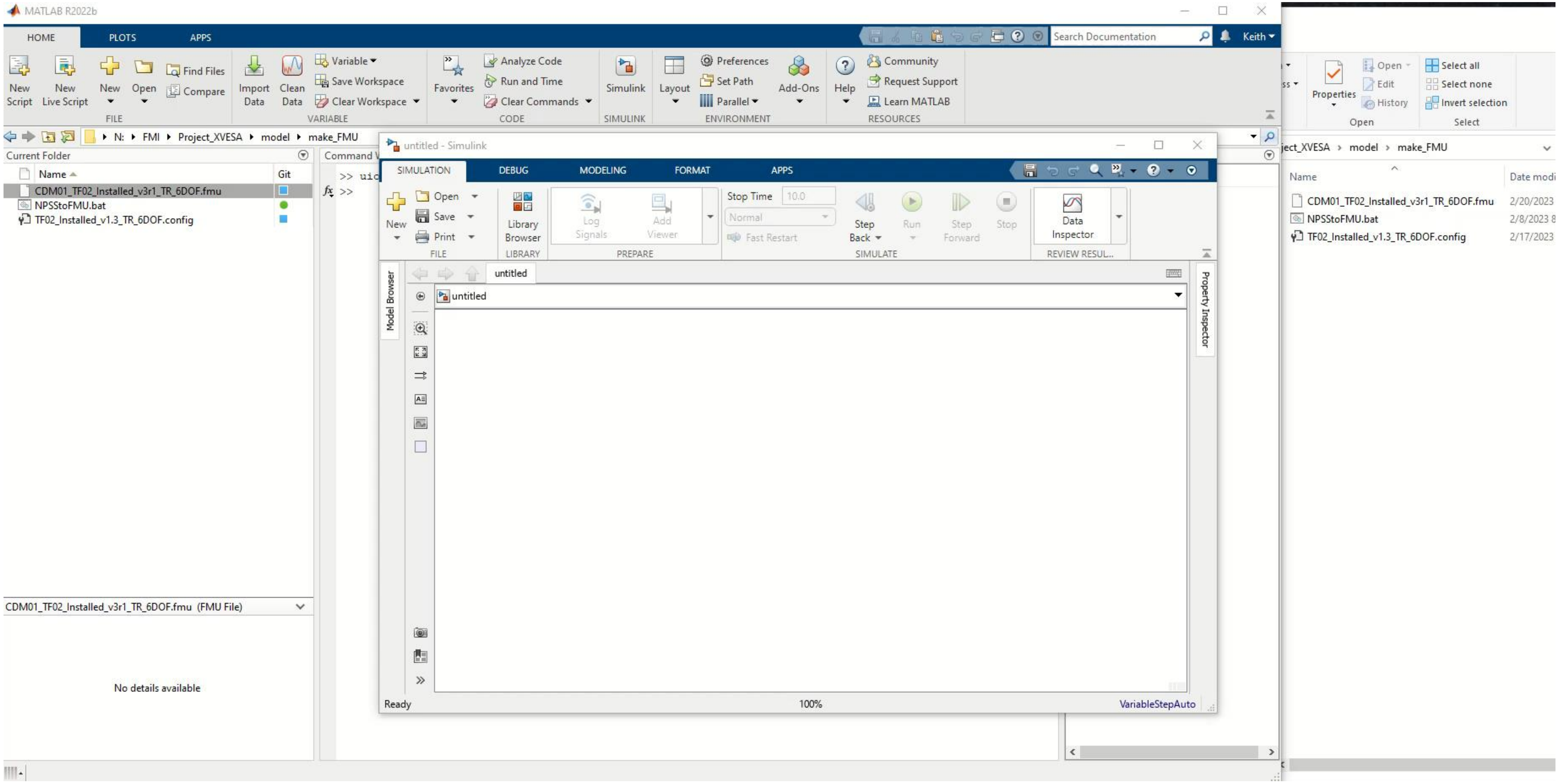
NPSS FMU Creation Demo





LOCKHEED MARTIN

NPSS FMU Execution Demo



NPSS FMU in Simulink Benefits and Features

- Integration time of NPSS model greatly reduced
- MathWorks provided preview of Simulink feature that auto codes the interface to the FMU using the Embedded Coder. Was able to autogenerate the interface code about the TF02 FMU.
- FMU tracks other information about the variables,
    - Units attribute can be set, but some differences in the string syntax as shown to the right.
    - FMI / NPSS / Simulink have some minor differences in unit string definitions, but can be handled in the Model description.XML interface.
    - Description also a useful attribute that can transfer over from the NPSS model.

Table B1 - Fundamental Unit Strings

Item	Units	Unit Strings	Unit Type	Simulink	FMI	NPSS
Altitude	foot	ft	U.S. Customary	ft	ft	ft
	meter	m	Preferred SI	m	m	m
Angle	degree	deg	U.S. Customary	deg	deg	deg
	revolution	rev	U.S. Customary	rev		rev
	radian	rad	Preferred SI	rad		rad
Frequency	Hertz	Hz (kHz, MHz, GHz)		Hz		Hz
Force	pound force	lbf	U.S. Customary	lbf		lbf
	kilonewton	kN (N)	Preferred SI	N		kN
Length	foot	ft	U.S. Customary	ft		ft
	inch	in	U.S. Customary	in		in
	mile	mi	U.S. Customary	mi		mi
	nautical mile	nmi	U.S. Customary	M		M
	meter	m (mm, cm, m, km)	Preferred SI	m	m	m
Mass	pound mass	lbm	U.S. Customary	lbm		lbm
	slug	slug	U.S. Customary			
	kilogram	kg (mg, g, kg)	Preferred SI	kg	kg	kg
Pressure	pound force per square inch	psi or lbf/in^2	U.S. Customary	psi,psia,psig		psi,psia,psig
	kilopascal	kPa (Pa, MPa)	Preferred SI	kPa		kPa
Power	horsepower	hp	U.S. Customary	hp		hp
	kilowatt	kW (W)	Preferred SI	kW W		kW
Rotational speed	revolution per minute	rpm or rev/min	U.S. Customary	rpm		rpm
	radian per second	rad/sec	Preferred SI	rad/s		rad/sec
Temperature	degree Rankine	degR	U.S. Customary	degR		R
	degree Fahrenheit	degF	U.S. Customary	degF		F
	Kelvin	K	SI and Imperial	K	K	K
	degree Celsius	degC	Preferred SI	degC		C
Time	millisecond	msec		ms		
	second	sec		s	s	sec
	minute	min		min		
	hour	hr		h		
Velocity	foot per second	ft/sec	U.S. Customary	ft/s		ft/sec
	nautical mile per hour	knot or nmi/hr	U.S. Customary	kn		knot
	meter per second	m/sec	Preferred SI	m/s		
Work	British thermal unit	BTU	U.S. Customary	Btu		Btu
	kilojoule	kJ (J)	Preferred SI	kJ J		Kj
	Celsius heat unit	CHU	Imperial			
Non-dimensional	e.g., Mach, Bypass Ratio, etc.	none				none

Closing remarks

- FMI seems to be making a big impact in the simulation interoperability world. Numerical simulations like NPSS are easily packaged into FMUs and provide interoperability with a number of Toolsets.
- Need to evaluate FMI 3 standard and its impacts on all interfaces.
- Desire for variable timestep capability supposed supported in FMI 3.
- FMU packaged model provides an easier package to configuration manage, deploy and track with PLM tools.
- FMUs also easy to execute thru python scripts allowing automated regression testing and PLM requirements testing.
- For non Consortium members Contact the NPSS Consortium for further information.

LOCKHEED MARTIN